NATIONAL UNIVERSITY OF IRELAND, GALWAY

DOCTORAL THESIS

Adaptive Knowledge Extraction and Loose Semantic Coupling in Multimedia Publish/Subscribe Overlay Networks

Author:
Tarek ZAAROUR

Supervisor:
Prof. Edward CURRY

Examiners:
Prof. Boris Koldehofe
Prof. John Breslin

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Insight SFI Research Centre for Data Analytics
College of Science and Engineering

April 2022

Declaration of Authorship

I, Tarek Zaarour, declare that this thesis titled, 'Adaptive Knowledge Extraction and Loose Semantic Coupling in Multimedia Publish/Subscribe Overlay Networks' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Tarek Zaarour

Date: 07 March 2022

“I… a universe of atoms, an atom in the universe.”

Richard P. Feynman

NATIONAL UNIVERSITY OF IRELAND GALWAY

Abstract

College of Science and Engineering
Insight SFI Research Centre for Data Analytics
Data Science Institute

Doctor of Philosophy

Adaptive Knowledge Extraction and Loose Semantic Coupling in Multimedia Publish/Subscribe Overlay Networks

by Tarek Zaarour

The primary use of communication networks today has shifted towards content distribution despite being originally designed for conversations between communicating endpoints. The massive amounts of data generated by technologies such as the Internet of Things (IoT) and Social Networks have resulted in group communication being the prominent way to disseminate information as event notifications. The recent emergence of multimedia-based services and applications in the IoT has given rise to a new paradigm coined the Internet of Multimedia Things (IoMT). In the IoMT, smart heterogeneous multimedia things interact and cooperate with one another and with other things connected to the Internet to facilitate multimedia based services and applications.

Connecting sensing devices to the internet is at the core of many IoMT applications and plays an immediate role in facilitating the acquisition and on-the-fly processing of data for real-time applications such as Autonomous Driving, Traffic Management, and Smart Agriculture. The scalability and interoperability offered by the publish/subscribe communication pattern has proved beneficial for connecting heterogeneous sensing devices and actuators to each other and to the internet. In this thesis, I focus on offering a group communication service that targets the dissemination of unstructured multimedia content such as images, videos, or audio, to a diaspora of applications at internet-scale.

Current implementations of distributed publish/subscribe systems take the form of application-level overlays that span two main infrastructures, namely, semi-centralised broker overlays and fully decentralized peer-to-peer overlays. The problem at hand entails integrating knowledge extractors, i.e. Operators, into the event processing pipeline of distributed publish/subscribe systems. These operators are in charge of uncovering meaningful semantic concepts from the multimedia data being disseminated. Extracted

concepts can then be evaluated against expressions submitted by subscribers as in conventional content-based event matching approaches. In the case of federated broker overlays, the closed nature of the system, and the availability of a single entity with explicit control over nodes in the overlay allows for the integration and management of such operators. We propose embedding light-weight binary filters (i.e. image classifiers) into the forwarding function of brokers organized into a federated overlay. We formulate the shared-filer placement and ordering problem and we provision event matching and routing algorithms for the adaptive ordering and distribution of classifier executions along paths towards interested subscribers.

In the case of peer-to-peer overlays, the decentralized and highly scalable nature of structured peer-to-peer networks makes them a great fit for facilitating the interaction and exchange of information between dynamic and geographically dispersed autonomous entities. The decoupled nature of publish/subscribe systems exacerbated by the decentralized and large-scale nature of peer-to-peer networks produces a semantic gap between publishers and subscribers. More precisely, the large semantic space of human-level recognition creates a very large data space of object labels, attributes, and relationships. The scale of the content space makes it nearly impossible for participants to agree on a bounded set of terms for subscribers to express their exact interests. We identify an inherent limitation of peer-to-peer networks lying in the exact-match property of their key-based routing primitives. We propose an approximate matching model where participants agree on a distributional model of word meaning that maps terms to a vector space. We overcome the exact-match limitation by proposing a novel distributed lookup protocol and algorithm to construct a peer-to-peer network and route content. We replace conventional logical key spaces with a high-dimensional vector space that preserves the semantic properties of the data being mapped. We further propose methods to partition the space, construct a semantic DHT via bootstrapping, perform approximate semantic lookup operations, and cluster nodes based on their shared interests.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Edward Curry for his continuous support and guidance. Ed has always been a great mentor. His constructive and valuable feedback, constant encouragement, patience and enthusiasm towards research assisted me throughout this journey. I would like to thank my friends, lab mates, colleagues and research team, Asra, Piyush, and Felipe, thank you all for your support, feedback, and for all the interesting discussions we had. My appreciation also goes out to the people of the Data Science Institute who are too many to name, thank you all for creating an environment that celebrates diversity, excellence in research, and respect, and for creating a relaxed and productive atmosphere. I would also like to give a special thanks to Brendan Smith who gave me the opportunity to get involved in various public engagement and science communication activities.

To my friends, lab mates, and colleagues, to Rajdeep, Heike, Ihab, Sameh, Chrisa, Duy, Bernardo, Annanda, and Alessia, Thank you for the walks and the talks, for the lunch breaks and the birthday cakes, for the coffee boosts and the Foosball escapes. Most importantly, thank you for helping me figure out why my code broke on many occasions, for telling me about the great open source tools, and for the cool shortcuts and the much needed resources, thank you for sharing this journey with me and making it easier for all of us.

To my family, I want to express my gratitude to my Mom and Dad. I would not have made it this far without their support and love. They created an environment for me and my siblings with a focus on respect, learning, and hard work. I would like to thank my brother Ahmad for his constant encouragement and for always reminding me that anything can be achieved with resilience and consistency. Thanks to my sisters Faten and Marwa for their continuous encouragement, for always being there, and for all the warm hugs. I also want to thank my nieces and nephew Sarah, Rina, and Ali for all the stories and the playtime.

To Mom, Dad, Ahmad, Faten, & Marwa

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Introduction

The advent of smart environments is creating swarms of data streams that are actively being processed to derive useful information about our world. Powered by a digital infrastructure that combines advances in data communication and sharing, stream and event processing, and artificial intelligence, we are able to enrich our daily decision making processes with real-time insights from the environment surrounding us [10, 11]. The early days of the internet were characterised by the ability to create a computer network that facilitates file sharing between two communicating end-points. The introduction of the world wide web, followed by modern distributed internet applications, e.g., social networks, e-commerce, and most recently, the Internet of Things (IoT), has led to necessary paradigm shifts in the data ecosystem surrounding intelligent systems in smart environments [12]. The production and consumption of information is now at the extremities of a complex system encompassing heterogeneous data sources, intelligent processing units, advanced communication protocols, and an ever-increasing number of data consumers. It is safe to say that the primary use of communication networks today has shifted towards data distribution despite being originally designed for conversations between communicating end-points.

The data ecosystem surrounding smart environments is characterized by a plethora of administrative operations that constitute the pillars of any large-scale data processing pipeline. Such operations include data sourcing, cleansing, storing, processing, and delivery. Extracting the most value from smart environments requires provisioning on-the-fly data processing pipelines that ensure the timely delivery of intermediary results and final outcomes to a plethora of real-time applications. At either end of the pipeline, and across its different stages, often sits a message broker that enables data marshalling and communication by crossing network and platform boundaries. Commercial message brokers such as Apache Kafka [13] and RabbitMQ [14] rose in popularity over the past decade. They implement the publish/subscribe communication pattern and facilitate delivering high volumes of messages coined events at low latency. Many research efforts have also culminated in a plethora of distributed publish/subscribe solutions. Popular systems include, SIENA [15], PADRES [16], HERMES [17], Scribe [18], and Poldercast [19].

The level of abstraction publish/subscribe messaging adds on top of computer communication networks provides for better scalability and flexibility when bridging the interoperable communications gap between geographically dispersed and heterogeneous data producers and consumers [20–22]. The characteristics of publish/subscribe messaging that make it attractive for smart environments are its support for unified messaging, dynamic scaling, on-the-fly and low latency processing of information, loose coupling between publishers and subscribers, and an inherent support for many-to-many communication [23].

Unstructured and semi-structured data now make up an estimated 80% of data collected by enterprises [24]. This stems from the rise of mobile devices, sensors, wearables, and the Internet of Multimedia Things (IoMT). The IoMT is an emerging paradigm in which smart heterogeneous multimedia things can interact and cooperate with one another and with other things connected to the Internet to facilitate multimedia based services and applications [25]. The unstructured and content-rich nature of multimedia data, e.g. images, audio, and videos, necessitates the introduction of integrative frameworks that focus on the extraction and delivery of useful semantic information. Advances in Computer Vision [26], Speech Recognition [27], Natural Language Processing (NLP) [28], Pattern Recognition [29], and Cognitive Analytics [30] are helping researchers and enterprises derive insights from unstructured multimedia data every day.

The recent shift in the data landscape, driven by the enormous growth of unstructured multimedia data, is posing new challenges in wide-scale data retrieval and dissemination. Real-time multimedia applications and services, e.g., Autonomous Driving, Smart Agriculture, and PPE Compliance, can be greatly enhanced by the asynchronous and decoupled communication paradigm offered by publish/subscribe messaging systems. Nonetheless, current distributed publish/subscribe systems lack native support of unstructured multimedia event types from the IoMT. We recognise that attempting to offer support for multimedia content dissemination while maintaining high expressiveness and scalability is a challenging task. In particular, input multimedia events lack any pre-defined or agreed upon schema and semantics. Extracting useful semantic information from multimedia content often requires costly pre-processing and knowledge

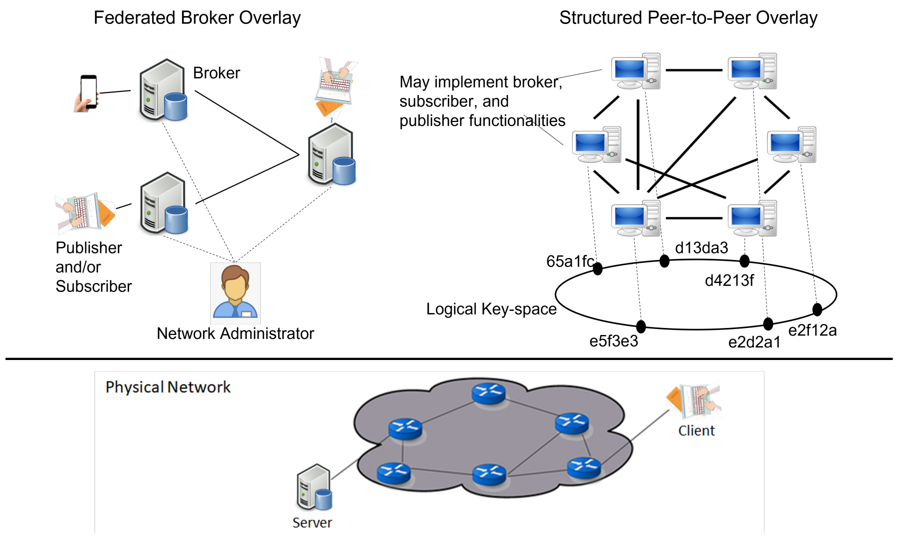


FIGURE 1.1: Overlay Network Infrastructures

extractions steps, which may have an impact on event delivery latency and overall system throughput. This aspect is further exacerbated by the wide semantic spectrum of human-recognizable concepts that can be stemmed from multimedia content, which in turn gives rise to a semantic boundary between event producers and consumers in the IoMT.

1.2 Distributed Publish/Subscribe Systems

A distributed publish/subscribe system is often implemented as an application-level overlay of nodes, connected via logical links, on top of an underlying physical communication network. Communicating end-points interact by registering themselves as either *publishers* of information or *subscribers* interested in receiving information. The overlay is responsible for managing users' queries (i.e. subscriptions), matching published events against registered subscriptions, and disseminating events as notifications to interested subscribers [31]. Participants are decoupled in space, time, and synchronization, meaning that they do not need to know of each other, they do not need to be active at the same time, and they are not blocked while producing or consuming events [23]. Such overlays are characterized by the organization of nodes in the network [32], the employed event matching model, and the employed event and subscription routing strategies. Fig. 1.1 shows two types of overlay network infrastructures commonly used to implement distributed publish/subscribe systems, namely, semi-centralized federated broker overlays and fully decentralized structured peer-to-peer overlays.

- **Federated broker overlays** consist of dedicated servers referred to as brokers. Each broker implements event matching and routing functionalities. Clients can access the system through any broker as publishers and/or subscribers and each broker stores a subset of or all the subscriptions that exist in the system. The topology is assumed to be managed by a single administrator and can often scale up to dozens of brokers. Popular content-based solutions include SIENA [33] and PADRES [16]. It is noteworthy to mention that the overlay can also be comprised of a single broker, which relaxes the need for event and subscription routing strategies and shifts the focus toward event matching.

- **Structured peer-to-peer overlays** are built on the basis of Distributed Hash Tables (DHTs) which offer key-based routing primitives (KBRs) to store and retrieve key:value pairs. A DHT organises nodes over a virtual key-space; each node is assigned a unique key and is responsible for a partition of the space with well defined boundaries. Exchanged data is also mapped to the same key-space and assigned to nodes in the form of key:value pairs. Structured peer-to-peer networks tend to be highly scalable and self-organizing, coping naturally with node arrivals and departures, and are resilient to node failures. The network usually scales up to thousands of nodes where each node may act as a client or a server. Such overlays have been commonly used to implement both, topic-based (e.g., Scribe [18] and Bayeux [34]), and content-based (e.g., Hermes [17] and Meghdoot [35]), publish/subscribe systems.

The design and implementation of a publish/subscribe node varies among systems depending on two main properties, namely, expressiveness and scalability. The level of expressiveness offered to subscribers is generally bounded by the data model used to express events. For instance, in the topic-based model, events are grouped under specific key-words (i.e. topics) where subscribers register their interest in a specific topic and receive relevant events. Such systems tend to be highly scalable due to their resemblance to multicast routing where messages are addressed to a group of recipients. In the content-based model, events are often comprised of attribute-value pairs where subscribers specify predicates over the values of attributes present in events. The content-based model is less scalable but more expressive by allowing subscribers to impose constraints on the contents of events they wish to receive.

1.3 Motivation and Problem Overview

We present a smart city scenario where we have a network of cameras observing streets, highways, and public spaces. Cameras are situated at different locations and angles,

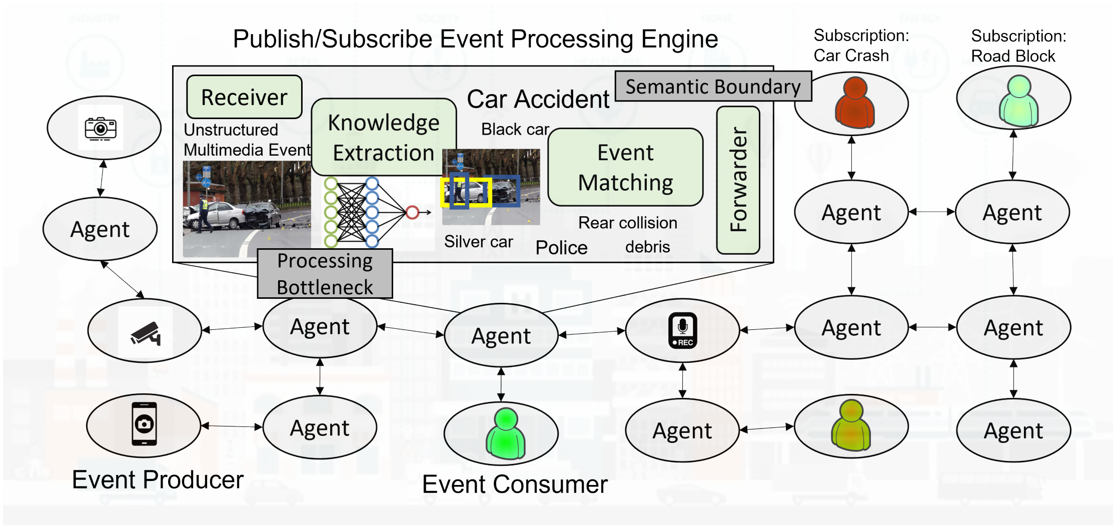


FIGURE 1.2: Multimedia knowledge extraction and sharing in wide-scale IoMT smart environments via the cooperation of asynchronous agents comprising IoMT publishers (Event Producers), IoMT subscribers (Event Consumers), AI knowledge extractors, and publish/subscribe event processing engines.

directed at traffic, pedestrian crossings, pavements, cycling lanes, and parks. We have various entities, e.g., traffic control authorities, emergency response teams, urban management or PPE compliance authorities, looking to analyze continuous video streams from around the city. The different services involved may have varying interests ranging from traffic management, and finding parking spaces, to optimizing refuse collection, and monitoring PPE compliance. Traffic accidents, formation of crowds, road works, parking violations, and road blocks, are all examples of events that occur often, at different locations throughout the city. Such events may occur in the same or different video streams, either coincide with each other, such as a traffic accident caused by a road block, or occur separately at different times or locations.

The extraction of meaningful semantic information from video streams requires a knowledge extraction step that materializes all the necessary objects labels, attributes, and relationships [36, 37]. Applying ML-based knowledge extractors on images, videos, or documents (unstructured texts), produces a set of descriptive textual labels that can then be evaluated against user queries. Currently, each of the different entities use monolithic custom solutions with specialized knowledge extractors. In many cases the queries are complex and likely to use multiple classifiers and feature extractors [38]. The wide semantic spectrum of human-level recognition results in a very large content space of semantic concepts to be extracted from incoming video streams. The scale of the content space increases the cost of maintainability of specialized and domain-specific classifiers and object detectors.

To this end, the integration of different services and applications is a crucial step toward maximising the city's operational efficiency through the timely detection and response to different types of multimedia events. We envision a cooperative multi-agent network comprising event producers, event consumers, and agents equipped with intelligent knowledge extraction and publish/subscribe messaging capabilities (Fig. 1.2). The asynchronous publish/subscribe communication layer defines a system as a composition of autonomous agents running in parallel and interacting while sharing information [39]. A network of message brokers can be leveraged to filter out unwanted data items, eliminate redundant processing, enable dynamic scaling, and disseminate popular multimedia events to a large number of users and applications [40].

Deep learning has shown to be highly effective at various multimedia data mining tasks, e.g., image classification, object detection, and motion detection [41–45]. However, employing advanced deep neural network architectures for inference is a costly process requiring expensive resources such as GPU equipped devices or server clusters in cloud environments [46]. Such models may pose as processing bottlenecks that increase end-to-end event processing and delivery latency. One can adopt the edge-fog-cloud computing paradigms as enablers for AI services in IoMT smart environments. For instance, resource constrained edge devices may implement IoMT publisher or subscriber functionalities, whereas, fog and cloud nodes may implement more resource intensive tasks such as knowledge extraction or event processing. However, in an attempt to reduce the transfer of bulky and bandwidth intensive multimedia content to the cloud, edge and fog nodes should also implement optimised and resource-aware knowledge extraction tasks. The processing workload should be intelligently distributed along the continuum between the things and the cloud with the goal of minimizing event delivery latency and maximizing overall system throughput.

Finally, allowing autonomous agents (i.e. fog and edge nodes) to participate in the knowledge extraction process gives rise to a semantic boundary between IoMT publishers and subscribers. The cooperation and exchange of information between heterogeneous and autonomous agents is limited by the wide semantic spectrum of human recognizable concepts that can be extracted from multimedia content [47, 48]. As depicted in Fig. 1.2, an image of a car accident may be tagged by the terms "car accident", "black car", "silver car", "police", "debris", and "rear collision". Such labels can be perceived as highly descriptive but might still fail to exactly match the terms present in subscriptions. Users may use different terms to refer to the same semantic concepts. The system should be able to identify the conceptual associations between subscriptions for a "car crash" or "road block" with the labels "car accident" and "debris" respectively.

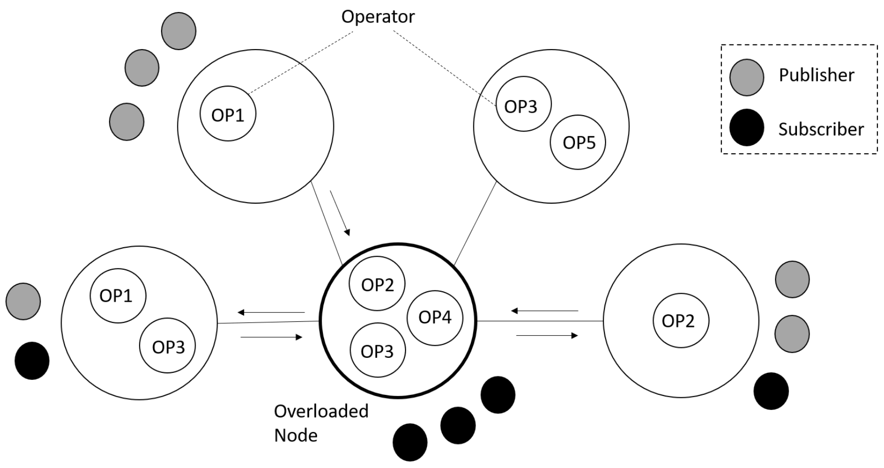


FIGURE 1.3: The integration, placement, and optimization of knowledge extractors (i.e. operators) in federated broker overlays

1.4 Problem Description

As discussed in section 1.2, content-based publish/subscribe overlays often organize nodes on the basis of two general infrastructures, namely, federated broker overlays and structured peer-to-peer overlays. To support multimedia event types under a federated broker overlay infrastructure, the static nature of the network and the presence of an administrative entity with explicit control over network, allows for the integration and management of specialized knowledge extractors. This would aid the system with labelling multimedia data for targeted predicate-based event matching and dissemination. On the contrary, under a peer-to-peer infrastructure, the scale and dynamics of the network, with autonomous nodes constantly joining and leaving, limits the ability to integrate and manage operators that extract semantic concepts. Participating publisher nodes or intermediary autonomous nodes would have to manage their own operators. The technical challenges that arise are outlined below.

Challenge 1 - Content-based matching and routing of multimedia event types in federated broker overlays: The first challenge concerns the integration of fast and resource-aware operators into the forwarding function of distributed publish/subscribe systems comprising federated broker overlays (Fig. 1.3). The goal is to extract high-level semantic concepts from unstructured multimedia events, on the fly, while avoiding bottlenecks and maintaining the scalability of the system. In a distributed processing environment, the static placement of query operators is prone to unpredictable factors such as uneven load-balance and data arrival rates. The multimedia event processing workload should be decomposed and adaptively distributed subject to dynamics in user subscriptions. The nature of the problem taps into well-known research problems such

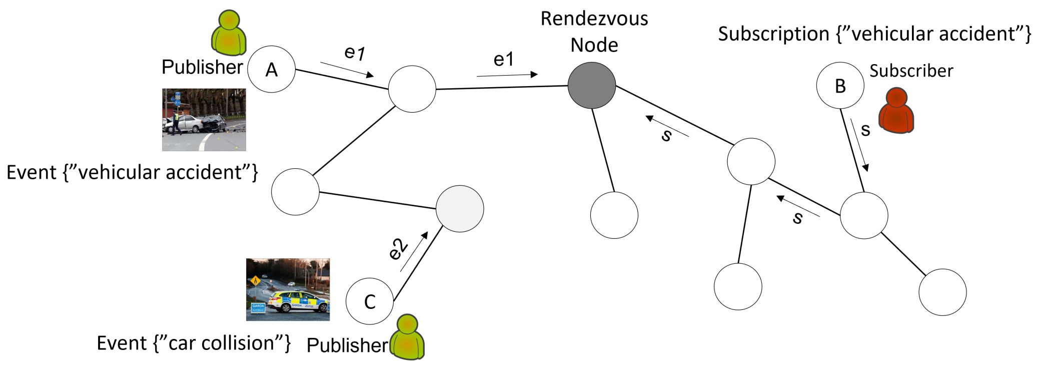


FIGURE 1.4: Semantic coupling in rendezvous routing atop structured peer-to-peer overlays

as the multi-operator placement problem [49], operator ordering in distributed stream processing systems [50], and service-chaining in software defined networks [51].

Challenge 2 - Semantic data integration and exchange in structured peer-to-peer overlays: In a peer-to-peer setting, allowing publisher or intermediary nodes to manage their own knowledge extractors may induce high semantic heterogeneity. This aspect results in routing challenges stemming from the wide semantic spectrum of human-level recognition. More precisely, many semantic concepts can be described using different terms. For example, a car 'collision' can also be referred to as a car 'crash' or 'accident'. Current structured peer-to-peer publish/subscribe systems leverage the exact-match lookup functionality offered by underling DHTs to map matching events and subscriptions to the same nodes in the network [17, 18, 35]. This property falls short when participants use different terms to refer to the same semantic concepts [52–54]. As depicted in Fig. 1.4, an image labelled with the term 'car collision' would be mapped to a random key and routed to the node responsible for the key-space partition where the key falls. On the other end, a subscription that includes the search term "vehicular accident" would be mapped to a totally different location in the network. Thus, this challenge entails overcoming the tight semantic coupling imposed by DHTs in structured peer-to-peer overlays.

Challenge 3 - Content-based matching and routing of multimedia event types in structured peer-to-peer overlays: This challenge concerns matching event and subscription semantics under a structured peer-to-peer overlay setting. Multimedia event types are known to be content-rich. For instance, Object Detection is a common knowledge extraction task in Computer Vision where a computer analyses an image and places bounding boxes and labels over all the different objects that appear in the image. A subscriber might be interested in a subset of or in all the identified objects. This aspect results a content-based event matching and routing challenge stemming from the lack of well defined domains of values (i.e. semantic concepts) and the unpredictable dimensionality of subscriptions and events [55–57]. The matching challenge is further exacerbated by the semantic heterogeneity arising from the wide semantic spectrum of human-level recognition. This challenge entails provisioning peer-to-peer content-based event and subscription routing schemes that allow the system to map multi-label events and subscriptions containing heterogeneous and semantically related terms to the same nodes in the network. This process should then be followed by a matching scheme that assesses or quantifies the semantic relevance between events and subscriptions.

1.5 Core Requirements and Research Questions

Based on the choice of network infrastructure and the assumptions made on the entities that manage the operators responsible for extracting high-level semantic concepts. We identify three main research questions that are further elaborated into technical system requirements.

Q1. The first question is concerned with enabling multimedia event matching and routing in federated broker overlays. The question is how to integrate fast and resource-aware knowledge extractors into the forwarding function of distributed publish/subscribe systems comprising federated broker overlays while ensuring timely event matching (R1.1.), efficient routing (R1.2.), and low event processing costs (R1.3.)?

Technical system requirements:

- **R1.1. Efficient extraction of semantic concepts.** Timely and resource-aware materialization and filtering of raw images that scales well with the increase in the number of subscriptions [38, 46, 58].
- **R1.2. Decomposable workload and adaptive routing decisions.** This entails the provisioning of event matching and routing techniques that facilitate a decomposable processing workload as mutually independent operations that leverage the distributed nature of federated broker overlays [59–61].
- **R1.3. Relay-free event propagation.** This entails eliminating or minimizing the participation of relay nodes in the event matching and routing process [62–65].

Q2. The second question concerns overcoming the tight semantic coupling between participants in structured peer-to-peer overlay networks. The question is how can loose semantic coupling be achieved within the exact-match lookup functionality offered by

DHTs underlying structured peer-to-peer networks while ensuring the dynamic organization (R2.2.), allocation, and retrieval (R2.1.), of data based on their shared semantic and syntactic properties.

Technical system requirements:

- **R2.1. Effective organization and mapping of semantic data.** This entails the distributed organization of data in a way that reflects the basic meaning of data items and the semantic relationships among them [53].
- **R2.2. Efficient and effective semantically decoupled lookup operations.** This entails departing from ID-centric file sharing to loose semantic coupling in DHTs. The retrieval of data items should be facilitated via a semantic routing model that relaxes strict agreements on semantics and locates the target data items at a low cost [52].

Q3. The third question is concerned with the matching of event and subscription semantics atop a structured peer-to-peer network. The question is how to support a composite content-based subscription language over multimedia event types spanning a large semantic content space while maintaining loose semantic agreements between publishers and subscribers (R3.1.), effective event matching (R3.2), and effective and efficient event and subscription routing (R3.3.)?

Technical system requirements:

- **R3.1. Low cost in defining rules.** Low barrier to entry for participants in terms of establishing semantic agreements on terms present in exchanged events and subscriptions [55].
- **R3.2. Effectiveness in matching event and subscription semantics.** This entails the accurate assessment of the semantic relevance between events and subscriptions containing heterogeneous terms that might describe the same or highly related semantic concepts [53, 55, 56].
- **R3.3. Effectiveness and efficiency in mapping event and subscription semantics.** This entails mapping arbitrary predicates and event attributes that share semantic properties to the same nodes in a large structured peer-to-peer overlay network in a small number of routing hops and while maintaining a low messaging overhead [66–68].

1.6 Current Approaches

We identify three broad categories of relevant approaches as follows.

1.6.1 Event Matching over Federated Broker Overlays

Content-based model: In broker overlays that follow the content-based event matching and routing model, subscriptions and events are defined over a fixed schema. The main characterization of such overlays is the way events flow from senders to receivers, notifications are routed to downstream brokers in a content-centric manner where events are continuously evaluated against subscriptions registered in a broker's routing table, and only forwarded if there is at least one matching subscription downstream. The main works in this category are those by Carzaniga et al. (SIENA) [33], Fidler et al. (Padres) [69], and Cugola et al. (JEDI) [70]. They rely on a match and forward model where an event is redundantly evaluated against subscriptions at every broker in the network until it reaches matched subscribers. This performs well due to various predicate indexing and tree-based matching algorithms. However, such algorithms are of limited use for subscriptions over unstructured inputs such as images and videos. Subscription predicates would remain stuck behind relatively expensive operators used to extract high-level semantic concepts.

Overlay topological reconfiguration: Poor overlay design can play a crucial role in degrading the performance of a distributed publish/subscribe system. To simplify routing strategies and reduce overall matching costs, solutions exploit topological properties based on client distributions to enhance event delivery efficiency. Current approaches build and maintain opportunistic links on top of the managed overlay network. Such links are leveraged to shortcut paths by bypassing intermediate brokers referred to as pure forwarders. Jaeger et al. [63] and Baldoni et al. [62] classify popular and unpopular events to preserve resources for events the serve higher numbers of subscribers thus improving the overall efficiency of the system. They propose dynamic protocols that look for nodes that share similar event traffic to modify the structure of the overlay by building direct links across them thus creating common interest regionalism.

1.6.2 Relational Platforms for Large-Scale Video Analytics

Modern systems for querying the content of images and videos rely on extraction techniques based on Deep Convolutional Neural Networks (CNNs). CNNs are known to be highly effective in common Computer Vision tasks such as Image Classification and

Object Detection. Lu et al. [37, 38] and Kang et al. [46] focus on accelerating the knowledge extraction process by designing probabilistic and specialized classifiers that are then applied in a manner similar to the conventional predicate push-down optimization. The intuition is that queried semantic concepts often occur rarely in large amounts of video data being processed by expensive CNNs. Thus, applying light-weight classifiers allows for filtering out many irrelevant frames, minimizing the processing cost, and speeding up the query answering process. In such systems, video data is collected and persisted in stable storage then analysed either by domain-specific learning models or accelerated by light-weight classifiers following a store and query processing model.

1.6.3 Event Matching over Structured Peer-to-Peer Overlays

Content-based approach: In this approach, publishers and subscribers have to establish an explicit agreement on the terms used to describe exchanged content. Structured peer-to-peer solutions, such as, Meghdoot [35] and Hermes [17], leverage the exact-match routing primitives of their underlying DHTs (i.e. CAN [71] and Pastry [6] respectively) to map exact-match subscriptions and events to the same nodes in the overlay network. This approach induces high semantic coupling between participants and would require defining a large number of rules to support multimedia content as native event types. There are a handful of content-based approaches that aim to form 'semantic' dissemination structures over unstructured peer-to-peer networks. Chand et al. [67] and Sub-2-Sub [66] , propose event routing strategies that aim to spread messages within a community that shares similar interests. Peers organise in a hierarchy that defines neighborhood relationships based on interest intersection, containment, or equivalence relationships. However, equivalence resembles an exact string match, whereas, intersection and containment are only defined for predicates over numerical value ranges.

Concept-based approach: In this approach, publishers and subscribers may use related terms to describe the same semantic concepts. Centralized solutions (systems comprised of a single broker), e.g., S-ToPSS [56] and G-ToPSS [72], made use of hierarchical semantic structures (i.e. ontologies) that link terms based on lexical and semantic relationships. They propose a Boolean semantic matching model that translates the content of incoming events to all possible synonyms and semantically related concepts before matching. In distributed peer-to-peer solutions, the exact-match limitation of DHTs has been mitigated through the use of Mapping Tables [52] and Ontology Routing Tables (ORTs) [53]. Mapping Tables [52] are designed to work over any DHT offering an exact-match key-based routing primitive. They store the correspondence between values and are shared among participating peers. Participants attempting to retrieve a file called X, consult a shared or local mapping table, to get the identifiers of X at various peers. However, the use of such tables may induce a high messaging overhead. Subscribers would be forced to initiate many lookup operations by translating the contents of their subscriptions to all possible semantic representations. On the other hand, Ontology Routing Tables, JTangCSPS [53], are designed to work over the Pastry DHT. They work over a semantic broker overlay built on the basis of an Ontology Class Weighted Tree (OCWT). The tree is mapped to a 1-dimensional space where each class is assigned a unique key. An Ontology Routing Table stores the IDs of ontology classes in different *sets* that reflect the taxonomy of classes in the OCWT. The system provides logarithmic bounds for the performance of the proposed semantic routing model as a function of network size. However, participants are limited to the domains covered by the Ontology used to construct the broker overlay.

Approximate semantic matching approach: Similar to the concept-based approach, this approach allows publishers and subscribers to use related terms to describe the same semantic concepts. Hasan et al. [55], propose an approximate semantic event processing model that leverages vector space semantics to perform pairwise comparisons between the vector representations of terms present in events and subscriptions. However, this approach is yet to be explored as a solution to the same problem under a decentralized infrastructure. We recognise that the mathematical richness of high-dimensional vector spaces and their scalability in terms of their semantic coverage render them an interesting candidate to support approximate semantic event matching in peer-to-peer publish/subscribe systems.

1.7 Proposed Approach

Our proposed approach comprises a generalizable light-weight image classification model and its integration into the event processing pipeline of federated broker overlays and a semantically loosely coupled and approximate event routing and matching model over structured peer-to-peer overlays networks. These models can be conceptually decomposed as follows (Fig. 1.5):

1.7.1 Adaptive Filtering of Multimedia Content over Federated Broker Overlays

Light-weight Binary Classifiers (Q1, R1.1.): This step consists of embedding light-weight binary filters (i.e. image classifiers), constructed using efficient neural network architectures, into the forwarding function of brokers in an overlay network. A set of binary classifiers serve as a generalizable and resource-aware model that facilitates the

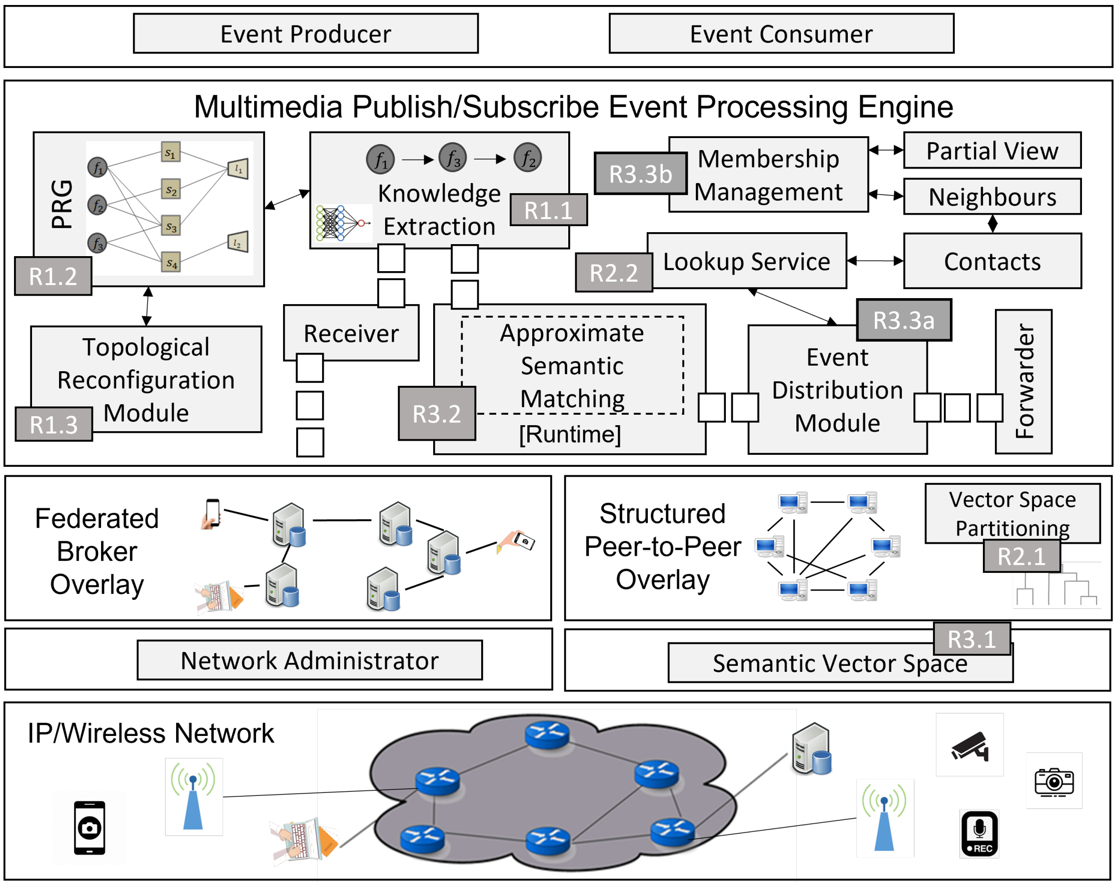


Figure 1.5: A Layered, distributed, and approximate semantic publish/subscribe multimedia event processing model

fast extraction of high-level semantic concepts from unstructured multimedia events. The intuition is that the cooperation of multiple binary classifiers provides for a simple and broadly applicable knowledge extraction technique that allows for extending the semantic coverage of the system's matching engine by provisioning new classifiers as needed to support a continuously evolving content-space.

Distributed shared-filer ordering (Q1, R1.2.): Broker overlays often serve high numbers of subscriptions. To maintain system scalability against high workloads, we propose a language model comprising a conjunction of binary filters (i.e. binary image classifiers). We model the relationship between filters, subscriptions, and links, into a Publication Routing Graph (PRG). We propose a multimedia event matching and routing model that exploits the order of execution and placement of filters, based on their selectivity, popularity (PRG edge converge), and execution cost, to reduce overall processing costs and minimize event matching and delivery latency.

Topological reconfiguration based on filter distributions (Q1, R1.3.): The structure of the broker overlay can influence the performance of the system in terms of matching and dissemination efficiency. We measure traffic similarity across brokers as in previous approaches. We also propose several broker workload similarity estimation techniques and an overlay topological reconfiguration algorithm that leverages the distribution of clients and classifier executions across brokers to modify the overlay structure.

1.7.2 Semantic Data Integration and Exchange over Structured Peer-to-Peer Overlays

Distributional Event Semantics (Q3, R3.1.): Assuming that semantic coupling can be quantified by the number of mappings between terms, then a semantic model that condenses these mappings can be very useful. Ontological models require granular agreements on term meaning while distributional vector space models leverage the statistics of term co-occurrences in a large corpus to establish semantics. The latter allows event producers and consumers to establish a loose semantic agreement on large corpora of text. Hence, we propose an event and language model rooted in vector space semantics. Events are comprised of a set tuples, each containing a word:vector pair. Each pair collectively represents a semantic concept that was extracted from a multimedia data item. Subscriptions comprise conjunctions of predicates, each also containing a word:vector pair.

Replacing Logical Key-spaces Underlying Conventional DHTs with a High-Dimensional Semantic Vector Space (Q2, R2.1.): We employ hierarchical clustering techniques over word embeddings based on a proximity measure that allows us to group together the vector representations of terms sharing semantic properties. We dynamically assign the resulting partitions to nodes in a structured peer-to-peer network where each node maintains a partition of a large semantic vector space comprising tight grouping of vectors representing terms that have semantic relationships.

Mapping Semantic Data (Q2, R2.2., and Q3, R3.3a): We propose a lookup protocol that can be initiated by any node in the overlay. It takes the vector representation v of a data item d, and recursively finds closer nodes until it identifies the target node maintaining the vector space partition where v falls. The receiving node stores data items that share semantic properties with d. This aspect allows us to map data items that share semantic properties to the same nodes via rendezvous routing.

1.7.3 Selective Multimedia Content Dissemination over Structured Peer-to-Peer Overlays

Approximate Semantic Matching (Q3, R3.2.): We propose a matching scheme that decides on the relevance between events and subscriptions based on distributional semantics. Distributional semantics offer a way of quantifying the semantic relatedness between two linguistic items based on their distributional properties in large corpora of text. A distributional semantic model maps terms to high dimensional numerical vectors where we can decide on the relevance between terms based on how close their vector representations are oriented in space. We quantify there overall semantic relatedness between event and subscriptions by obtaining a similarity score for each pairwise mapping between the vectors in subscription predicates and event tuples. We then identify the most-likely mapping and compute an aggregated overall similarity score.

Semantic Regionalism via Gossiping (Q3, R3.3a and R3.3b): A loosely semantically coupled lookup function (R2.2), allows related terms to rendezvous at certain nodes in the overlay. However, due to the high dimensionality of the space and the unsupervised nature of hierarchical clustering, lookup operations can be highly probabilistic. Some events or subscriptions might get erroneously mapped to nodes maintaining distant semantic groupings. To mitigate this issue, we propose a membership management protocol that helps constrain the propagation of events and increase the probability of a match by maintaining sub-overlays of nodes that share semantic properties. The event distribution module (R3.3a) first maps the terms present in events and subscription to nodes in the network via the native lookup service (R2.2.). Upon reception of an event e, the event distribution module forwards e to neighbours hosting similar subscriptions by consulting the local neighbour table, which is in turn populated by the membership management service (R3.3b)

1.8 Research Methodology

The methodology followed in the work contains the following steps:

1. Review the literature and define research problems associated with supporting multimedia data as native events types in distributed publish/subscribe systems.

2. Formulate the research questions and technical requirements for supporting multimedia event types in distributed publish/subscribe systems under two types of overlay infrastructures, namely, federated broker overlays and structured peer-to-peer overlays.

3. Formulate the main hypothesis to answer the research questions and design the experiments for testing.

4. Design and implement the simulation environment for constructing and operating a dedicated broker overlay network. Real-world overlay topologies were constructed based on real-world traces taken from the public Planet Lab research network.

5. Build prototype event processing engine with light-weight binary image classifiers (i.e. filters). A filter set was synthesized based on extensive prototyping with real image classifiers and test data from ImageNet.

6. Implement event processing components for a federated broker overlay infrastructure. That includes, an event matching algorithm, a routing protocol, and an overlay adaptation protocol and algorithm.

7. Design and implement a framework for experimental evaluation, identify metrics, introduce event and subscription workloads, implement a suitable baseline, and obtain results.

8. Setup simulation environment for constructing and operating a structured peer-to-peer overlay network. Synthesize an evaluation event set by semantically expanding a seed event set comprising image labels from well-known training datasets and create a subscription set from the generated event set.

9. Build a structured peer-to-peer network on the basis a high-dimensional semantic vector space. This entails obtaining word embeddings (i.e. vectors), and implementing, a hierarchical clustering algorithm to partition the vector space, the overlay construction algorithm and protocol, and a lookup protocol that maps data items to nodes in the network.

10. Implement event processing components for a structured peer-to-peer overlay network infrastructure. That includes, an event matching algorithm, event and subscription management components, a routing protocol, and a membership management protocol.

11. Design and implement a framework for experimental evaluation, identify metrics, introduce event and subscription workloads, implement a suitable baseline, and obtain results.

12. Recognize the trade-offs and limitations of the proposed approaches and provide comparative analysis with current matching and routing strategies in distributed publish/subscribe systems.

1.9 Contributions

- The integration and continuous provisioning of fast and resource-efficient binary classifiers as Boolean event filtering operators in distributed publish/subscribe systems comprising federated broker overlays. We propose adaptive and distributed event matching algorithms and routing protocols (MEC and EMEC [40]) that exploit the ordering and placement of classifier executions across brokers.
    - Experiments show up to 70% decrease in event delivery latency when compared to performing filtering operations in publisher-end brokers.
    - Experiments also show over 80% decrease is processing redundancy overhead when compared with deferring all computation to subscriber-end brokers.
- We modify the topological organization of brokers in an overlay network in response to workload variations across brokers. We propose broker workload similarity measures based on shared filter executions (PRG similarity).
    - Experiments show up to 20% decrease in processing redundancy overhead when compared to utilizing shared event traffic (Jaccard index).
- We address semantic coupling in structured peer-to-peer networks based on DHTs by replacing conventional logical key-spaces with a high-dimensional semantic vector space. We propose techniques based on hierarchical agglomerative clustering to partition the space. The resulting partitions comprise traversable groupings of word embeddings that fall in semantic proximity to each other.
    - **SemanticPeer** [47]: An algorithm and protocol to dynamically assign the resulting partitions to nodes in a large structured peer-to-peer network and to route a vector v to the node retaining the partition where v belongs in a few routing hops.
        - Experiments show that the proposed model achieves more than 97% recall in routing accuracy, that is, locating a node responsible for storing a data item in a few routing hops.
        - results also show that the network achieves over 90% recall in approximately matching two semantically related terms via rendezvous routing.
    - **OpenPubSub** [48]: A decentralized and effective content-based approximate semantic event processing model. The model uses distributional semantics and requires loose semantic agreements between participants on large corpora of texts. We augment the overlay with a hybrid peer-to-peer routing technique that combines the efficient and selective nature of rendezvous-based routing

protocols with the probabilistic and dynamic nature of gossip-based routing protocols.

  * Experiments with approximate events and subscriptions from image annotations and knowledge graphs show a 15% improvement in event matching recall as a result of utilizing shared KNN as a distance measure for hierarchical agglomerative clustering.
  * The rendezvous routing approach reduces overall messaging overhead by up to 44% and increases event matching recall by up to 53%.
  * The utilization of Gossiping reduces the overall messaging overhead by up to 59% and increases event matching recall by up to 35%.

1.10 Thesis Outline

The rest of the thesis is organized as follows:

- Chapter 2. Background and Preliminaries: This chapter discusses key concepts that make up a distributed publish/subscribe system. We introduce and motivate the event processing paradigm, the publish/subscribe communication pattern, event and language models, event matching models, overlay infrastructures, and routing protocols. We also discuss the emerging Internet of Multimedia Things and how it emphasises the need for supporting multimedia data as native event types in distributed publish/subscribe systems.

- Chapter 3. Problem and Motivation: This chapter elaborates on the problem tackled it this thesis and the research questions posed in this chapter along with their respective technical requirements. We analyse the problem, align the identified requirements with the literature, and we present current approaches and analyse their strengths and limitations.

- Chapter 4. Adaptive Filtering and Dissemination of Multimedia Content in Federated Broker Overlays: This chapter addresses research question Q1. We propose an adaptive and distributed event matching algorithm and routing protocol that leverage the distributed nature of broker overlays to manipulate the placement and order of execution of binary classifiers on the basis of their popularity, selectivity, and cost.

- Chapter 5. SemanticPeer: A Distributional Semantic Peer-to-Peer Lookup Protocol for Large Content Spaces at Internet-scale. This chapter addresses research question Q2. We propose SemanticPeer, an algorithm and protocol to construct a structured peer-to-peer network on the basis of a high-dimensional semantic vector space, and a lookup protocol that allows for the allocation and retrieval of data items on the basis of a semantic routing model that allows for the use of heterogeneous terms to describe the same semantic concepts.

- Chapter 6. OpenPubSub: Supporting Large Semantic Content Spaces in Peer-to-Peer Publish/Subscribe Systems. This chapter addresses research question Q3. We propose OpenPubSub, a peer-to-peer content-based publish/subscribe system that's equipped with an approximate semantic event matching scheme and a hybrid routing protocol that combines the selective nature of rendezvous routing protocol with the dynamic and probabilistic nature of gossiping protocols.
- Chapter 7. Conclusions and Future Work: This chapter concludes the thesis and discusses avenues for future work. We reiterate the problems tackled, align our contributions with the research questions, discuss the strengths and limitations of the proposed approaches, and finally, we align the current limitations and shortcomings with avenues for future work.

1.11 Associated Publications

- Zaarour, Tarek, Anuraag Bhattacharya, and Edward Curry. "OpenPubSub: Supporting Large Semantic Content Spaces in Peer-to-Peer Publish/Subscribe Systems for the Internet of Multimedia Things." IEEE Internet of Things Journal (2022).
- Zaarour, Tarek, and Edward Curry. "SemanticPeer: A distributional semantic peer-to-peer lookup protocol for large content spaces at internet-scale." Future Generation Computer Systems 132 (2022): 239-253.
- Zaarour, Tarek, and Edward Curry. "Adaptive filtering of visual content in distributed publish/subscribe systems." 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA). IEEE, 2019.
- Zaarour, Tarek, et al. "Automatic anomaly detection over sliding windows: Grand challenge." Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems. 2017.

Chapter 2

Background and Preliminaries

2.1 Introduction

The event processing paradigm is a prominent solution for building distributed applications that revolve around real-time data processing, analytics, and sharing. This thesis addresses research problems that concern multimedia event processing. In particular, the work deals with supporting multimedia content, i.e. images, videos, and audio, as native event types in distributed publish/subscribe systems. In this chapter, I introduce preliminaries, background concepts, and terminology. In section 2.2, I start by introducing the event processing paradigm. Sections 2.2.1 and 2.2.2 discuss event-driven architectures and prominent dedicated event processing platforms including publish/subscribe systems, data stream processing systems, and complex event processing systems. Following that, I narrow down the focus of this chapter to the publish/subscribe communication model and the problematic aspects which we identify and tackle in this thesis. More precisely, the elements of a distributed publish/subscribe system are introduced in section 2.3, that includes, event models and rule languages (section 2.3.1), overlay infrastructures, and event routing strategies (section 2.3.2). Section 2.4 focuses on the Internet of Multimedia Things and discusses how its emergence necessitates provisioning new event processing strategies and techniques to deal with the influx of large amounts of heterogeneous multimedia data traffic. I provide comparisons between multimedia data and scalar data in section 2.4.1. The functionalities that multimedia event processing platforms should implement are discussed in section 2.4.2, the need to bring the processing of such data to the edge of the network is discussed in section 2.4.3. I discuss computer vision and image understanding in section 2.4.4, and finally, section 2.5 summarises the chapter and highlights the key concepts introduced.

2.2 The Event Processing Paradigm

Event processing is a type of computing that performs operations on a continuous flow of events in order to detect and respond to occurrences of interest in a timely way [73]. Common types of operations include reading, transforming, deleting, or delivering events. An event is an encapsulated unit of information describing any form of action or occurrence that can be identified by a computer program. The conceptualization of an event is rather encompassing and generic in the sense that it can describe any form of occurrence in the real world, like a car accident or coffee being spilt, or any virtual phenomenon occurring within a computer system, such as clicking on an ad or pausing a video. Event processing concerns the representation of events in a computerized environment, their identification, processing, and delivery to interested recipients. This paradigm can be classified as a subject area similar to Database Management and Artificial Intelligence and is not to be confused with Event-based Programming [74]. The concept of an event in programming appeared far before the emergence of event processing as a stand-alone paradigm. Earlier forms of events include exceptions for run-time errors in various programming languages and user interactions with a Graphical User Interface such as dragging a mouse or pressing a button. Event processing on the other hand, tends to be more concerned with the consumption, representation, detection, and reaction to events that occur in the real-world.

2.2.1 Event-Driven Architectures

An Event-Driven Architecture (EDA) is a software design pattern that is centered around events [75, 76]. Different components of a computer program work together by producing, consuming, and reacting to event notifications. This design pattern departs from monolithic architectures by enabling the decomposition of systems into multiple services that cooperate and communicate through event notifications [75]. For instance, an online payment system can be broken down into four different services that work together to fulfil a transaction. Service A would be responsible for customer accounts, Service B for processing payments, Service C would be concerned with charge-back and dispute management, and Service D would be responsible for fraud detection. This decomposition allows businesses to develop faster and deploy more flexibly by assigning small teams of developers to build and maintain certain parts of the work and understand how they fit into the bigger picture.

Similarities can be drawn between Event-Driven Architectures and Service-Oriented Architectures (SOA) [77]. The goal of an SOA is to enable the re-usability of software components via service interfaces that are discoverable over a network. Each service in

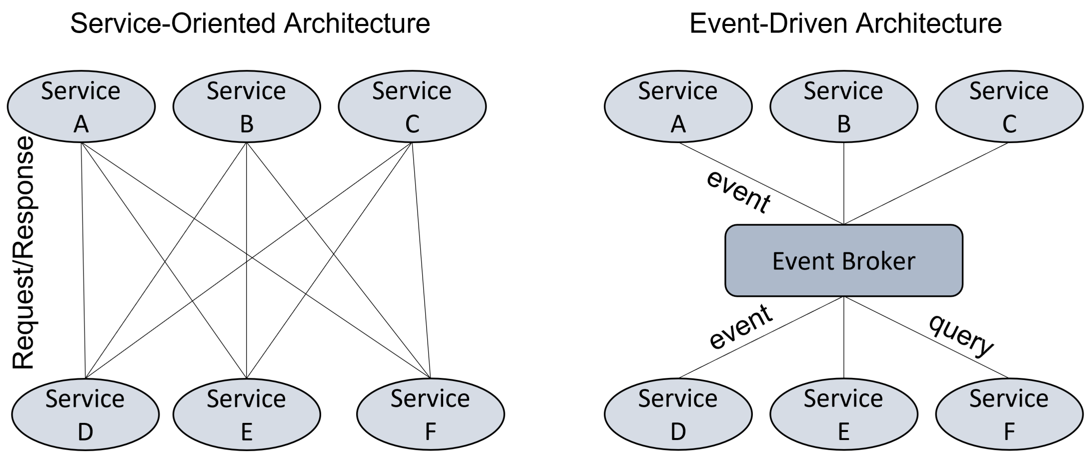


Figure 2.1: Service interactions in Service-Oriented Architectures (XML-based SOAP Requests [1]) vs. Event-Driven Architectures

an SOA implements a stand-alone functionality by housing all the data and code needed to execute a discrete business operation. As depicted in Fig. 2.1, interactions between the components of an SOA are often based on the conventional Simple Object Access Protocol (SOAP) request-response communication pattern [1]. Conversely, EDAs rely on event-driven interactions to maintain long-running asynchronous process capabilities [78]. An event-driven interaction contradicts with the conventional request-response communication pattern along two main themes. On the one hand, the latter often deals with accumulating data or data at rest while the former is mostly concerned with data in flight as its moving from one place to another. On the other hand, in a request-response interaction, a service usually submits a request in the form of an update or a query and waits or blocks until it receives a response. Whereas in an event-driven interaction, a service usually submits a long-standing query and continues to operate normally until it receives an event notification.

There are three main components that drive an EDA: event producers, event consumers, and an event broker. Components interact as follows: An event consumer submits a long-standing query signifying its interest in receiving certain events as soon as they occur. When a service performs an action that other services might be interested in, it acts as an event producer by submitting an event notification to the event broker. The broker evaluates incoming events against registered queries and notifies interested event consumers. It is noteworthy to mention that conventional event-based programming approaches allow for incorporating events to trigger processing logic without the use of dedicated event brokers. In such case, the flow of events is hard-wired following explicit event flow logic coupled with event handlers that often run in parallel, on a separate thread or processing unit, without interrupting the execution flow of the current task. This minimalist approach to event processing is further elaborated through the use of

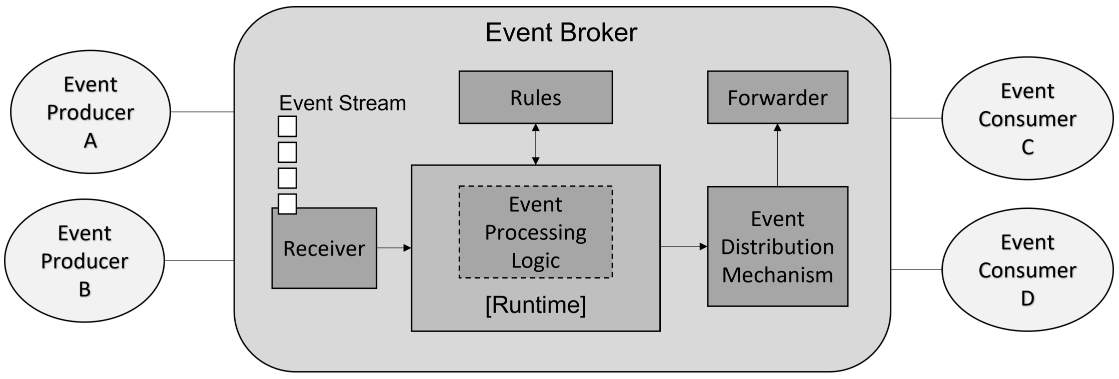


FIGURE 2.2: Dedicated Event Processing Platform High-level Architecture, Cugola et al. [2]

dedicated event processing platforms that separate the event processing logic from the event-driven environment they coordinate.

2.2.2 Dedicated Event Processing Platforms

In a distributed and event-driven processing environment, we often have multiple event producers operating asynchronously and firing events that may be of interest to one or more event consumers. The processing of events can be as simple as employing operations that select events based on certain criteria and discard irrelevant events. Or, it may involve passing an event through multiple complex phases of processing such as transforming an event instance to a new form or looking for patterns in collections of events. These aspects result in the emergence of a many-to-many communication pattern that involves various types of mediating operations to be performed over a scattered flow of events. Ensuring the dissemination of event notifications to all interested consumers demands explicit configurations for event routing throughout what can be viewed as a mesh network of services. Hard-wiring such interactions while ensuring effective and efficient event processing and distribution can become challenging, especially when the number of interacting services and the complexity of event processing operations increases.

A dedicated event processing platform offers a way for separating the event processing logic from the application logic [2]. This is facilitated through an event router or broker component that consolidates event processing operations and decouples the interaction between event producers and consumers. As depicted in Fig. 2.2, such platforms usually provide a language (i.e. rules) for expressing event processing logic, a run-time to execute event processing logic, and an event distribution mechanism. Each of these functionalities contribute to increasing the agility of building, managing, and operating event-driven architectures.

To facilitate effective event processing, high level programming abstractions are in place to allow for provisioning updates to the event processing engine for detecting various types of events. Such abstractions enable the detection of events at a higher level than those provided by conventional programming languages. In terms efficiency, having event processing logic reside outside interacting services allows for employing techniques to scale the system and to cope with an increasing number of event producers and consumers. For instance, certain costly event processing operations can be allocated additional resources to elevate bottlenecks and improve efficiency. Aside from functional requirements, a dedicated event processing platform allows users to have more control over non-functional requirements such as reliability, availability, and security.

Dedicated event processing platforms tend to be more concerned with real world events and their corresponding representation and detection within a computerized environment. Common applications include monitoring energy consumption, identifying anomalous behaviours in manufacturing equipment, and detecting traffic accidents or violations within a traffic management system. Event processing applications can be grouped into various categories such as observation, information dissemination, dynamic operational behaviour, active diagnostics, or predictive processing [73]. As such, there are several implementations of event processing platforms that usually pair with the type of event processing application they best attend to. Despite having a common goal, these systems differ in a wide range of aspects, including architecture, data models, rule languages, and processing mechanisms [2]. In what follows we outline three known solutions, namely, the publish/subscribe communication model [23], the data stream processing model [79], and the complex event processing model [80].

2.2.2.1 Publish/Subscribe Systems

The publish/subscribe communication model provides a framework for exchanging messages coined events between dispersed communicating end-points. A human agent or an automated service interact with a *broker* component by registering themselves as either *publishers* of information or *subscribers* interested in receiving information (Fig. 2.3). The system is mainly responsible for managing user subscriptions, matching incoming events against registered subscriptions, and disseminating events as notifications to interested subscribers. The matching process can either filter events on the basis of preset event groupings, e.g., Scribe [18] and Bayeax [34], or on the basis of the actual content of individual events, e.g., Siena [15], Padres [69], and Hermes [17]. Common applications fall under the observation and information dissemination categories. For instance, stock trading platforms use a publish/subscribe system to monitor the stock market and send real-time updates to traders about current bid and ask prices, trading volumes, or stock

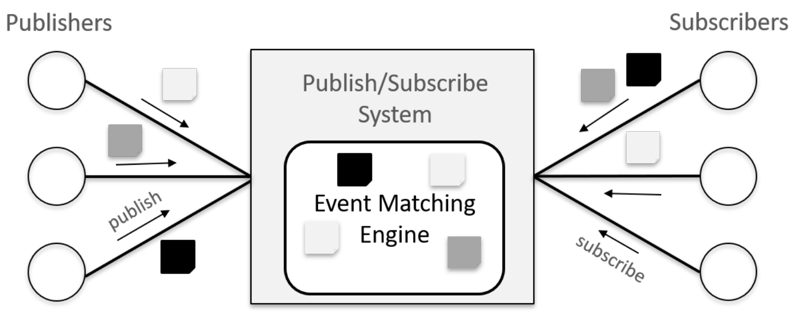


Figure 2.3: Publish/Subscribe Notification Service

indices. Traders submit subscriptions specifying certain stock names, prices ranges, or target volumes, and receive event notifications as soon as the constraints they specified are met.

2.2.2.2 Data Stream Processing Systems

Events in the real world come at different levels of abstraction. For example, an increase in energy consumption in manufacturing equipment is a high-level event that can either signify an underlying problem or can be simply due to an increase in production. A low-level event usually shows symptoms in the form of higher-level events. The real-time detection of low-level events requires processing streams of data that may contain information that we would not normally view as events. A stream processing platform provides an event processing framework based on a data flow model. Distributed data stream processing engines, such as, Borealis [81], Storm [82], Spark [83], and Flink [84], create a Directed Acyclic Graph (DAG) of operators and inject data streams into the processing graph. They expose programming abstractions based on functional programming languages to build operators. The processing pipeline follows an on-the-fly processing model comprising multiple tasks that can be performed serially, or in parallel [85]. Tasks are performed by operators that implement actions such as aggregations, transformations (e.g. map, filter, and groupBy), predictive analytics, enrichment, or ingestion. Insights drawn by the system are either disseminated to end-users in the form of push notifications, visualized via dashboards, or fed into automated services that trigger subsequent processing logic.

2.2.2.3 Complex Event Processing Systems

Complex Event Processing (CEP) is concerned with identifying and responding to low-level occurrences that present in the form complex patterns in collections of events [86, 87]. A CEP engine continuously matches incoming event streams against patterns defined by event consumers. This is facilitated via built-in operators such as time or count windows that capture and retain sequences of events over a period of time or a number of events from one or multiple event sources. Conversely to stream processing, CEP engines expose an SQL-like query language for event consumers to write queries that describe conceptual or temporal event relations. As a consequence, their processing model relies on the ability to specify composite events through event patterns that match incoming event notifications on the basis of their content and on some ordering relationships between them. For instance, a publish/subscribe system is capable of notifying subscribers about the current price of a certain stock, whereas, a CEP system would be able to identify patterns in stock price oscillations over a period of time. Such patterns allow the system to detect lower-level events that would aid applications with predicting possible future variations in the price of a stock.

2.3 Elements of a Publish/Subscribe System

The publish/subscribe communication pattern lays the foundation for event-driven interactions. It provides a simple and effective communication model that allows data producers and consumers to exchange events without the need to address each other directly. The decoupling aspect makes this type of communication a natural fit for building and operating large-scale and dynamic distributed systems. Interacting components have the ability to exchange encapsulated units of information while maintaining full decoupling across time, space, and synchronization [23]. There are several prominent research solutions that use the paradigm to tackle various types of applications ranging from information dissemination [88] and group communication [89], to information level interoperability in smart environments [90], coordinating distributed transactions in micro-service architectures [91], and workflow management [70].

A distributed publish/subscribe system is often comprised of a set of nodes over a communication network. Nodes are organized into an application-level overlay on top of an underlying physical network. The exchange of events between clients takes place through the overlay network. Nodes in the overlay match incoming event streams against managed subscriptions and route matching events towards interested subscribers. In what follows, we discuss variations and trade-offs in the design and implementation of distributed publish/subscribe systems (Fig. 2.4).

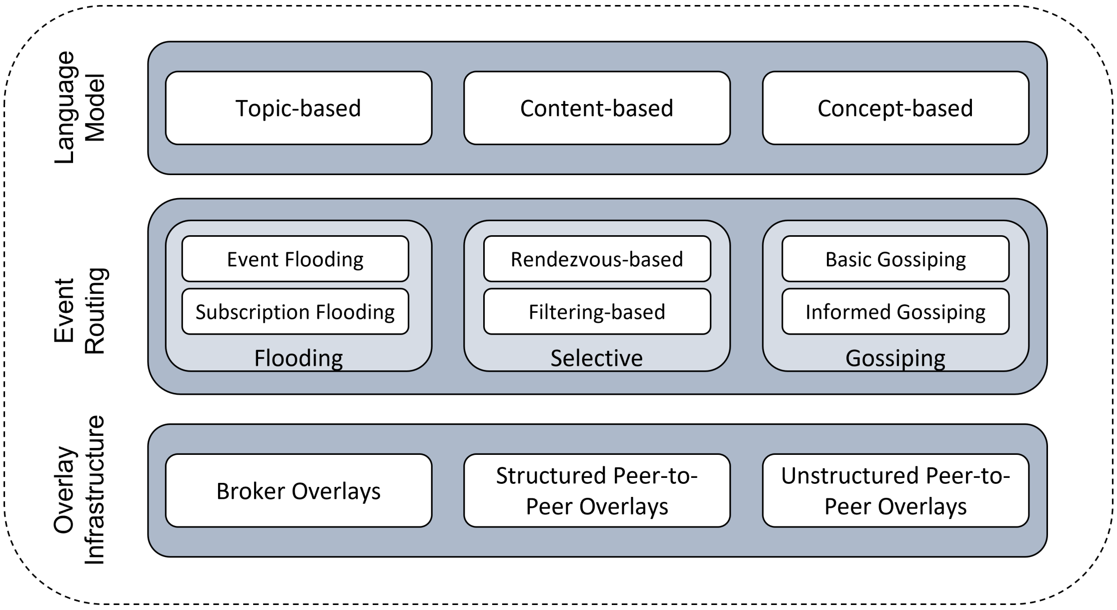


FIGURE 2.4: Distributed Publish/Subscribe Layered Architectural Model [3]

2.3.1 Language Model

One of the central features of a publish/subscribe system is its expressiveness. This aspect essentially constrains or promotes the degree of control subscribers have over the selectivity of the event matching process. The language model of a publish/subscribe system dictates the structure of events and subscriptions and introduces the operators used for matching. The following language models can be identified:

2.3.1.1 Topic-based model

In the Topic-based model, publishers submit events after associating them with a specific topic. A topic is a simple character string symbolising a logical channel that connects publishers to their respective subscribers. A subscriber declares its interest in a topic and receives events related to that topic. A topic can either be included as part of an event or a subscription or it can be addressable in the form of a channel created inside the event matching engine. This model is the earliest and simplest form of publish/subscribe communication. It is known to be highly scalable due to its resemblance to IP multicast where data traffic is addressed to a group of recipients. This aspect simplifies event routing which often takes the form of epidemic broadcast or multicast trees connecting nodes that deliver events under the same topic to a group of subscribers [92]. Certainly, the topic-based model is a natural fit when the event space is inherently divided into groups of events that can be mapped to distinct communication channels. Examples of systems that employ this model include Scribe [93], Bayeux [34], Tera [94], Poldercast [19], and Lipsen [95].

The limitation of this model lies in its low expressiveness. Its simplistic approach falls short in contexts where subscribers are only interested in a subset of all events submitted under a specific topic. Some solutions tackle this limitation through the use of topic hierarchies that store nested relationships between topics [96, 97]. A parent topic in the hierarchy groups together events submitted under related sub-topics. As subscribers go down topic hierarchies, they are in a sense employing more granular filters over what events they want to receive. Such hierarchies need to be manually constructed and made available as rule-bases that dictate the flow of events.

2.3.1.2 Content-based model

In the content-based model, subscribers have the ability to specify filters over the contents of events. Events usually follow a fixed structure or schema consisting of a set of tuples. Each tuple comprises an attribute-value pair and each attribute has a name and a type. The type can be any of the basic data types commonly found in programming languages, such as integers, strings, or floats. Subscriptions are often comprised of a conjunctive set of predicates that specify constraints over the values of attributes present in events. Constraints commonly include basic comparison operators (i.e. $=, <, >, \leq, \geq$). Generally, an event whose attributes hold values that match all the constraints imposed by a subscription is considered to be a match and disseminated towards the issuing subscriber. Examples of existing content-based publish/subscribe solutions include Padres [69], Siena [33], Hermes [17], and Meghdoot [35].

The high expressiveness of this model comes at the expense of higher resource consumption when evaluating incoming events against registered subscriptions. The complexity of event matching algorithms is influenced by several factors that include the following: (1) The number of possible attributes an event may carry, (2) the range of values included in subscription predicates, (3) whether attribute values are discrete or continuous, and (4) whether attributes have well defined domains of values [31]. Solutions often resort to predicate indexing schemes [33] and tree-based algorithms [98] to speed up and scale the matching process. Kale et al. [99] provides an analysis of various content-based event matching algorithms.

2.3.1.3 Concept-based model

The models discussed thus far perform exact string comparisons between the terms mentioned in events and subscriptions. For the matching scheme to work, publishers and subscribers have to establish an explicit agreement on the structure and semantics of exchanged events. The concept-based model extends the expressiveness of content and topic-based systems by incorporating conceptual hierarchies that retain semantic and syntactic relationships between addressable concepts. The use of such hierarchies loosens the agreement between publishers and subscribers on naming conventions and allows them to use related terms to describe the same semantic concepts. The main functionality of this semantic matching model can be illustrated in the following example. If a user is interested in receiving periodic updates on the 'energy consumption' of various equipment in a manufacturing environment, the publish/subscribe system should not only match events that contain the same phrase, but also events that contain the terms 'energy usage' or 'power consumption'. Popular solutions that propose a semantic matching model based on conceptual hierarchies include works by Petrovic et al. [56, 72], and Zeng et al. [100].

The limitation of this model lies in the associated cost coming from establishing semantic agreements. More precisely, building a concrete and encompassing conceptual model may require a lot of time and manual work from domain experts. Additionally, conceptual models, such as ontologies and knowledge graphs, provide a syntactic structure that models the semantics of a particular domain. Tying the event matching process to a specific conceptual model limits its re-usability with different models of meaning [55].

2.3.2 Overlay Infrastructure

The initial vision of the internet was to create a medium of interaction between individuals around the world by creating a global communications network connecting smaller, geographically dispersed, and heterogeneous computer networks. The internet rapidly transformed into a widespread information infrastructure that has to guarantee high levels of performance, availability, scalability, and security. The invention of the world-wide-web in 1989 [101], followed by the emergence of distributed internet applications such as e-commerce, social networks, real-time video conferencing, and most recently, the Internet of Things, has lead to a massive increase in the amount of daily generated data traffic. The ever-growing data ecosystem surrounding modern internet applications imposed stringent requirements on an underlying communications infrastructure that inherently lacked the capacity to meet such demands.

Redesigning and updating legacy infrastructure to cope with today's needs presented a costly and failure prone approach. Instead, research efforts resorted to the use of virtual networks, i.e. overlay networks [102], by following the same concept as creating a virtual machine as an abstraction layer over computer hardware. A virtual machine allows the hardware elements of a single computer, i.e. processors, memory, and storage, to be divided into multiple virtual computers. An overlay network on the other hand abstracts network infrastructure by allowing computers to implement networking functionalities such as content routing and transport at the application level. Nodes in an overlay network are connected via virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. For instance, distributed systems such as peer-to-peer networks and client-server applications are overlay networks because they run on top of the internet.

There are plenty of examples of overlay networks that target various requirements imposed by modern distributed applications. Content Delivery Networks (CDNs) [103, 104] are comprised of geographically dispersed servers and data centers that work together to provide fast delivery of cached internet content. When a user requests content from a website, such as Netflix, Facebook, or Amazon, if the content is cached on a CDN, the request is redirected to the nearest server hosting the cached content. Another example is Resilient Overlay Networks (RONs) [105], this type of overlay allows for the detection of path outages by monitoring the underlying physical links that make up a single virtual link and choosing the optimal paths to deliver application-specific content. The ability to route content at the application level allows such networks to recover from failures in a few seconds as opposed to underlying wide-area routing protocols that take several minutes to recover. In the following, we present three popular types of overlay networks that are commonly used for information dissemination and group communication by overlaying a publish/subscribe service atop their routing substrates (Table 2.1).

2.3.2.1 Federated Broker Overlays

Broker networks are a popular solution for medium to large-scale content-based publish/-subscribe systems with prominent research solutions that include TIB/RV [97], SIENA [33], PADRES [69], REDS [106], Gryphon [107], and JEDI [108]. Such overlays follow the client-server communication model and are usually comprised of a group of servers, commonly referred to as brokers. Each broker implements event matching and routing functionalities. Clients can join through any broker and take on the role of publishers that submit events or subscribers that express their interest in receiving events. Event routing decisions are not based on destination IP-addresses but on the content of events and the locations of subscribers that have expressed an interest in that content.

Event routing solutions targeting broker overlays can be classified into three main categories: Flooding algorithms, Filtering-based algorithms, and Multicast-based algorithms. Flooding algorithms can be further classified into subscription flooding and event flooding approaches. Such approaches are resource intensive especially in cases where high numbers of subscriptions and events are disseminated into the entire system. Filtering-based algorithms are generally based on building selective paths based on advertisement flooding and a reverse path selective propagation of subscriptions towards publisher-end brokers. Multicast-based approaches are generally based on grouping subscriptions based on similarities and intersections into cluster groups. Each publication is matched once to determine the matching multicast group(s) to which the message should be dispatched.

Flooding Approaches are known to be the earliest and most trivial solution to event routing with solutions such as Medym [109] and Elvin [110]. In the case of event flooding, events are broadcasted into the entire system, whereas subscriptions are only stored at subscriber-end brokers. Each broker evaluates received events against local subscriptions and dispatches matches to interested subscribers. This approach does not scale well due to its high messaging overhead, many brokers constantly receive events that have no local interest, which would eventually be faced with bandwidth limitations. In the case of subscription flooding, every broker would be aware of all the subscriptions that exist in the system. Once a broker identifies a matching event, it is dispatched to the corresponding subscriber-end broker via a one hop routing strategy. The obvious drawback of this approach is that the matching algorithms must scale to very high numbers of subscriptions. Furthermore, in cases where subscribers are highly dynamic, subscription and unsubscription messages would be constantly flooded throughout the system, which leads to noticeable spikes in messaging overhead.

Filtering-based routing is the most popular approach for event routing in broker overlays. Solutions include Siena [15] and Padres [69]. This approach is based on a deterministic propagation of subscriptions along reverse paths towards publisher-end brokers. More precisely, upon arrival of a publisher, it submits an advertisement message containing schematic information about events it is going to publish. Advertisements are then broadcasted and maintained in subscription routing tables at all brokers. As brokers receive subscription messages, they are first evaluated against locally stored advertisements, and if matched, they are recursively back-propagated towards the neighboring brokers that sent the matched advertisements. Following this selective propagation of subscriptions, every publisher-end broker would ultimately hold subscriptions that are only interested in its locally connected publishers. The only drawback of this approach is that to avoid the overhead of sending matching state along with matched events (e.g. the IDs of matched subscriptions), approaches implement highly optimized matching algorithms to redundantly match subscriptions along the paths towards interested subscribers.

Multicast-based event routing, also referred to as the channelization problem [111–113], divide the content space into multiple regions in a multi-dimensional space (the number of dimensions is equal to the number of event attributes). Typically, the ranges of each attribute would be bounded. Subscriptions are then assigned to different partitions. As a result, matching and transmission of a publication message happens at most once, thus incurring minimal delivery delay. However, compared to the filter-based approach, subscribers in a multicast group may receive unwanted publications because subscribers with even slightly different interests may still be assigned to the same group.

2.3.2.2 Structured Peer-to-peer Overlays

Structured peer-to-peer networks were originally motivated by systems such as Freenet [114], Gnutella [115], and BitTorrent [116]. Such systems took advantage of resources distributed across the Internet to provide decentralized, collaborative, and large-scale file-sharing systems. Such networks are built on the basis of Distributed Hash Tables (DHTs). Just like hash tables (hash maps), DHTs are used to store and retrieve key-value pairs. All keys in a DHT follow a consistent format and every node in the network is responsible for a partition of the key-space with well defined boundaries. The resulting overlay is a logical organization of nodes that often takes the shape of specific toplogies such as rings, hyper-cubes, or trees. The formation of such topologies mimics the structure of underlying virtual key-spaces. DHTs are characterized by being decentralized, highly scalable, and self-organizing, coping naturally with churn (i.e. node joins and departures), and are resilient to node failures.

Research solutions such as Chord [117], Pastry [6], CAN [7], and Kademlia [5], all offer Key-based Routing primitives (KBRs) that route a message m with key x to the node responsible for key x in a relatively small number of routing hops. The routing scheme works by iteratively finding closer nodes to x until the target node is identified. This requires that each node maintains enough information about nodes responsible for key-space partitions thus allowing for deterministic routing. Routing primitives work by hashing the identifier of a data item and mapping it to a location in the key-space. By choosing a large enough size of the key-space along with an appropriate hash function, solutions can guarantee the uniqueness of each generated key. Hence, KBR primitives guarantee an exact-match every time a participant attempts to store or retrieve a data item.

	Federated Broker Overlays	Peer-to-Peer Overlays
Examples	Siena [15] and Padres [16]	Scribe [18], Hermes [17], and Tera [94]
Governance	Semi-centralized	Fully decentralized
Interaction Model	Client-Server	Peer to Peer
Topology	Static	Dynamic and self-organizing
Reliability	High	Requires churn handling mechanisms
Scalability	Dozens of nodes	Thousands of nodes
Routing	Flooding, Filtering-based, and Multicast-based	Rendezvous-based and Gossip-based

Table 2.1: Federated Broker Overlays vs Peer-to-Peer Overlays

The underlying key-based routing primitive is in itself useful for other types of applications aside from file sharing. A KBR primitive can be very conveniently used to dynamically construct group spanning trees. If a group of nodes route to the same key, the union of their paths form a spanning tree that can be persisted and maintained. This method was used by Bayeux [34] and Scribe [18] to construct peer-to-peer topic-based publish/subscribe systems. The node responsible for the target key (i.e. Topic) forms the root of the spanning tree while subscriber nodes form the leaves to the tree. Event routing solutions in structured peer-to-peer networks rely on the exact-match property offered by the underlying lookup function to employ rendezvous-based routing approaches. Baldoni et al. [3] describe rendezvous-based routing as a two phase process: a publisher sends their events to a set of nodes, named rendezvous nodes, which match them against the subscriptions they host. Content-based event routing often works by mapping attribute-value pairs present in exchanged events and subscription predicates to the same location in the network. Examples of content-based systems built on top of a structured peer-to-peer overlay include Hermes [17], Meghdoot [35], PastryStrings [118], and Prefix Hash Tree [119].

2.3.2.3 Unstructured Peer-to-Peer Overlays

Unstructured peer-to-peer overlays organize nodes into a random structure that often takes the form of a mesh network. Each node in the network is connected to a subset of all the other nodes, irrespective of their ID or location. Such overlays leverage gossip-based protocols to design membership management protocols that facilitate building random graphs with a few basic properties. Random overlay construction algorithms strive to maintain (1) a low overlay diameter, (2) bounded node degrees where each node should be aware of a maximum preset number of neighbours, and (3) the overlay should remain highly connected in the case of node departures or failures. The basic idea behind gossiping protocols is based on a cyclic (periodic) or reactive random pairwise exchange of information between nodes. Each node maintains a *partial view* of the network which consists of the contact information (i.e. IP addresses and port numbers) of a subset of all the other nodes. The partial view of a node establishes neighbouring relationships with other nodes. It is constantly updated via a basic swapping procedure where nodes exchange part of their partial view with other nodes in the network. For scalability, partial views should be considerably smaller than the size of the full network, for instance, several solutions resort to having the partial view size grow logarithmically in proportion to the number of nodes in the network [120]. The dynamic nature of such protocols, with partial views constantly being updated, ensures high resilience to churn (i.e. nodes joining and leaving) and node failures.

Gossip-based routing protocols work in an epidemic manner, such that, if a node wants to send a message m, it broadcasts m to some of its neighbours. Upon reception of m, neighbouring nodes repeat this process until a preset *path length* is reached [92]. Gossip-based membership management protocols such as SCAMP [121], T-man [122], Cyclon [123], and HyparView [124] construct dynamic unstructured overlay networks without imposing constraints on the overlay structure. The randomized creation of such overlays results in a highly flexible organization of nodes that can be tuned to accommodate for various applications involving information dissemination and sharing between geographically dispersed and autonomous entities.

Conversely to structured peer-to-peer overlays, unstructured overlays are not ideal for ID-centric search queries and selective routing strategies, as there is no correlation between the identifiers of nodes that establish neighboring relations. However, their random nature has been exploited to build large-scale decentralized publish/subscribe systems. Popular approaches such as Tera [94], SpiderCast [125], Vitis [68], Poldercast [19], Stan [126], and Rappel [127], propose membership management protocols to support the dissemination of events in topic-based publish/subscribe systems. Such protocols work by constantly evolving and fine-tuning the topology of the overlay to create efficient data dissemination structures such as multicast trees or connected clusters of nodes with shared interests. The dynamic construction of such groupings of nodes is often followed by a selective flooding routing strategy where events are disseminated to all members of a single group.

2.4 Internet of Multimedia Things

Throughout the past two decades, many research efforts have been dedicated to a promising paradigm, widely referred to as the Internet of Things (IoT). The origins of the term come from the work of the Auto-ID Labs at the Massachusetts Institute of Technology (MIT) on networked radio-frequency identification (RFID) infrastructures [128]. There have been plenty of attempts to define what the term ”Internet of Things” actually means, we resort to a semantic elaboration that defines it as ‘a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols” [129].

The fundamental objective of the IoT is to collect and analyse data from the physical world in order to derive insights that feed into daily decision making processes. The IoT is comprised of physical objects that are equipped with sensors and with the ability to interact with the physical environment and with each other. In traditional IoT deployments, physical things are monitored via sensing devices with wide ranging sensing capabilities that vary across a wide spectrum of information to be stemmed from the physical world. Most sensing devices measure quantifiable physical properties, such as temperature, speed, and pressure. Such measurements often fall within well-defined numerical ranges and require minimal pre-processing steps before being presented in a human-readable format. On the contrary, qualitative physical properties, such as traffic collisions, formation of crowds, and PPE compliance, are descriptive in nature and often require significant knowledge extraction steps upon sensing and before being transformed into a human-readable format.

The Internet of Multimedia Things (IoMT) concerns observing the aforementioned qualitative physical properties from the environment and their acquisition and integration into multimedia-based services and applications. Alvi et al. [130] define the IoMT as an emerging paradigm in which smart heterogeneous multimedia things can interact and cooperate with one another and with other things connected to the Internet to facilitate multimedia-based services and applications. In what follows, we elaborate on the inherent characteristics of multimedia data traffic and the associated architectural and processing requirements that led the research community to draw a distinction between this paradigm and the IoT.

2.4.1 Multimedia Data vs. Scalar Data

Multimedia sensor networks in the IoMT collect multimedia data such as digital images, video, and audio, along with scalar data. Scalar data is characterised by having a single value that can be of any data type commonly found in programming languages such as strings, integers, or floats. On the contrary, multimedia data is multidimensional, often represented using information dense data structures (e.g. matrices, cubes, or graphs)

	Scalar Data	Multimedia Data
Examples	Temperature, pressure, or humidity reading	Raw Images, Video, or Audio
Size	Single Value (e.g., integer, float, string)	Bulky (e.g., cube, matrix, graph)
Format	Structured / Semi-structured (e.g., RDF, CSV, JSON)	Unstructured Native Formats (e.g., JPEG, WAV, AVI)
Storage	Relational / Graph Databases and Data Warehouses	Data Lakes
Querying	SQL, GraphQL, MySQL	Requires mining features and patterns

TABLE 2.2: Multimedia Data vs. Scalar Data

that hold a multitude of attributes and features. Unlike scalar data, multimedia data is bulky in nature and rich in content (Table 2.2).

Scalar data can be easily fit into a predefined schema or structure (e.g. RDF, CSV, JSON, etc.). Querying scalar data is widely supported with a plethora of query languages that target different forms of data stores ranging from relational databases to graph databases and many more (e.g. SQL, MySQL, GraphQL, etc.). Whereas, multimedia data is usually stored in native unstructured formats. Querying multimedia data requires mining features and patterns that can be transformed into addressable semantic concepts. Multimedia data mining is a multidisciplinary field that concerns the extraction of interesting patterns from multimedia content. It integrates image processing and understanding, computer vision, natural language processing, data mining, and pattern recognition.

2.4.2 Multimedia Event Processing

Multimedia event processing concerns the representation, identification, detection, and response to events that present in the form of multimedia content [36, 131, 132]. For example, the detection of traffic accidents by processing live video streams from cameras monitoring highways is one of many instances of IoMT-based multimedia event processing applications. In this thesis, we focus on multimedia event processing in the context of publish/subscribe systems. Connecting sensing devices to the internet is at the core of many IoMT applications and plays an immediate role in facilitating the acquisition and on-the-fly processing of data for real-time applications such as Autonomous Driving, Traffic Management, and Smart Agriculture. The scalability and interoperability offered by the publish/subscribe communication pattern has proved beneficial for connecting heterogeneous sensing devices and actuators to each other and to the internet.

There are many publish/subscribe protocols being used in IoT applications today, perhaps most notably MQTT [90] and DDS [133] but there are numerous others.

2.4.2.1 Multi-Agent Systems

Alvi et al. [130] emphasise the need for Multi-agent systems to facilitate the objectives of distributed IoMT applications and integrate heterogeneous services. A Multi-agent system can be defined as a loosely coupled network of autonomous self-organizing software agents that cooperate to solve problems that are hard to tackle in monolithic systems [134]. We argue that the objectives of Event-Driven Architectures (discussed in section 2.2.1) align perfectly with the proposed multi-agent processing model. Consequently, a distributed multimedia event processing system should enable the following functionalities:

- **Service Composition**: The integration of various heterogeneous services that can be built on the basis of the same or different multimedia content. This functionality allows for the reusability of data acquired from heterogeneous multimedia networks and sensor infrastructures. It also entails the customization of multimedia-based services according to user preferences, available compute and storage resources, and various Quality of Service (QoS) requirements [131].
- **Extraction of Semantic Concepts**: The extraction and materialization of necessary information and patterns from large amounts of multimedia data. The unstructured nature of multimedia content demands the need for deriving useful semantic information via the cooperation of multiple agents implementing Artificial Intelligence (AI) models with a wide spectrum of knowledge extraction capabilities [36, 135].
- **Content Sharing and Delivery**: Traditional point-to-point communication models would not be able to fulfil user Quality of Service (QoS) requirements and strict bandwidth constraints due to the multiplicity of multimedia data sources and the high bandwidth requirements of multimedia content. A many-to-many communication model that decouples the production and consumption of multimedia content and allows for the seamless cooperation of participating agents is prime to the full realization of the IoMT.

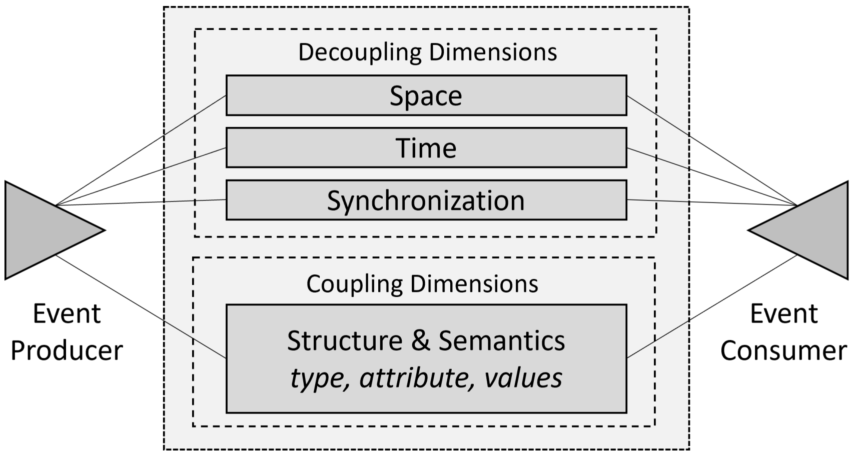


FIGURE 2.5: Dimensions of De/coupling in publish/subscribe systems [4]

2.4.2.2 Semantic Coupling in Publish/Subscribe Systems

EDAs rely on event-driven interactions to maintain long-running asynchronous process capabilities [78]. Unlike the request/response communication pattern, participants in an event-driven interaction are decoupled in space, time, and synchronization [23] (Fig. 2.5).

- **Space decoupling** means that event producers and consumers do not need to know of each other.
- **Time decoupling** means that event producers and consumers do not need to be active at the same time.
- **Synchronization decoupling** means that event producers and consumers are not blocked while producing or consuming events.

Yet, participants are tigtly coupled by the structure and semantics of exchanged events. Semantic coupling poses a challenge to scalability in IoT's highly semantically heterogeneous and dynamic environment [136]. Such challenges are further exacerbated by the wide semantic spectrum of heterogeneous multimedia content from the IoMT. The decoupling aspects have been useful for enabling service composition in Event-Driven Architectures. However, to fully enable content sharing and delivery in the IoMT, the publish/subscribe communication model has to offer native support for multimedia event types.

Multimedia Event Types: Participants in a publish/subscribe interaction are tightly coupled by the semantics of exchanged events [55, 57]. For the matching scheme to work, event producers and consumers have to reach an explicit agreement on the terms used to describe exchanged content. Establishing explicit agreements on structure and semantics has previously presented no issues when applied in closed environments with well defined semantics such as Stock trading platforms. The attributes that subscribers may be interested in, such as, a Stock's name, symbol, price, or volume, are well defined and easy to agree upon throughout the system [16]. However, as the nature of events departs from structured textual and numerical values to unstructured and heterogeneous multimedia content, e.g., images, audio, and video, from the IoMT, such an agreement becomes much harder to achieve and much costlier. For instance, querying video content often requires a pre-processing step that materializes all the necessary information in the form of object labels, relationships, and attributes [37, 131]. The terms (labels) used to describe exchanged content come from image datasets used to train Computer Vision models. Popular datasets, such as, ImageNet [26], OpenImages [137], and Visual Genome [138], cover tens of thousands of labels. The scale of the content space gives rise to a semantic boundary between IoMT publishers and subscribers.

2.4.2.3 Semantic Coupling in DHTs

The dynamic and self-organizing nature of peer-to-peer infrastructures makes them a great fit for providing compute, storage, and communication, functionalities for large-scale dynamic environments such as the IoMT. Quite conveniently, parallels can be drawn between information exchange and sharing in the IoT and the traditional peer-to-peer interaction model. Korzun et al. [139] argues that every smart device, sensor, actuator, or analytics process, could be represented by a corresponding agent (or peer) in an overlay network. Services then emerge as a result of cooperative work of multiple agents in the smart environment. Each agent contributes to the service by sharing its portion of knowledge from the surrounding physical environment.

Structured peer-to-peer overlays have often been favored over unstructured overlays to implement content-based publish/subscribe systems [66]. Solutions employ consistent hashing and rendezvous routing by mapping the attribute space of the latter to unique identifiers in the identifier space of the former. Rendezvous routing allows two messages, e.g., a query and a data item, both addressing the same unique identifier, to meet at the same node in the network, called rendezvous node [3]. However, a slight inconsistency in the naming of attributes as specified by publishers and subscribers can result in mapping conceptually matching content to distant locations in the network. The problem can be attributed to the underlying logical key-space with no physical properties. A hash function is often used to map data items to identifiers that are merely logical pointers with no meaningful relationship to the data they represent. As an example (Fig. **??**), an event that describes a car collision as follows {'vehicle', 'crash'}, might fail to coincide

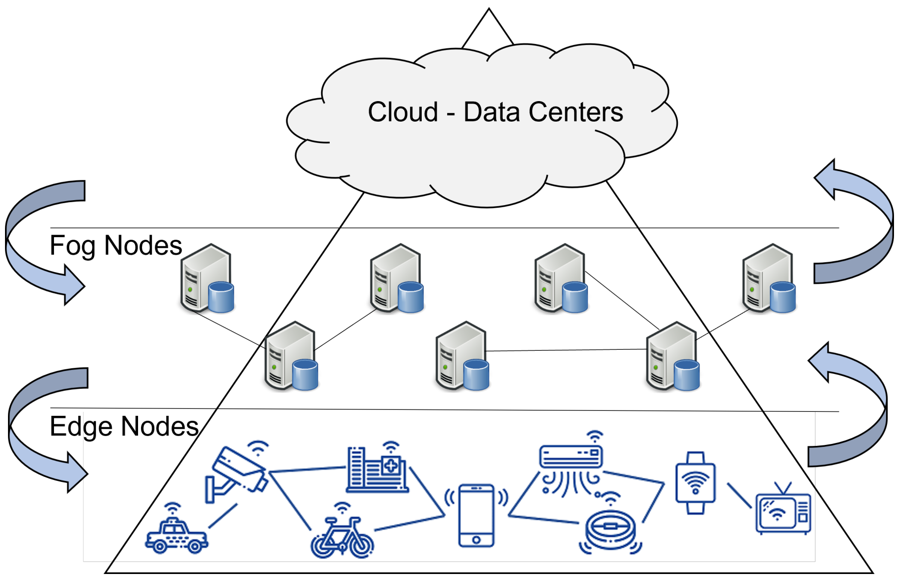


Figure 2.6: Fog and Edge Computing High-level Architecture

with a subscription that specifies a conceptually very similar expression as follows {'car', 'collision'}. Hashing the terms 'car', vehicle', 'crash', and 'collision', will result in four unique identifiers (hashes) mapped to locations in the space following a uniform random distribution. Hence, the probability of the event rendezvousing with the subscription at the same node is exceptionally low.

2.4.3 Fog and Edge Computing

The enormous growth of multimedia sensing devices in the IoMT has emphasised the need for bringing additional functionality that deals with the bulky nature of multimedia content. The high compute, memory, and storage resources required to process such content necessitates transporting all data from physical assets to the cloud for storage and advanced analysis. Once in the cloud, the data is used for cognitive prognostics. Due to the high bandwidth requirements associated with transporting multimedia content to the cloud, recent research efforts have been focusing on bringing the processing of such content closer to the network's edge, in close proximity to the physical location where the data was created.

Edge computing [140, 141] comprise network and system architectures that attempt to collect, analyze, and process data from physical assets more efficiently than traditional cloud architectures. These architectures share similar objectives which include reducing the amount of data sent to the cloud, decreasing network and Internet latency, and improving system response time in remote mission-critical applications [142]. Fog computing, a term created by Cisco, also refers to extending computing to the edge of the

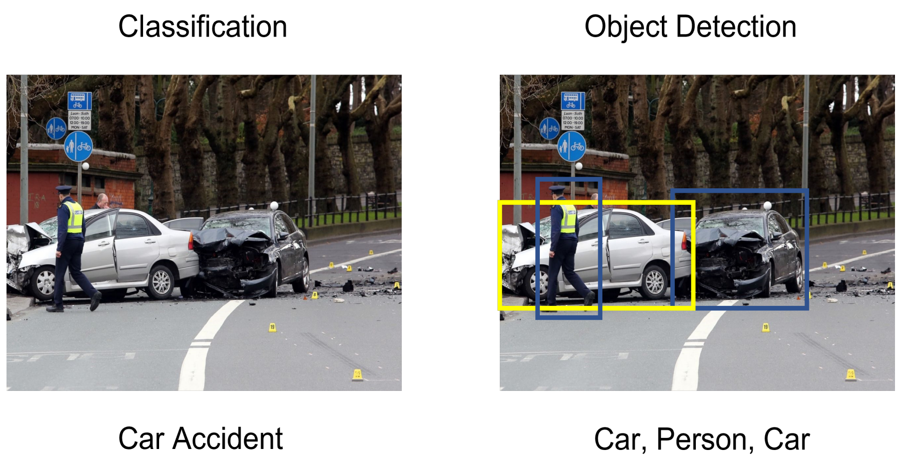


Figure 2.7: Image Classification vs. Object Detection

network [143]. This is realized via compute and storage nodes (i.e. fog nodes) that are spread out along the continuum between the things and the cloud (See Fig. 2.6).

The sensing and perception capabilities offered by IoT and IoMT smart environments create the data and knowledge on which daily decisions are made. These are used by the reasoning and decision-making technologies to deliver; edge, fog, and cloud-based decision making, planning, search and optimization in systems, and the multilayered decision-making necessary for AI, data and robotics systems operating in complex environments [144]. We adopt the fog and edge computing paradigms as enablers for peer-to-peer Ambient Intelligence (Aml) services in IoMT smart environments. Korzun et al. [39] defines Aml services in IoT environments as intelligent software agents that help orchestrate all aspects of an environment. For instance, resource constrained edge devices may implement IoMT publisher or subscriber functionalities, whereas, fog nodes may implement more resource intensive tasks such as multimedia data mining, knowledge extraction, and multimedia event processing. Fitting fog and edge nodes with publish/subscribe many-to-many communication capabilities is a crucial requirement for supporting the dissemination of multimedia content generated by IoMT physical things to a diaspora of real-time applications at internet scale.

2.4.4 Computer Vision

Computer Vision is the field of Artificial Intelligence (AI) that allows computers to replicate parts of the complexities of human vision. It enables computers to interpret and identify objects in images and videos. Until recently, computer vision only worked in limited capacity. Due to advances and innovations in deep learning and neural networks,

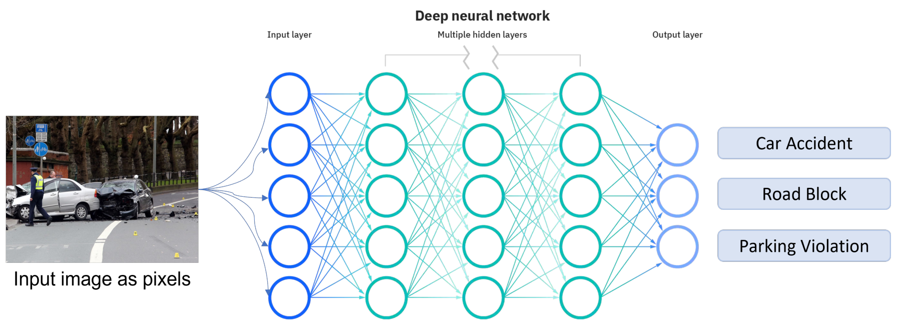


Figure 2.8: Illustrative example of Image Classification using Deep Learning

the field has grown in recent years and has been able achieve outstanding accuracy in some tasks related to detecting and labeling objects.

2.4.4.1 Image Understanding

Image understanding is the process upon which a computer program analyses an image and interprets whats happening or visible inside of it. Several Computer Vision methods can extract high-level semantic information that describe human-recognisable objects or regions within an input image. Ultimately, the extracted descriptions are useful for taking subsequent decisions or actions. Two of the most common and useful knowledge extraction tasks in Computer Vision today are Image Classification and Object Detection (see Fig. 2.7).

Image Classification is where an algorithm analyses an image and identifies the class it belongs to. The algorithm often returns the probability of the input image belonging to a class. A class is essentially a label, representing a semantic concept, for instance, ‘car’, ‘person’, ‘sky’ and so on. For example, you input an image of a car, Image Classification is the process of the computer analysing the image and telling you it’s a car along with a score that signifies the confidence or the probability of this classification.

Object Detection entails the identification and localization of objects that exist in an input image. An algorithm analyses an image and places bounding boxes on all the human-recognisable objects that appear. An object is a boxable semantic concept such as a ’car’, ’person’, ’dog’, or ’table’. Essentially, this process breaks up an image into blocks similar to a grid and tries to assign each of the blocks to a class. Object Detection is commonly used in applications such as face detection, activity recognition, or vehicle counting.

2.4.4.2 Deep Learning

Common Computer Vision tasks have been traditionally tackled using hand crafted features like SIFT [145]. Features had to be identified by a domain expert to reduce the complexity of the data and make patterns more visible to learning algorithms. The quality of features limited the performance of such models and prevented them from scaling to complex representations. Efforts to scale these algorithms on larger datasets have reached a breakthrough in 2012 during the ILSVRC competition [146], which involved, the task of classifying an image into one of thousand categories. For the first time, a Convolutional Neural Network (CNN) commonly known as AlexNet [26] brought down the error rate on that task by half, beating traditional hand-crafted approaches. AlexNet was then followed by a plethora of optimizations and architecture modifications for better performance with solutions such as VGGNet [43], Inception [42], and ResNet [41]. The biggest advantage with Deep Learning algorithms is that they try to learn high-level features from data in an incremental manner. This eliminates the need of domain expertise and hard core feature extraction.

As shown in Fig. 2.8, a CNN is composed of multiple layers, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features. It uses a three-dimensional neural network to process the Red, Green, and Blue elements of the image at the same time. This considerably reduces the number of artificial neurons required to process an image. During the training process, a CNN receives many 'examples' of images that belong to a target class, after several iterations, a classifier is created and used to employ an inference task on unseen images. Inference is the second phase, in which the model is put into action on live data to produce actionable output such as classifying an image or placing bounding boxes over detected objects.

This brief overview is not intended to cover all the works proposed in the literature, neither to provide a thorough explanation of the inner workings of deep learning algorithms. It was introduced mainly for the sake of completeness, since the focus of our work is on the adaptation of such models to work within a distributed publish/subscribe system. A formal analysis and comparison of deep neural network models can be found in [147].

2.5 Summary

In summary, this chapter introduced and discussed the event processing paradigm and the different forms of dedicated event processing platforms that fall under its umbrella. I went on to discuss the publish/subscribe communication model in detail, shedding light on its various language models, distributed implementations spanning semi-centralized broker overlays and full decentralized peer-to-peer overlays, and the routing strategies identified in the literature. Furthermore, the Internet of Multimedia Things is introduced as a prominent source of large amounts of multimedia content that need to processed in real-time and fed into a plethora of distributed applications. I break down the inherent features of multimedia data and the gaps in current event processing infrastructures while highlighting the need for multimedia event processing platforms that support service composition, multimedia data mining and analytics, and content sharing and delivery. Finally, fog and edge computing infrastructures are discussed, in addition to methods and techniques for image understanding in modern computer vision.

Chapter 3

Problem and Motivation

3.1 Introduction

In this chapter, I delve into and motivate the various research problems tackled in this thesis. In section 3.2, the main objective of supporting multimedia content as native event types in distributed publish/subscribe systems is introduced and the associated challenges are identified. Section 3.2.1 motivates the case for decentralized and highly scalable peer-to-peer overlay networks via a scenario relating to traffic management based on processing video streams from the Internet of Multimedia Things. In section 3.2.2, the case for broker overlays is motivated via a scenario concerning the dissemination of multimedia events relating to clothing classification and clothing item retrieval. Following that, section 3.3 details how the problem manifests itself under a federated broker overlay infrastructure. In particular, it discusses the multimedia event matching and routing challenges that arise, it highlights the need for the efficient extraction of meaningful semantic concepts from unstructured multimedia content, for provisioning a decomposable workload and adaptive routing decisions, and for ensuring relay-free event propagation. In section 3.4, I detail how the problem manifests itself under a structured peer-to-peer overlay. The challenges arising from the wide semantic spectrum of human-level recognition are discussed. Section 3.4.1 focuses on semantic data integration and exchange in distributed hash tables, whereas section 3.4.2 focuses on the event matching and routing challenges arising in the form of a semantic boundary between event producers and consumers. Finally, section 3.5 summarises the chapter and highlights the main challenges tackled in this work.

3.2 Problem Overview

In this thesis, I focus on supporting multimedia content as native event types in distributed publish/subscribe systems. Generally, supporting multimedia event matching and dissemination entails integrating knowledge extractors into the event processing pipeline within publish/subscribe event matching engines. The on-the-fly and near real-time nature of this processing paradigm imposes event matching and routing challenges stemming from the unstructured nature of multimedia content. In particular, an important requirement when building a publish/subscribe system is that event consumers, i.e. subscribers, need to be notified as soon as relevant events occur with minimal latency. However, as the nature of events departs from structured textual or numerical content to unstructured multimedia content, expressions submitted by subscribers would remain stuck behind costly operators that have to be utilized to extract meaningful semantic concepts from incoming streams of unstructured multimedia content. Moreover, the wide semantic spectrum of human-recognizable concepts that can be stemmed from multimedia content creates a large semantic content space. Subsequently, event matching and routing challenges arise in the form of a semantic boundary between event producers and consumers. In particular, participants may use different terms to refer to the same or highly related semantic concepts. This aspect introduces a semantic matching problem and limits the applicability of conventional exact string matching and routing approaches.

The work tackles distributed implementations of publish/subscribe systems, which, as discussed in section 2.3.2, take the form of application-level overlays that fall under two main categories: Semi-centralized federated broker overlays and fully decentralized peer-to-peer overlays. The underlying overlay infrastructure dictates event matching and routing methodologies. Hence, in what follows, we discuss in detail how the aforementioned challenges are manifested based on the underlying overlay infrastructure.

3.2.1 Motivational Scenario - 1

The Internet of Multimedia Things is a prominent source of various types of multimedia content that feed into a plethora of real-time applications. The multiplicity and heterogeneity of multimedia data sources in the IoMT demands the participation of various agents that cooperate to process and deliver insights in real-time. Salient to any multi-agent system is a messaging layer that allows for the dissemination of information across multiple cooperating agents and the subsequent amalgamation of the outputs of distributed data sources and intelligent processing units. As shown in Fig. 3.1, in a

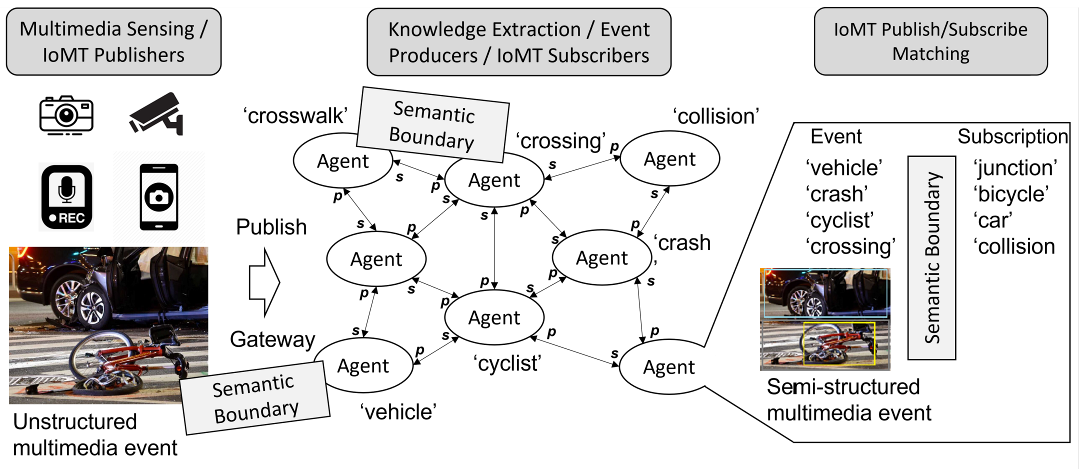


FIGURE 3.1: The arising semantic boundary between heterogeneous data producers and consumers in the IoMT

traffic management scenario, agents equipped with AI models may act as event producers that submit labelled multimedia content after applying knowledge extraction tasks such as classification, object detection or segmentation, or speech recognition. They mediate the interaction between IoMT publishers and a publish/subscribe messaging layer. On the other end, other agents or end-user devices act as subscribers that submit expressions describing their interests.

The interaction between the various cooperating entities is facilitated through a peer-to-peer network of self-organizing nodes (i.e. agents) where any node can join at any time and take on the role of an IoMT publisher, a knowledge extractor, a publish/subscribe event matching engine, an IoMT subscriber, or any simultaneous combination of roles. However, the cooperation and exchange of information between heterogeneous and autonomous agents is limited by the wide semantic spectrum of multimedia content. For instance, a traffic collision can also be described as a car crash or accident, the types of vehicles involved may fall into various categories and types such as compact cars, SUVs, sports cars, or trucks etc., even further, subscribers might be interested in information about whether the accident was on a highway, freeway, crossing, or crossroads, or whether pedestrians or cyclists have been involved. Establishing a strict and explicit agreement on the terms used to describe semantic concepts is challenged by the wide semantic spectrum of multimedia content and the dynamics of peer-to-peer networks.

3.2.2 Motivational Scenario - 2

We consider two distinct entities looking to analyse clothing attributes and colour patterns in images being uploaded by different users on social media platforms. A fashion design company wants to predict the future popularity of different fashion items. An

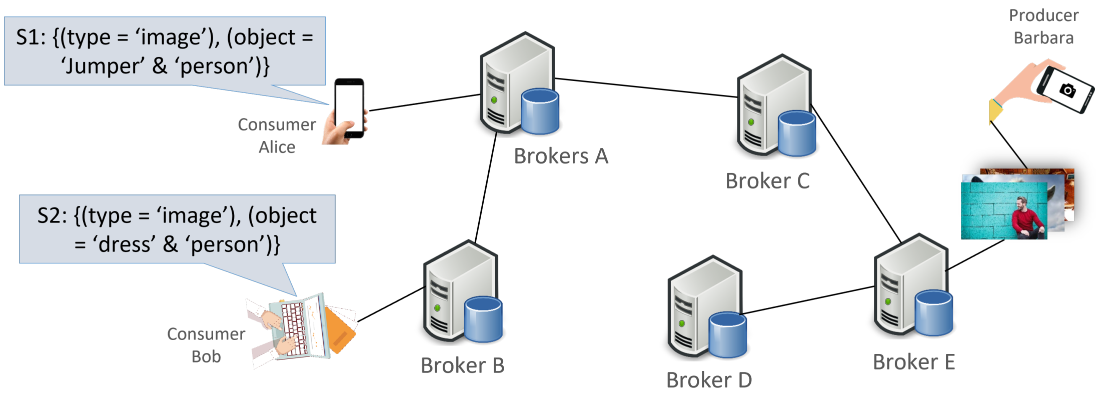

S1: {(type = 'image'), (object = 'Jumper' & 'person')}
Consumer Alice
Brokers A
Broker C
Producer Barbara
S2: {(type = 'image'), (object = 'dress' & 'person')}
Consumer Bob
Broker B
Broker D
Broker E

FIGURE 3.2: Selective dissemination of images showing clothing and fashion items via federated broker overlays

online shopping retailer wants to identify potential customers by analysing their clothing patterns. Extensive research efforts have been devoted to clothing classification and clothing item retrieval [148, 149]. Consider the following example queries requiring the User IDs and Image URIs out of a pool of users by specifying criteria concerning their age, gender, clothing items, and colours.

```
SELECT UserID, ImageURI,
FROM users
WHERE Age = [20,25] ∧ Gender = 'Female'
∧ Dress_Type = 'Formal_Dress' ∧
 Color = 'Black';

SELECT UserID, ImageURI,
FROM users
WHERE Gender = 'Female' ∧ Jacket = 'Denim'
∧ Color = 'Blue';
```

The variation in styles, textures, and colours results in a large number of concepts to be identified. The scale of the content space makes the cost of maintainability of knowledge extractors very high when new unseen concepts constantly appear. Currently, each of the different entities use custom solutions with specialized AI models. In many cases the queries are complex requiring multiple classifiers and feature extractors. As depicted in Fig. 3.2, we propose the use of publish/subscribe systems as a common substrate that allows retailers and fashion designers to act as subscribers to published social media content while preserving a decoupled and asynchronous interaction. The challenge to be tackled is the optimization of the execution of multiple classifiers across an overlay of brokers in a way that avoids bottlenecks, minimizes communication and processing costs, and speeds up event matching and delivery.

3.3 Multimedia Event Processing in Federated Broker Overlays

A federated broker overlay is comprised of a group of servers that are often managed by a single administrative entity. The closed and semi-centralized nature of such overlays allows for integrating and managing operators, i.e. knowledge extractors, that extract meaningful semantic concepts from multimedia content. Extracted concepts, i.e. labels, can then be evaluated against expressions submitted by subscribers as in conventional content-based event matching approaches. However, the integration of such operators may impose various challenges relating to the efficiency and effectiveness of event matching and delivery.

In what follows, I elaborate on research question Q1. and its associated technical requirements.

Q1. How to integrate fast and resource-aware knowledge extractors into the forwarding function of distributed publish/subscribe systems comprising federated broker overlays?

Technical system requirements:

- **R1.1. Efficient extraction of semantic concepts.** Timely and resource-aware materialization and filtering of raw images that scales well with the increase in the number of subscriptions.
- **R1.2. Decomposable workload and adaptive routing decisions.** This entails the provisioning of event matching and routing techniques that facilitate a decomposable processing workload as mutually independent operations that leverage the distributed nature of federated broker overlays.
- **R1.3. Relay-free event propagation.** This entails eliminating or minimizing the participation of relay nodes in the event matching and routing process.

3.3.1 Efficient Extraction of Semantic Concepts

Deep Neural Networks have been pushed to the for-front of computer vision applications due to their high effectiveness in uncovering high-level semantic concepts from raw images and video frames. Yet, it is generally acknowledged that for real-time applications with high data rates, applying DNNs is prohibitively expensive, requiring extensive resources such as GPU equipped devices or server clusters in cloud environments [46]. To overcome this limitation, there have been a rising interest in building efficient small neural networks in an attempt to trade of accuracy to much higher throughput and lesser memory footprint. Works such as SqueezeNet [150] and MobileNets [151] exploit the neural network size and the size of input images among other parameters to build small and fast models that achieve comparable accuracy to state-of-the-art models on tasks such as image classification and object detection. However, the utilization of such models to extract queryable textual labels in a timely manner is challenged by the large content-space of human-recognizable semantic concepts that can be stemmed from multimedia content. Subscribers may submit expressions as conjunctive predicates specifying an ever-growing number of semantic concepts such as "car", "cat", or "butterfly", which puts additional constraints on the maintainability of such models. This aspect demands provisioning an efficient methodology that allows the system to constantly update the semantic coverage of existing classifiers while maintaining fast and resource-aware knowledge extraction in order to cope with a growing number of subscriptions.

3.3.2 Decomposable Workload and Adaptive Routing Decisions

Clients acting as publishers of multimedia events or subscribers interested in receiving multimedia events may join the overlay through any broker. Routing decisions should be made in order to propagate event notifications from brokers connecting publishers towards brokers connecting subscribers. On the one hand, extracting semantic concepts at publisher-end brokers may result in bottlenecks and constrain the efficiency of the system in terms of event matching and delivery latency. On the other hand, differing the extraction of semantic concepts to subscriber-end brokers may result in congesting links by flooding bulky multimedia content through-out the overlay network. Hence, there is a need for leveraging intermediary nodes and for provisioning a selective event routing strategy to intelligently distribute the processing workload along the routing paths towards subscribers or across brokers in the overlay network. This should be manifested via a decomposable and adaptive workload processing strategy as mutually independent operations that leverage the distributed nature of broker overlays.

3.3.3 Relay-free Event Propagation

Relay nodes are brokers that participate in the event matching and routing process without having any locally connected subscribers that are interested in the multimedia content being processed. The participation of such nodes may add additional processing costs and decay the efficiency of the system. This challenge entails modifying the overlay topology in order to minimize event matching latency, reduce redundant processing, eliminate bottlenecks, reduce overall processing overhead, reduce messaging overhead,

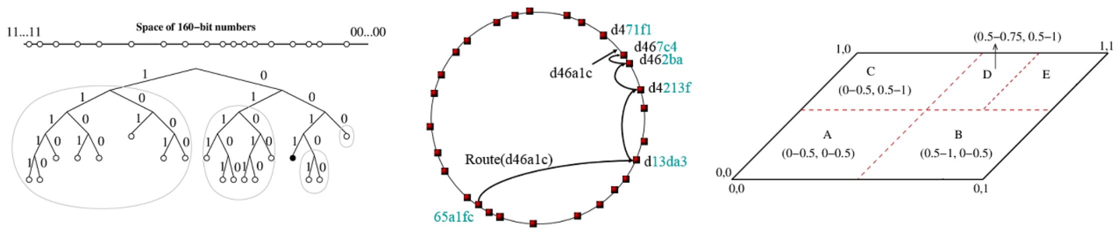


Figure 3.3: From left to right: Illustrative representation of the key-spaces underlying Kademlia [5], Pastry [6], and CAN [7]

and speed up event delivery. Modifying the overlay structure in response to skewed distributions of subscribers across brokers has previously been tackled in both topic-based systems [64, 65] and content-based systems [62, 63]. However, the existence of costly operators that extract semantic concepts from multimedia content impose additional constraints and limit the applicability of current overlay reorganization techniques.

3.4 Multimedia Event Processing in Structured Peer-to-Peer Overlays

Structured peer-to-peer overlays are known to be decentralized, highly scalable, and self-organizing. The absence of a single entity with explicit administrative control over the network limits the ability to integrate and manage operators that extract meaningful semantic concepts from the multimedia content being disseminated. Consequently, participating nodes such as event publishers, subscribers, or intermediary nodes, would have to manage their own operators. This aspect makes it highly challenging for publishers and subscribers to reach an explicit agreement on the semantics of exchanged events and subscriptions, which in turn results in event matching and routing challenges as we elaborate below.

3.4.1 Semantic Data Integration and Exchange

Structured peer-to-peer overlays have often been favored over unstructured overlays to implement content-based publish/subscribe systems [66]. Such networks are built on the basis of DHTs that provide a lookup service to store and retrieve key:value pairs. Solutions employ consistent hashing and rendezvous-based routing by mapping the attribute space of events and subscriptions to unique identifiers in the identifier space of the underlying DHTs. However, a slight inconsistency in the naming of attributes as specified by publishers and subscribers may result in mapping conceptually matching content to two distant locations in the network. As an example, an event that describes a car collision as follows {’car collision’, ’junction’}, might fail to coincide with a subscription that specifies a conceptually very similar expression as follows {’vehicular accidents’, ’crossing’}. The problem can be attributed to the underlying logical key-space with no physical properties. A hash function is often used to map data items to identifiers that are merely logical pointers with no meaningful relationship to the data they represent.

In what follows, I elaborate on research question Q2. and its associated technical requirements.

Q2. How can loose semantic coupling be achieved within the exact-match lookup functionality offered by DHTs underlying structured peer-to-peer networks while ensuring the dynamic organization, allocation, and retrieval, of data based on their shared semantic and syntactic properties.

Technical system requirements:

- **R2.1. Effective organization and mapping of semantic data.** This entails the distributed organization of data in a way that reflects the basic meaning of data items and the semantic relationships among them.
- **R2.2. Efficient and effective semantically decoupled lookup operations.** This entails departing from ID-centric file sharing to loose semantic coupling in DHTs. The retrieval of data items should be on the basis of a semantic routing model that relaxes strict agreements on semantics and locates the target data items at a low cost.

3.4.1.1 Effective Organization and Mapping of Semantic Concepts

DHTs such as Chord [117], Pastry [6], CAN [7], and Kademlia [5], are built on the basis of virtual key-spaces that take the shape of rings, trees, and d-dimensional coordinate spaces (Fig. 3.3). The key-spaces are partitioned arbitrarily and each node is assigned a unique key as its identifier and maintains a partition of the space with well-defined boundaries. As a consequence, and with the aid of consistent hashing, data items are mapped to locations in the key-space following a uniform random distribution. This inherent property carries a major benefit in terms of load-balancing across nodes in the overlay. It also allows for employing exact match search queries, but stands in the way of matching two semantically related terms. The random distribution of data items leads to mapping semantically related terms to arbitrary locations in the network. Hence, the first challenge entails devising a strategy to organize and map data items across a structured peer-to-peer overlay in a way that reflects their semantic and syntactic relationships.

3.4.1.2 Efficient and Effective Semantically Decoupled Lookup Operations

Conventional DHTs leverage Key-based routing primitives to perform ID-centric lookup operations. Such primitives are aided with a closeness metric that allows nodes initiating lookup operations to iteratively find closer nodes to the key being mapped. Prior approaches have tackled the semantic matching problem through the use of mapping tables. Such tables store the correspondence between values and are shared among participating peers [52]. With such tables, when a participating peer attempts to retrieve a file called X, it first consults a shared or local mapping table to find the names of X in each of the target peers. It then uses the underlying KBR primitive to issue exact-match lookup operations for all the identified names of X. However, as such tables grow in size, they can hinder the performance of the system by forcing peers to initiate many lookup operations. Moreover, they may take a lot of manual work and expert knowledge to build, and their semantic coverage can be considered limited compared to the wide semantic spectrum of recognition at the scale of human ability. The challenge at hand entails replacing conventional KBRs with a semantically decoupled routing primitive, i.e. lookup protocol, that has the ability to map data items via a closeness metric that reflects shared semantic and syntactic properties.

3.4.2 Content-based Matching and Routing of Multimedia Event Types

Generally, under a peer-to-peer setting, the targeted expressiveness of content-based publish/subscribe systems defines the requirements placed on the underlying overlay structure [31]. The notion of content is typically implemented assuming a well-known schema for both subscriptions and events. The schema often takes the form of an ordered set of attribute-value pairs. Quite conveniently, DHTs are suitable for ID-centric queries, whereby, any data item having a given ID, (e.g. attribute name), can be mapped to the node with the closest ID. As stated by Voulgaris et al. [54], "Any indexable set of data can be appropriately mapped to the nodes of a DHT overlay and spread accordingly". Consequently, several approaches for content-based event matching rely on subscription indexing by decomposing predicates or strings into successive nodes in a binary tree [118, 152]. Events are first mapped to the root node via rendezvous-routing, then they follow down a predicate-based exact-match path towards the leaf nodes. It is clear that the matching scheme in DHT-based solutions is dictated by the event-schema and the underlying exact-match key-based routing primitives. On the contrary, in the case of multimedia content, the interaction between publishers and subscribers is mediated by classifiers and object detectors that place a finite number of labels on disseminated content for target data searchability. The resulting event schema

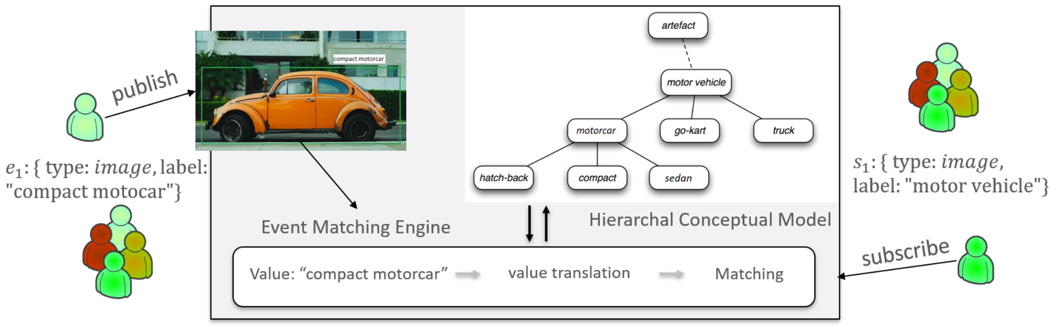

publish
e_1: { type: *image*, label: "compact motocar"}
compact motorcar
artefact
motor vehicle
motorcar
go-kart
truck
hatch-back
compact
sedan
s_1: { type: *image*, label: "motor vehicle"}
Event Matching Engine
Hierarchal Conceptual Model
Value: "compact motorcar"
value translation
Matching
subscribe

Figure 3.4: Boolean semantic event matching based on a hierarchical conceptual model of word meaning

comprises a collection of terms of unknown length and no pre-defined attribute names. In a way, each event resembles a collection of values with no explicit agreement between publishers and subscribers on the terms used to describe the semantic concepts that values might carry.

In the case of visual multimedia content, i.e. raw images and video frames, classifiers and object detectors acquire labels from the datasets used to train computer vision models. Such datasets can be classified into two broad categories, namely specialized small-scale datasets, and generic large-scale datasets. Specialized small-scale datasets (e.g. COCO [153] and PascalVOC [154]) cover 10s to 100s of semantic concepts. The bounded concept space of such datasets facilitates an event matching model based on term-level full agreement but limits participants to a relatively small number of semantic concepts. On the contrary, generic large-scale datasets (e.g. ImageNet [155] and OpenImages [137]) structure the labeled data following hierarchical semantic structures borrowing concepts from Cognitive Science. Motivated by the observation that for recognition at the scale of human ability, categories will necessarily overlap and display a hierarchical structure [156]. ImageNet, which is one of the largest available image datasets, organizes the different classes of images according to WordNet [157]. WordNet can be seen as a combination of dictionary and thesaurus. It is a network of words linked by lexical and semantic relations. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (i.e. synsets), each expressing a distinct concept. There are around 80,000 noun synsets in WordNet. ImageNet covers over 30,000 synsets with on average 500-1000 images per synset. OpenImages by GoogleAI is another large-scale image dataset that provides images with 19794 different classes, 600 boxable objects of which are organized into a semantic hierarchy. As depicted in Fig. 3.4, the use of large-scale datasets for training facilitates an event matching model based on concept-level shared agreement via ontologies.

In a peer-to-peer setting, an agreement on a conceptual model would be utilized in a manner similar to the use of mapping tables as discussed in section 3.4.1.2. A node issuing a subscription, would first consult a local or shared conceptual model and fetch all the synonyms and semantically related concepts to all the terms present in the subscription. The subscriber would then map all the acquired terms to various locations across the overlay. However, the same challenges associated with mapping tables would follow suit. As a consequence, and in the absence of an explicit agreement on event semantics, whether it being a term-level full agreement or an agreement on a conceptual model, event matching and routing challenges arise.

In what follows, I elaborate on research question Q3. and its associated technical requirements.

Q3. How to support a conjunctive content-based subscription language over multimedia events spanning a large semantic content space in structured peer-to-peer networks?

Technical system requirements:

- **R3.1 Low cost in defining rules.** Low barrier to entry for participants in terms of establishing semantic agreements on terms present in exchanged events and subscriptions.
- **R3.2. Effectiveness in matching event and subscription semantics.** This entails the accurate assessment of the semantic relevance between events and subscriptions containing heterogeneous terms that might describe the same or highly related semantic concepts.
- **R3.3. Effectiveness in mapping event and subscription semantics.** This entails mapping arbitrary predicates and event attributes that share semantic properties to the same nodes in a large structured peer-to-peer overlay network.

3.4.2.1 Low Cost in Defining Rules

In a peer-to-peer network where participants constantly join and leave, reaching an agreement on the training datasets used may not be possible. No one dataset covers all the concepts that can be recognized at the scale of human ability. In addition, the same semantic concept can be described using different terms by different datasets. This aspect yields a matching model based on term-level full agreement insufficient. Besides, even if we were to take advantage of the hierarchical semantic structures offered by some large-scale datasets, at the scale of 10s of thousands of semantic concepts, there is high difficulty in creating mappings across different semantic hierarchies. Therefore, due the dynamics of the network, the multiplicity of available training datasets, and the wide semantic scale of human-level recognition, the publish/subscribe system should facilitate low barriers to entry for participants in terms of establishing semantic agreements on terms present in exchanged events and subscriptions.

3.4.2.2 Effectiveness in Matching Event and Subscription Semantics

Following a content-based matching model, multimedia events are first disseminated in their native unstructured formats. Classifiers and/or object detectors would then perform a knowledge extraction task by placing a finite number of labels describing the published multimedia content. On the other end, subscribers submit expressions as conjunctive predicates comprising a number of labels representing semantic concepts that describe the multimedia content they're interested is receiving. The challenge at hand concerns assessing the relevance between events and subscriptions containing heterogeneous and arbitrary terms that might refer to the same of highly related semantic concepts. Conventional approaches that perform exact string comparisons between the terms present on both ends would not suffice and would result in a high numbers of missed matches, i.e. false negatives.

3.4.2.3 Effectiveness in Mapping Event and Subscription Semantics

In a structured peer-to-peer network offering a publish/subscribe communication service, events and subscriptions may be submitted by any participating node in the network. For the matching process to take place, events and subscriptions should be mapped to the same node in the overlay. Section 3.4.1.2 discussed the limitation of conventional key-based routing primitives and how they should be replaced with semantically decoupled lookup operations. However, in order to map events and subscriptions comprising multiple labels (i.e. attributes), such routing primitives should be aided with sufficient selective routing strategies that focus on maximizing the hit ratio while keeping messaging overhead to a minimum.

3.5 Summary

This chapter elaborates on the three research questions tackled in this thesis along with their associated technical requirements (Table 3.1). First, I present two motivational scenarios that showcase the problem under semi-centralised broker overlays and full decentralized peer-to-peer overlays. I then discuss the challenges associated with supporting multimedia event types in federated broker overlays. The focus is on the integration of operators that extract semantic concepts from multimedia content and how

Requirement	Contribution	Related Work	Associated Publications
R1.1. Efficient extraction of semantic concepts	The integration of fast binary classifiers as Boolean event filtering operators into the fo-rwarding function of federated broker overlays	NoScope[46], Optasia [37, 38], Mobilenets [151]	[40]
R1.2. Decomposable workload and adaptive routing decisions	An adaptive and distributed event matching algorithm and routing protocol that exploits the ordering and placement of classifier executions across a federated broker overlay	Siena [33], Padres [16], Shared-filter Ordering [59, 60], Stream Processing [50]	[40]
R1.3. Relay-free event propagation	We modify the organization of brokers in an overlay network in response to workload variations across brokers and shared filter executions	Interest Regionalism [62–65]	NA
R2.1. Organization and mapping of semantic data	We replace logical key-spaces underlying DHTs with a semantic vector space. We partition the space via hierarchical clustering resulting with traversable groupings of word embeddings that fall in semantic proximity to each other.	CAN [7], Word2Vec [158], Hierarchical Clustering [159, 160], Jarvis Patrick Clustering [161]	[47], [48]
R2.2. Semantically decoupled lookup operations	A lookup protocol that routes the vector representation of a data item v, to the node retaining the partition where v belongs in a few routing hops	Chord [117], Pastry [6], Kademlia [5]	[47]
R3.1 Low cost in defining rules	A distributional semantic event and language model where participants in a peer-to-peer network establish loose agreements on large corpora of texts.	Mapping Tables [52], Qian et al. [53], S-G-ToPSS [56, 72]	[47], [48]
R3.2. Matching event and subscription semantics	A decentralized content-based approximate semantic event matching model. We perform pairwise comparisons between the vector representations of terms present in events and subscriptions	Hasan et al. [55, 55]	[48]
R3.3. Mapping event and subscription semantics	A hybrid routing approach that combines the efficient and selective nature of rendezvous routing protocols with the probabilistic and dynamic nature of gossip routing protocols.	Tera [94], Sub2Sub [66], Vitis [68], Poldercast [19], Stan [126], SpiderCast [125], and Rappel [127]	[48]

TABLE 3.1: Requirements and Contributions

they impact the event matching and routing requirements. In particular, the broker overlay should ensure fast and resource aware knowledge extractions, a decomposable workload, adaptive routing decisions, and relay free event propagation. I then delve into the challenges associated with structured peer-to-peer overlays, which can be split into two main challenges. The first concerns semantic coupling in distributed hash tables, whereas the second focuses on establishing loose agreements between publishers and subscribers on event semantics, matching event and subscription semantics, and provisioning efficient and effective routing strategies.

Chapter 4

Adaptive Filtering and Dissemination of Multimedia Content in Federated Broker Overlays

4.1 Introduction

This chapter tackles research question (Q1.) and proposes an encompassing approach targeting its three technical requirements stated below.

Q1. How to integrate fast and resource-aware knowledge extractors into the forwarding function of distributed publish/subscribe systems comprising federated broker overlays?

Technical system requirements:

- **R1.1. Efficient extraction of semantic concepts.** Timely and resource-aware materialization and filtering of raw images that scales well with the increase in the number of subscriptions.
- **R1.2. Decomposable workload and adaptive routing decisions.** This entails the provisioning of event matching and routing techniques that facilitate a decomposable processing workload as mutually independent operations that leverage the distributed nature of federated broker overlays.
- **R1.3. Relay-free event propagation.** This entails eliminating or minimizing the participation of relay nodes in the event matching and routing process.

The chapter is organised as follows. Section 4.2 discusses related work. It elaborates on the strengths the limitations of three main categories of approaches, namely, content-based publish/subscribe broker overlays, optimized neural network inference over video databases, and continuous queries over commutative Boolean filters. Following that, section 4.3 formalizes the problem, provides a cost model comprising the main metrics that need to be minimized, and provides an overview of the proposed approach. Section 4.4 details the proposed solution and elaborates on the implementation details of its various components. In particular, section 4.4.1 discuses the event matching and distribution component which implements an adaptive greedy algorithm for the ordering of execution and placement of binary classifiers across the overlay network, section 4.4.2 discusses the topological reconfiguration component and elaborates on two proposed metrics for quantifying the similarity between brokers. Section 4.5 provides an empirical evaluation of the proposed algorithms. Section 4.5.1 discusses the methods employed in the experimental setup, which includes, the simulation environment, the construction of an overlay based on real-world traces, and building binary classifiers using optimized neural network architectures. Section 4.5.2 provides and analyses the experimental results. Finally, section 4.6 summarises the chapter and highlights the contributions of this work.

4.2 Related Work

The following categories of approaches are recognized.

4.2.1 Content-based Publish/Subscribe Broker Overlays

Prior work on improving the performance of distributed content-based publish/subscribe systems manipulate three main dimensions that highly impact in-network processing costs and event delivery latency, namely, event matching, event routing, and overlay design. In the following, I discuss different approaches for the realization of each dimension and how they interplay towards improving the efficiency of the entire system.

Content-based Event Matching Algorithms: The content-based publish/subscribe event matching problem generally comprises a set of subscriptions and an incoming stream of events. A subscription s consists of a collection of predicates each of which is a triple consisting of an attribute, a value, and a relational operator (e.g., $>$, $<$, $=$, etc.). An event e on the other hand, consists of a set of tuples, each comprising an attribute:value pair. An event e satisfies a subscription s if every predicate in s is matched by some tuple in e. Well known algorithms for event matching can be split into two main classes, namely, predicate indexing algorithms [61, 162–164] and tree-based algorithms [98, 99, 165]. The former conveniently allocate subscription predicates into forwarding tables by leveraging criteria such as the intersection, coverage, and popularity of predicates. Whereas the latter, insert the set of subscriptions into a matching tree where internal nodes and edges represent individual predicates and leaf nodes represent subscriptions.

Fabret et al. [162] propose a cache conscious matching algorithm that leverages an efficient data structure comprising a set of indexes, a predicate bit vector and a vector of references to subscription cluster lists. To cluster subscriptions, they group them based on their size and common equality predicates so that several subscription attributes can be evaluated using a single comparison. Carzaniga et al. [61] propose an optimized counting algorithm for event matching based on predicate indexing. The algorithm relies on a dictionary data structure that maintains the different Boolean predicates indexed based on the specific properties of constraint operators (i.e. $=$, $<$, $>$, etc.). During event matching, the algorithm maintains two running data structures, the first structure is a table of counters for partially matched filters, the second data structure is a set containing the interfaces (i.e. neighbours) to which the message should be forwarded. They leverage the early identification of matched interfaces to shortcut the matching process by eliminating all subscriptions that came through the same interface. They halt the matching process at a broker once at least one subscription was matched from each neighbouring broker. More recently, Qian et al. [163] focused on quickly filtering out unlikely matches. They propose REIN (REctangle INtersection) where they transform the event matching problem into a rectangle intersection problem. Rectangles are defined over an m-dimensional space where m is the number of attributes present in events. A rectangle r intersects another rectangle r' if r and r' have at least one point in common. A concise index structure is utilized to efficiently address the rectangle intersection problem by using bit operations. Contrary to identifying matched subscriptions by counting satisfied constraints (e.g. SIENA [61]), REIN initializes a bit set where the number of bits equals the number of subscriptions, it first marks unlikely matched subscriptions in the bit set by applying efficient bit operations. The unmarked bits represent the matched subscriptions. Zhang et al. [164] propose OpIndex, a two-level partitioning scheme to organize the subscription predicates into disjoint subsets each of which is independently and efficiently indexed to minimize the number of candidate subscriptions accessed for event matching. Each subscription is associated with a pivot attribute. Subscriptions are first partitioned based on their pivot attributes into subscription lists, and the predicates in each subscription list are then further partitioned based on the predicate operator into predicate lists. Each predicate list is then independently indexed using an efficient method that is appropriate for the predicate operator. To identify matching subscriptions for an input event e, the candidate pivot attributes are first enumerated from the set of distinct attributes in e. All attribute:value pairs in e are then evaluated against the relevant predicate lists that were indexed in the previous step.

Aguilera et al. [98] follows a tree-based approach to event matching. The matching algorithm initially pre-processes the subscriptions into a tree structure. The event model is based on attribute-value pairs where each subscription is a conjunction of elementary predicates, each predicate represents one possible result of an elementary test. They define an elementary test as a simple operation on one or more attributes of an event. In a matching tree: leaves represent subscriptions, non-leaf nodes contain individual elementary tests, and outgoing edges represent the results of the tests. An event traverses down the tree by following all matching paths and reports a matching subscription for each leaf node reached. Kale et al. [99] argue that tree-based algorithms tend to computational outperform matching algorithms based on predicate indexing. They observe that in real-world publish/subscribe applications, many events have only a few relevant properties by which they will be matched. First, they define the light query model as a binary model used to quantify relevant properties in events. They partition the content space taking each partition as a binary property that maps to relevant attributes where subscriptions are also partitioned based on their interest in the content space partitions. Then, they propose RAPIDMatchTree, a tree-based mechanism where they build a search tree per subscription set. Upon reception of an event, the algorithm identifies relevant properties and maps the event to its corresponding search trees. Hence, minimizing the number of subscriptions to evaluate against the event. Finally, Qian et al. [165] propose H-Tree, an index structure for subscriptions that is a combination of hash lists and hash chaining. The value domain of each indexed attribute is divided into cells on which a hash list is built. Multiple hash lists are then chained into a hash tree. The intuition is to design an index structure which is capable of filtering out most unmatched subscriptions. By chaining multiple hash lists, subscriptions that finally need to be checked are exponentially decreased with the number of chained indexed attributes.

Event and Subscription Routing Algorithms (R1.2.): Matching is a way to support routing decisions. Its scalability is impacted by the event matching workload, that being, the event input rate and the number of subscriptions. Routing can have an effect on the total matching cost and the time for events to reach interested subscribers. As depicted in Fig. 4.1, routing controls the propagation of subscriptions and events throughout the system. The number of brokers, subscriptions, and events should not cause a serious degradation in performance. Hence, approaches aim for limiting event propagation to brokers hosting interested subscribers and for reducing the amount of

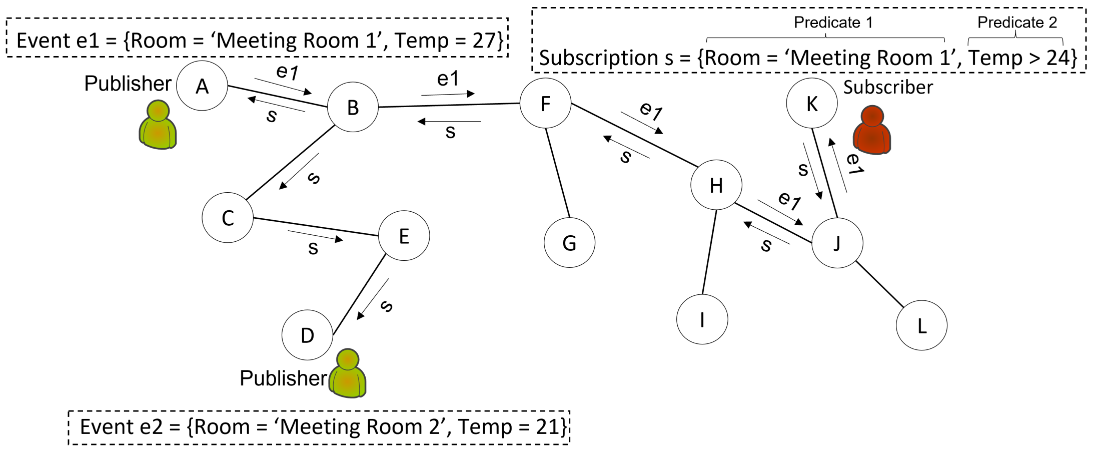

Event e1 = {Room = 'Meeting Room 1', Temp = 27}
Subscription s = {Room = 'Meeting Room 1', Temp > 24}
Predicate 1
Predicate 2
Publisher
Subscriber
Publisher
Event e2 = {Room = 'Meeting Room 2', Temp = 21}
A
B
C
D
E
F
G
H
I
J
K
L
e1
s

FIGURE 4.1: An illustration of a broker overlay employing Filtering-based event routing.

routing information (i.e. subscriptions) maintained at each broker. Event and subscription routing solutions targeting broker overlays can be classified into three main categories: Flooding algorithms, Filtering-based algorithms, and Multicast-based algorithms. I refer the reader to section 2.3.2.1 for a detailed breakdown of each approach.

Topological Reconfiguration Approaches (R1.3.): The cost of event processing can be influenced by the number of hops it takes for an event to reach interested subscribers. The performance of the system is sometimes degraded due to redundantly matching events along paths towards subscribers. By eliminating pure-forwarders and placing brokers with similar interests within a limited number of hops, prior approaches were able to reduce the total matching and routing costs. Pure-forwarders are brokers that participate in event matching while having no locally interested subscribers. The elimination of such brokers has been attained by modifying the overlay structure and creating interest regionalism [62–65]. In these approaches, brokers of similar interests are wired together thus reducing the publication forwarding overhead. The notion of broker similarity is determined based on the amount of shared subscriptions and event traffic. Chockler et al. [65] first introduced the Minimum Topic-Connected Overlay (Min-TCO) problem which aims to strike a balance between the scalability of the overlay (i.e. node fan-out) and the message forwarding overhead incurred by individual brokers. Topic connectivity is achieved using a distributed protocol that maintains a separate overlay per topic. A Min-TCO aims to minimize the number of links needed to create a topic-connected overlay for the given interest assignment. They prove that the problem is NP-Complete and propose the Greedy Merge (GM) algorithm which provides an approximate solution that iteratively finds the next best edge that maximizes topic connectivity. Chen et al. [64] propose divide and conquer (DC) algorithms that efficiently solve the problem of dynamically joining two or more topic-connected overlays.

The aim is to tackle three main drawbacks of past approaches, (1) prohibitively high running time cost, (2) requirement of full system knowledge and centralized operation, and (3) constructing overlay from scratch. They propose the Star Merge algorithm which constructs the TCO with a logarithmic approximation ratio compared to the optimal solution. Their DC algorithm successfully produces low average node degree TCOs with significantly lower running time cost than alternative solutions. Baldoni et al. [62] propose a distributed on-line heuristic that automatically adapts the structure of the broker overlay network to the communication costs of the links, the processing costs of brokers, and the patterns in the notification flows between brokers. The intuition is to reduce the distance between brokers that consume a lot of identical notifications, while respecting communication and processing costs. The notion of shared interest between brokers is defined based shared event traffic (i.e., the notifications that two brokers consume or publish). They introduce a cache in every broker to keep track of delivered or received notifications. They use a bloom filter to represent the cache of each broker. A Bloom filter is a space-efficient probabilistic data structure that helps reduce the amount of exchanged data between brokers. Jaeger et al. [63] studied the effect of the introduction of a self-organizing algorithm in the context of a specific system implementing a tree-based routing strategy (i.e. SIENA [15]). To assess the similarity between two brokers B_i and B_j, they compute a probability measure that captures the ratio of events matched by B_i that also match subscriptions registered at B_j. They propose a convenient overlay reorganization algorithm that only exploits information local to each broker. Furthermore, they aim to strike a balance between optimizing overlay hops and network-level metrics, like latency or bandwidth, whereby a reconfiguration of the overlay network does not spoil the initial overlay topology with respect to network-level metrics.

Discussion: Conversely to the discussed approaches, the goal of this work is to support multimedia content as native event types. This requires fitting brokers with appropriate content extractors to facilitate the matching process **(R1.1.)**. However, the cost of uncovering meaningful semantic concepts from raw unlabelled multimedia content may impose a burden on the scalability of the system. Hence, the aim is to strike a balance between matching, routing, and overlay design, in order to minimize processing costs while enhancing the efficiency and expressiveness of distributed publish/subscribe systems comprising federated broker overlays.

4.2.2 Optimized Neural Network Inference over Video Databases

Recent advances in computer vision backed by deep learning breakthroughs have helped surpass human accuracy in detecting and labeling objects in images and videos. Employing advanced CNN architectures for inference, however, is a costly process requiring

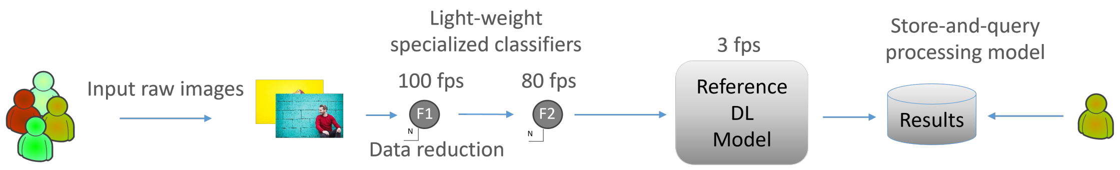


FIGURE 4.2: Inference optimization via light-weight specialised filters in large-scale image and video databases.

expensive resources such as GPU equipped devices or server clusters in cloud environments [46]. In response, recent approaches have focused on inference optimization to handle selective user queries on large video databases. As depicted in Fig. 4.2, this has been achieved by devising light-weight specialized image classifiers with a trade-off between accuracy and latency to be pushed to the start of a query processing pipeline in a manner similar to the conventional predicate push-down optimization **(R1.1.)**.

For instance, NoScope [46] proposes the use of model specialization as a way to build smaller models that can mimic the behaviour of a reference deep CNN on a particular task. Instead of passing all video frames through a generic model that can detect thousands of classes, they were more interested in identifying a handful of objects that can be initially probabilistically detected by much less costly models as a method for intelligent load shedding. Lu et al. [38] later extended NoScope by exploring the design space of specialized filters and using both CNN architectures and more traditional machine learning methods such as SVMs and Kernel Density Estimators (KDEs) for a range of different tasks and data types. Furthermore, Anderson et al. [58] manipulate the prepossessing phase of the pipeline by treating the physical representation of input images as part of the query optimization process. More precisely, they explore techniques for image transformations such as resolution scaling and color-depth reduction.

Discussion: The major limitation with the aforementioned approaches is that they all follow a store-and-query model with per-query optimizations. This opposes the goal of serving long-standing queries in the form of user subscriptions across a broker overlay. The high number of potential publishers and subscribers and the dynamic and distributed nature of publish/subscribe broker overlays stand in the way of designing specialized models per subscription. These aspects render this category of approaches infeasible considering the real-time processing requirements of publish/subscribe systems.

4.2.3 Continuous Queries over Commutative Boolean Filters

Prior to the adoption of deep learning as the main driver for conducting computer vision tasks, practitioners used to rely on building cascades of binary classifiers using

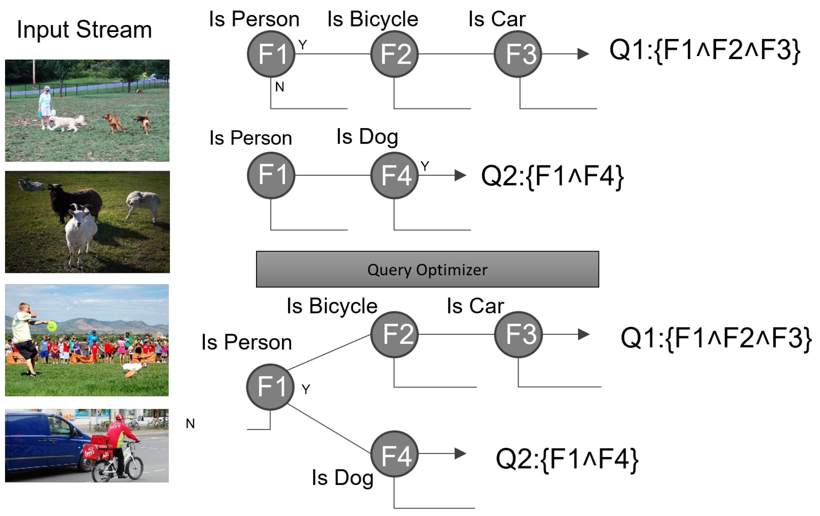


FIGURE 4.3: Utilizing filter sharing and ordering to optimize the processing of overlapping queries.

traditional machine learning techniques. This can be attributed to the complexity of building models that can detect multiple objects (i.e. multi-class classifiers and object detectors) by manually identifying the features for every object or semantic concept. Fig. 4.3 shows how the use of binary classifiers to answer user queries represented as conjunctions of filters allows for leveraging the sharing and ordering of classifier executions to enhance system performance in response to overlapping queries **(R1.2.)**. This problem is known in the literature as the Shared-Filter Evaluation Problem [59].

The Shared-Filter Evaluation Problem consists of a set of queries Q, and a set of commutative Boolean filters F. Each query is represented as a conjunction of filters where the evaluation of each filter on an event results in a Boolean outcome of either *true* or *false*. The goal is to find a near-optimal ordering of filter executions to resolve all the queries that exist in the system. The problem was premiered and formally defined by Munagala et al. [60], they identify three important factors that decide the utility of executing individual filters.

- **Cost**: c_i denotes the time cost required by a filter f_i to process an event e. Intuitively, less costly filters should be evaluated earlier.
- **Selectivity**: x_i denotes the probability that an event e will satisfy filter f_i resulting in a *true* outcome. Executing filters with high selectivity earlier would result in more data reduction.
- **Popularity**: p_i denotes the participation of a filter f_i in the registered subscriptions. It's intuitive that filters with high popularity should be executed earlier.

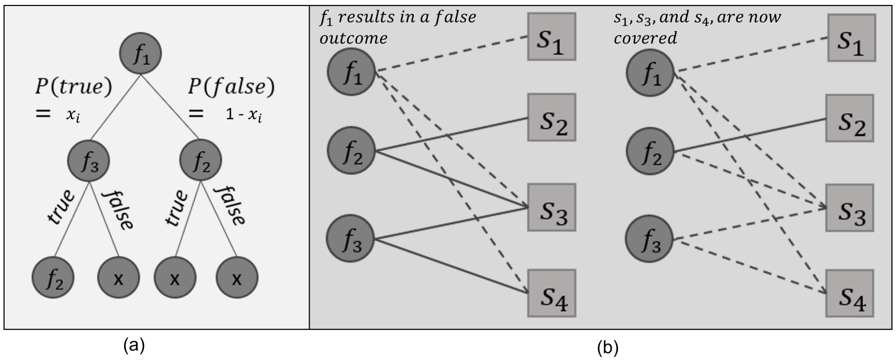


FIGURE 4.4: Adaptive filter evaluation on the basis of a binary decision tree (a) or a bipartite graph by leveraging edge-coverage (b).

Munagala et al. [60] identify that the problem is NP-hard and that the nearest approximation can be achieved with an adaptive strategy that greedily decides the next filter to execute. They propose an adaptive greedy algorithm that constructs a decision tree on the basis of a pRank metric to determine the order of execution of filters (Fig. 4.4 (a)). The metric is calculated as $pRank(f_i) = \frac{c_i}{p_i(1-x_i)}$. At each iteration, the algorithm picks the filter that is expected to resolve the maximum number of unresolved queries per unit cost (i.e. the filter with the lowest pRank). Liu et al. [59] later identify that the outcome of the current filter being executed has an impact on the popularity of the remaining filters. For instance, provided that queries are conjunctive, a single *false* outcome is enough to decide that an event doesn't match an entire query, which allows for the elimination of a number of queries hence changing the popularity of the remaining filters. As depicted in Fig. 4.4 (b), they model the problem as a bipartite graph (i.e. Residual Graph) that maps queries to filters, they order filter evaluations based on a metric composed of their cost, selectivity, and edge-coverage in the generated graph. The notion of edge-coverage helps determine the impact of filter executions on the remaining queries. Recently, the work by Xie et al. [166] focuses on the variation in data streams and its impact on the selectivity of filters at run-time. They propose a framework for monitoring and updating the selectivity of filters. For single filter monitoring, they record the estimation outcome in a window, estimate its selectivity, and determine whether it should be updated by applying confidence interval.

Discussion: The limitation with this category of approaches is that they all assume a single processing engine that connects all the data producers and consumers and resolves all queries registered in the system. On the contrary, in a distributed publish/subscribe scenario, the system comprises a federated overlay of brokers. With a high number of publishers and subscribers, even an optimal ordering of filter executions at a single broker can generate bottlenecks and degrade system performance.

		Content-based Pub/Sub Broker Overlays			Optimized NN Inference over Video Databases		Continuous Queries over Commutative Boolean Filters	
		Event Matching: Predicate Indexing [61, 162–164], Tree-based [98, 99, 165]	Filtering-based Routing [15, 16]	Interest Regionalism [62, 63]	NoScope [46], Lu et al. [38]	Anderson et al. [58]	Munagala et al. [60], Liu et al. [59]	Proposed Model
R1. Content Extraction	Method	NA	NA	NA	Specialized light-weight content extractors	Image resolution scaling & color-depth reduction	Commut-ative boolean filters	Distributed commut-ative boolean filters
	Efficiency	NA	NA	NA	Medium to High	Medium to High	Medium to High	High
R2. Workload Decompos -ability	Method	Redundant matching	Adaptive routing by short-cutting the matching process	NA	Combining light-weight classifiers with large DL models	NA	Exploit shared-filter ordering	Exploits shared-filer ordering & placement
	Efficiency	High	High	NA	Medium to High	NA	Medium to High	High
R3. Relay-free Event Propagation	Method	NA	Advertisement -based selective routing via back propagation	Overlay reorganization based on shared event traffic	NA	NA	NA	Overlay reorganization based on shared filter executions
	Efficiency	NA	Medium	Medium to High	NA	NA	NA	Medium to High

Table 4.1: Current Approaches

4.2.4 Discussion and Analysis

Based on our study of the state of the art (Table 4.1), we identify a number of limitations with past approaches that we aim to overcome in this thesis. The first category comprises approaches on improving the efficiency of content-based publish/subscribe broker overlays. We further split those approaches into three groups, namely, predicate indexing and tree-based event matching algorithms, filtering-based event routing protocols, and techniques for creating interest regionalism via overlay topology reorganization. We identify that such approaches lack support of unstructured multimedia event types (R1.1.), but generally offer efficient methods for workload decomposability, adaptive routing decisions (R1.2), and relay-free event propagation (R1.3.). As later discussed in the chapter, we adopt the adaptive routing technique utilized by Siena [61] to shortcut the processing of subscriptions. We also augment techniques for overlay topology adaptation [62, 63] by adopting a novel broker similarity estimation measure based on shared filter executions across brokers. The second category of approaches focuses on optimized neural network inference over video databases. We identify that the surveyed approaches utilise optimized neural network architectures and traditional machine learning techniques (e.g., SVMs, KDEs, etc.) to maximize content extraction throughput and lessen memory footprint. The proposed models offer efficient techniques for real-time content extraction (R1.1.) and intelligent load shedding, but lack support for adaptive routing (R1.2) and relay-free event propagation (R1.3). We re-purpose the proposed specialised lightweight content extractors to work over a distributed publish/-subscribe system comprising federated broker overlays as opposed to the store-and-query model presented by Kang et al. [46] and Lu et al. [38]. Finally, the third category of approaches comprises techniques for optimizing the execution of commutative Boolean filters against overlapping queries. Such approaches are highly generalizable in terms of content extraction (R1.1.). The Boolean filters used by Munagala et al. [60] and Liu et al. [59] can be replaced with more recent light-weight neural network architectures. Moreover, filters being Commutative (i.e. mutually independent) offers a highly efficient technique in terms of workload decomposability (R1.2). However, surveyed approaches assume a centralized environment where a single server has access to all the filters and processes all the queries that exist in the system. On the contrary, in a broker overlay setting, the workload has to be intelligently distributed across the available resources to be able to answer a growing number of subscriptions under near real-time latency constraints.

4.3 Proposed Model and Problem Statement

We propose embedding optimized content extraction models (i.e. binary classifiers) into the forwarding function of brokers in an overlay network. The binary classifiers work together in a *commutative* manner to support multimedia content as native events types. In distributed publish/subscribe systems comprising federated broker overlays, a routing substrate is commonly in place to control the propagation of subscriptions and events. We leverage the distributed nature of the system to design a forwarding algorithm that distributes the ordering and execution of classifiers (i.e. filters) along various paths towards subscribers. The identified problem can be referred to as the shared-filter placement and ordering problem.

The shared-filter placement and ordering problem consists of a set of subscriptions S, a set of commutative Boolean filters F, and a set of brokers B. Each subscription is represented as a conjunction of filters where the evaluation of each filter on an event results in a Boolean outcome of either *true* or *false*. The objective is to minimize the Processing Delay Cost which may be defined as the time required by filters executed along a path towards a subscriber to process an event. Let B be the set of n brokers comprising the path between a subscriber s and the corresponding publisher of interest rooted at broker $B_0 \in B$. Each broker $B_i \in B$ executes a subset of all the required filters according to the employed execution plan. Let c_{B_i} be the cost (i.e delay) at broker B_i, then the cost of an execution plan P as perceived by s is given by:

$$cost(s, P) = \sum_{i}^{n} c_{B_i} \tag{4.1}$$

The shared-filter ordering problem in distributed publish/subscribe systems can be formally defined as follows:

Definition 4.1. Given a set of subscriptions S, a set of filters F, and a set of brokers B. Find a shared execution plan P that minimizes $cost(S, P)$.

Since the filtering workload will be distributed across various brokers, the same filter might occur multiple times at brokers belonging to separate paths. This introduces a redundancy overhead trade-off which can be measured as the total processing cost for a notification $n \in N$ along all paths towards interested subscribers. A poor overlay topology would lead to increasing the redundancy overhead due to a high degree of sparsity in the distribution of subscribers with similar workload on separate paths. This problem can be formally defined as follows.

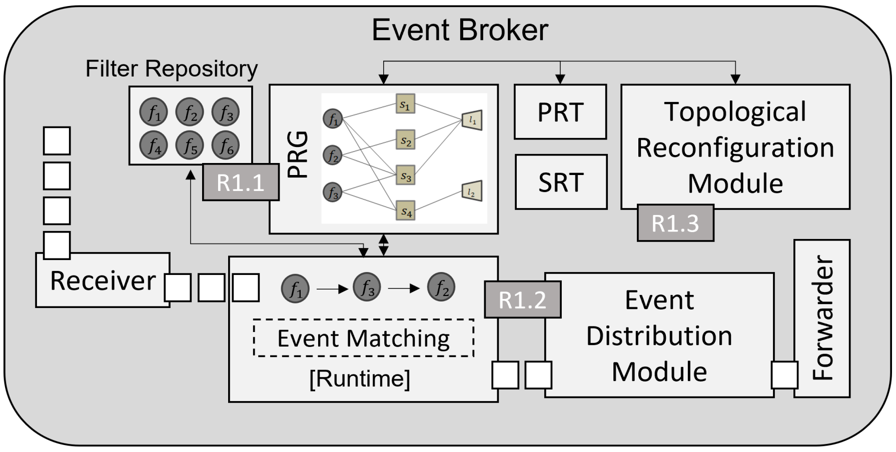


FIGURE 4.5: Broker Architecture.

Definition 4.2. Given a Graph $G(V, E)$, V representing nodes in the graph (i.e. brokers), and E representing edges (i.e. links). And given a set of notifications N. Find an acyclic topology T of G that minimizes $cost(N, T)$.

4.4 Distributed Multimedia Event Processing Model

The proposed solution comprises three main techniques, an event matching model based on a cascade of binary classifiers, an event distribution protocol that utilises a novel publication routing graph for event forwarding, and a topological reconfiguration algorithm that utilises a novel broker workload similarity estimation technique. Fig. 4.5 shows the different components comprising an event broker and how they interact to ultimately match and route incoming unstructured multimedia events throughout a broker overlay network.

4.4.1 Matching and Distribution of Multimedia Events

As presented in Padres [16] and Siena [61], filtering-based event routing in content-based publish/subscribe broker overlays can be split into three stages: Advertisement flooding, a deterministic reverse path propagation of subscriptions, and a selective propagation of events. As depicted in Fig. 4.6, when publishers join, they issue advertisement messages that provide schematic information about events they intend to publish. On reception of an advertisement, a broker registers it along with the link it came from in a local Subscription Routing Table (SRT), then forwards it to all its neighbours. Subscribers interested in receiving published events send their subscriptions to their connecting broker. Subscriptions are then evaluated against advertisements in the local SRT and routed

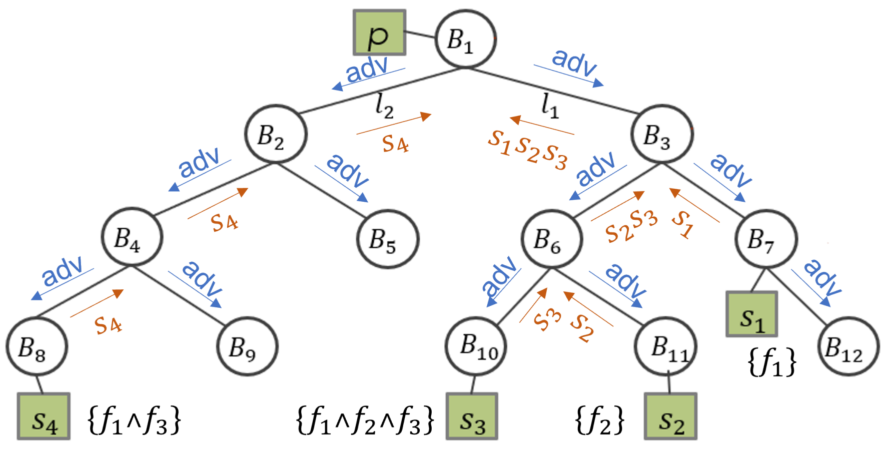


FIGURE 4.6: Filtering-based event routing in the form of advertisement flooding followed by a deterministic reverse path propagation of subscriptions towards publisher-end brokers.

along reverse paths towards publisher-end brokers. At each broker, received subscriptions are registered along with the links they came from in Publication Routing Tables (PRTs). Each broker is then aware of a subset of all the subscriptions based on its position in the overlay. Event matching is carried out in each broker by comparing subscription predicates against numerical or textual event attributes; on identification of a match, the notification is forwarded to the neighbouring broker where the matching subscription came from. The matching process is then repeated at each broker along the various paths towards interested subscribers. To this end, the introduction of content extractors (i.e. filters) challenges this match-and-forward model. More precisely, executing a portion of the required filters at a broker would enrich a multimedia event by uncovering some of the semantic concepts it represents. Consequently, downstream brokers receiving the event need to be aware of the concepts previously identified and the filters executed in order to avoid redundant processing and carry on with the matching process. In what follows, I outline how the proposed event matching and routing process is carried out.

4.4.1.1 Publication Routing Graph

To maintain the loosely-coupled nature of interaction in publish/subscribe overlays, the adaptive ordering of filter executions should only consider the local information available at each broker. Consequently, based on each broker's PRT, a Publication Routing Graph (PRG) is constructed as two connected bipartite graphs. The PRG is used to adaptively order and distribute the execution of filters across brokers in the overlay. It facilitates the event matching and routing process. Fig. 4.7 shows the PRGs of brokers B_1 and B_3 (See Fig. 4.6). The left-hand side of the graph (i.e. first bipartite graph) represents the

PRT (B_1)	
$S_1 = \{f_1\}$	$l_1 \Rightarrow B_3$
$S_2 = \{f_2\}$	$l_1 \Rightarrow B_3$
$S_3 = \{f_1 \wedge f_2 \wedge f_3\}$	$l_1 \Rightarrow B_3$
$S_4 = \{f_1 \wedge f_3\}$	$l_2 \Rightarrow B_2$

PRT (B_3)	
$S_1 = \{f_1\}$	$l_1 \Rightarrow B_7$
$S_2 = \{f_2\}$	$l_2 \Rightarrow B_6$
$S_3 = \{f_1 \wedge f_2 \wedge f_3\}$	$l_2 \Rightarrow B_6$

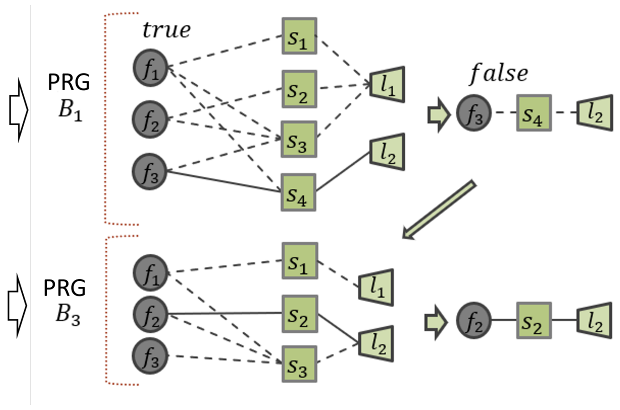


FIGURE 4.7: The Publication Routing Tables (PRTs) and Publication Routing Graphs (PRGs) of broker B_1 and B_3.

Residual Graph proposed by Liu et al. [59] (Fig. 4.4 (b)). An edge between two nodes in the graph simply resembles that the filter f_i is part of the conjunction of filters required by the subscription s_i. The right-hand side (i.e. second bipartite graph) represents a mapping between subscriptions and the links they were received through in the overlay.

4.4.1.2 Match-and-forward Edge-Coverage-based algorithm (MEC)

The Event Matching component implements the MEC algorithm. The goal of MEC is to adaptively determine what filters to execute at the current broker, their order of execution, and what filters to defer to downstream brokers. The MEC algorithm starts with an initial PRG at the publisher-end broker and evaluates filters one-by-one in a sequential order then terminates when all edges have been resolved (i.e. covered). If a filter f_i's evaluation outcome comes out as *true*, only the outgoing edges denoted Δ_t are covered. This essentially means that the associated subscriptions (i.e. subscriptions including f_i in their conjunction of filters) are only partially matched. Otherwise, if f_i yields a *false* outcome, all its outgoing edges are covered along with all the outgoing edges of the associated subscriptions denoted Δ_f. Given the conjunctive nature of subscriptions, only a single *false* outcome from one of the filters they include is enough to identify that the current event doesn't yield a match, which allows us to eliminate the subscriptions from the PRG along with all their associated edges.

$$UP(f_i) = \frac{c_i}{x_i\Delta_t + (1 - x_i)\Delta_f} \tag{4.2}$$

At each step, the decision to execute a filter is based on its unit price $UP(f_i)$ as in equation (4.2). The filter with the lowest unit price is evaluated and then removed from the graph. A low unit price reflects a low filter execution cost in addition to higher expected edge coverage (i.e. popularity among subscriptions). Based on the outcome of the current filter being executed, all satisfied subscriptions are removed as well as their outgoing edges. The unit prices of all impacted filters are then updated to select the next filter in the adaptive ordering.

Graph Short-cutting: The algorithm shortcuts the evaluation of filters on the basis of a subscription-link mapping that helps extend the edge coverage of filters. If a link is satisfied by a matching subscription, it is also removed with its associated subscriptions and their edges. The intuition here is that any additional executions of filters required by subscriptions received through the same link would not provide additional information in terms of event forwarding. In other words, if an event matches one subscription associated with one of the links, there is no need to evaluate the same event against other subscriptions received through the same link. Consider the example in Fig. 4.7, B_1 is aware of subscriptions s_1, s_2, and s_3 through link l_1 and s_4 through l_2. The outcome of the first filter (f_1) is *true*, hence subscription s_1 and link l_1 are satisfied. This yields the removal of subscriptions s_2 and s_3 in addition to filter f_2 and by that short-cutting the execution of filters by deferring the execution of f_2 to B_3.

Extended Edge Cover (EMEC): Prior to executing a filter f_i having probability x_i of resolving a subscription, we incorporate the possible edges that can also be covered by link elimination into $\Delta_T(f_i)$. That is, all the edges associated with the other subscriptions that were received through the same link. This simple procedure results in more expected edge coverage and decreases the unit price of f_i (i.e. $UP(f_i)$). This ultimately impacts the adaptive ordering of filter executions and leads to faster evaluation.

4.4.1.3 Message Propagation Schema

When the local PRG is empty, the algorithm proceeds to attach the results to the event and forwards it through matched links. When an internal broker receives an event, it first fetches the results of prior filters, updates its local PRG, and proceeds to process the event using the resulting graph. By eliminating all subscriptions received through a link as soon as a subscription that came through the same link is satisfied, we end up with two important advantages: (1) The updated PRG at the current broker is tailored towards choosing the next best filter for satisfying the remaining links without considering subscriptions that can be answered by downstream brokers. And (2) the distribution of subscriptions across brokers gets sparser as we go further down the tree, and by deferring computations downstream, the popularity distributions of filters from the point of view of each broker will change and by that the decisions on filter ordering will be more tailored for the subset of subscribers in each broker's local PRT.

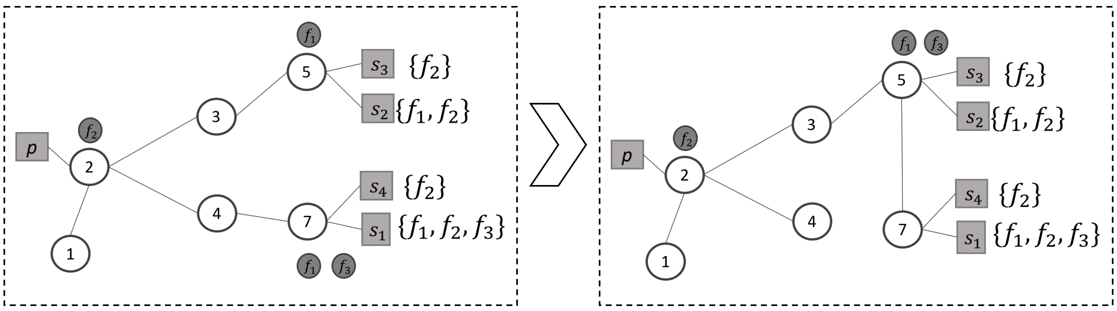


FIGURE 4.8: An illustration of modifying the overlay topology by identifying high workload similarity between brokers 5 and 7

4.4.2 Topological Reconfiguration

Based on the distributed filter ordering algorithm described in section 4.4.1.2, the early offloading of filters to downstream brokers by short-cutting the PRG is a result of prioritising the execution of filters with the highest possible edge-coverage. Each broker in the overlay has its own locally connected subscribers with a distribution of filters that can be different from other brokers leading to a different local priority ranking for filter executions. Offloading filter executions downstream is highly beneficial to speed up event processing and delivery and minimize latency as perceived by individual subscribers. However, it leads to the execution of the same filters on the same event multiple times at different brokers along separate paths in the overlay. This aspect increases the total processing cost which we refer to as the redundancy overhead. By creating regionalism in the overlay and grouping together brokers with a similar workload, we can enhance the performance by reducing the redundancy overhead. Prior works on overlay design focus on eliminating pure-forwarders which are brokers that participate in the processing of events but have no local interest in them. Regionalism in these works is created by connecting brokers that have the most overlapping traffic and the least non-overlapping traffic. However, the introduction of filters uncovers a limitation in this similarity metric because it does not capture how similar two brokers are in terms of the filters they execute. In the following, I discuss two methods that utilize vector distance measures to quantify how close two brokers are as per their subscription workload. On the basis of this similarity metric, the overlay topology can be modified by eliminating links between brokers with low similarity and provisioning new links between brokers with high similarity. These modifications also entail keeping the overlay connected, that is, ensuring the existence of a tangible path between every publisher and its associated subscribers.

4.4.2.1 Popularity Distribution Similarity

In this method we record the frequency of occurrence of every filter in the local subscriptions of a broker and load the values into a vector of n components corresponding to all the filters available in the system. To measure the similarity between two brokers we use the Bhattacharyya distance [167] which measures the similarity of two probability distributions.

4.4.2.2 PRG Impact Similarity

By only including the popularity of filters through their frequency of occurrence, we do not capture the impact of their execution on the local subscriptions. Given that we base the ordering on the PRG, we derive an impact measure from the unit price of a filter. More precisely, the impact of a filter f_i can be defined as its possible edge coverage given by $(x_i\Delta_t + (1 - x_i)\Delta_f)$ divided by the total number of edges in the PRG. Similar to the previous method, we load the impact of all filters into a vector and we measure the similarity between brokers using the Cosine Similarity measure which is a well-known measure for how close two vectors are in space by obtaining the cosine of the angle between them.

4.4.2.3 Overlay Construction

To construct the overlay we adapt Kruskal's Minimum Spanning Tree (MST) algorithm [168] by first calculating the distance between all brokers in the system using either of the similarity measures discussed above, then we rank them from the most similar to least, and we start to select edges in a descending order making sure at every step that (1) they do not introduce a cycle, and (2) they do not violate a preset maximum node degree. Fig. 4.8 shows how the overlay structure can be modified by identifying that brokers 5 and 7 are highly similar based on the filters they execute. In particular, based on the distributions of subscribers, the MEC algorithm executes filter f_2 at broker 2 (the publisher-end broker). A true outcome of f_2 satisfies subscriptions s_3 and s_4, hence there is no need to execute any more filters locally. The event is propagated towards the subscriber-end brokers which in turn execute filters $f1$ and f_3. As depicted in the figure, filter f_1 is executed twice at the two brokers due to having them on two separate paths with each having locally connected subscribers that need to be served. By eliminating the link between brokers 4 and 7, and introducing a link between brokers 5 and 7, we can make sure that the two brokers are now on the same path. This modification results

Label	Parameters $[\alpha, r]$	Load (ms)	Inference (ms)
Sofa	[1.0 * 224]	190	170
Motorcycle	[1 * 128]	190	140
Refrigerator	[0.75 * 192]	160	140
Sheep	[0.5 * 160]	110	118
Bus	[0.25 * 128]	75	95

TABLE 4.2: Load and Inference time of binary Classifiers based on MobileNets with multiple combinations of the Width Multiplier α and image Resolution r

in executing f_1 only once at broker 5 and by that reducing the processing redundancy overhead and improving the efficiency of the overlay structure.

4.5 Evaluation

In this section the proposed solution is empirically evaluated via simulation. To illustrate the performance benefits of applying the different heuristics, we implement two extreme baselines that resemble the upper and lower bounds for the shared filter placement and ordering problem in publish/subscribe broker overlays. The two baselines are based on Residual Graph [59]; we implement the edge-coverage based algorithm to be executed in publisher-end brokers and subscriber-end brokers respectively. Without adaptively distributing or offloading the execution of filters along intermediary brokers. In the first case, referred to as All in Root (AIR), publisher-end brokers carry out the adaptive ordering until all subscriptions are resolved before disseminating matched events. Whereas for the second baseline, namely All in leaves (AIL), all events are flooded towards subscriber-end brokers where the adaptive filter ordering is carried out based on local residual graphs that only consider directly connected subscribers.

4.5.1 Experimental Setup

This section details the methodology for overlay construction based on real-world traces, the filter prototypes used (i.e. binary classifiers), and workload generation (Fig. 4.9).

4.5.1.1 Overlay Construction

The overlay network implementation is based on a distributed publish/subscribe system simulation written in Java. We construct tree topologies based on real-world traces that provide network latencies across servers in the PlanetLab public research network. The dataset collected by Zhu et al. [169] includes round-trip latency readings between 490

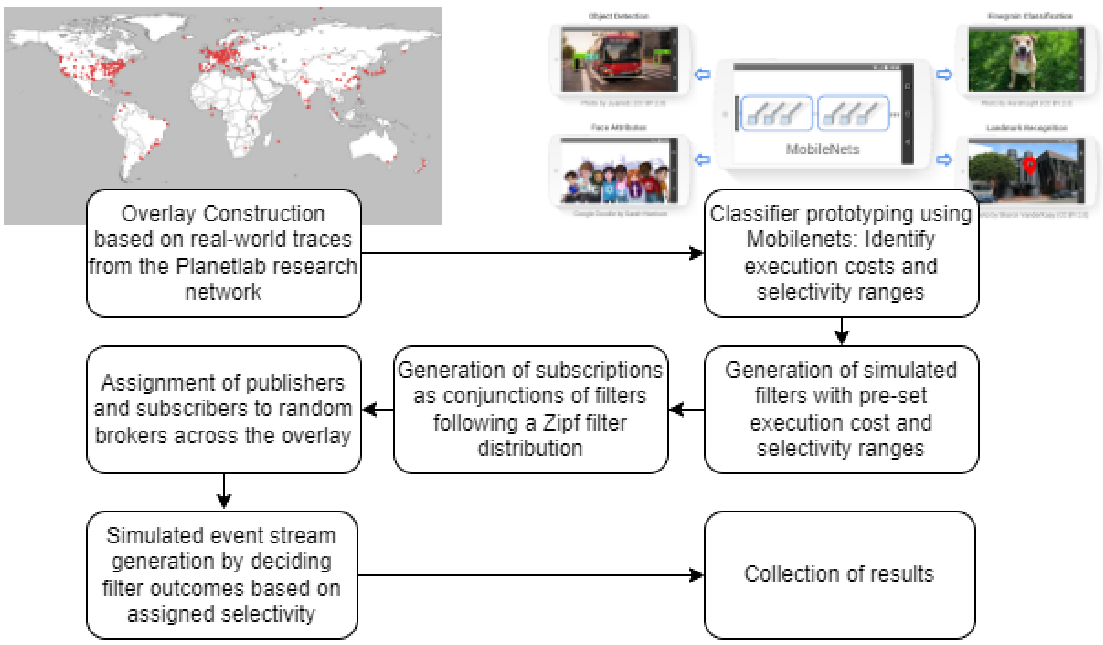

Overlay Construction based on real-world traces from the Planetlab research network
MobileNets
Classifier prototyping using Mobilenets: Identify execution costs and selectivity ranges
Generation of simulated filters with pre-set execution cost and selectivity ranges
Generation of subscriptions as conjunctions of filters following a Zipf filter distribution
Assignment of publishers and subscribers to random brokers across the overlay
Simulated event stream generation by deciding filter outcomes based on assigned selectivity
Collection of results

FIGURE 4.9: Experiment Workflow Chart

servers. It comprises several 490 $\times$ 490 matrices measured over multiple time slices, where the (i, j)-th entry of a matrix m_t indicates the measured Round Trip Time (RTT) from node i to node j at time t. We select 100 nodes at random to host the brokers and compute the Minimum-Spanning-Tree (MST) that connects them. We follow this process to create five different broker overlay topologies.

4.5.1.2 Binary Filters

To construct image classifiers with a good trade-off between cost and accuracy, we use MobileNets [151]. MobileNets is an efficient CNN architecture that achieves comparable accuracy to deep CNN models while allowing us to tune two hyper-parameters to build very small, low latency models. We build 40 binary classifiers with different configurations and object categories using a fine-tuned implementation of MobileNets that had been pre-trained on ImageNet. We select 40 classes from the ILSVRC 2014 [170] dataset and acquire up to 500 training images per category and roughly the same amount of images not belonging to the category of interest. For each classifier we vary two parameters out of 16 possible combinations made from the cross product of the width multiplier and resolutions. Table 4.2 shows 5 different instances of the classifiers we created and the impact of varied parameter combinations on the load and inference time.

4.5.1.3 Instance Generation

Based on the work by Mungala et al. [60], the filters are generated by specifying their cost and selectivity values based on a random distribution from preset value ranges. We specify the costs of individual filters out of the range $[12, 16]$ that was decided via extensive prototyping using MobileNets. The selectivity range was set to $[0, 0.25]$ because the matching ratio of subscriptions is known to be low [171] in the publish/subscribe literature. We then generate subscriptions comprising combinations of filters selected following a Zipf Distribution due the close resemblance between image filtering based on object category occurrences and text search engines and the inverse relationship between the frequency of word occurrence and its rank. Finally, we generate an event stream of consecutive event items that are simulated by deciding for each event whether each filter f_i has a *true* or *false* outcome based on its selectivity x_i. Subscriptions are distributed across brokers based on a load-value $\beta \in [0, 1]$ chosen randomly for each broker. For each run, the publisher issues 1000 events; each experiment instance is repeated 5 times using 5 different topology structures resulting in a total of 25 runs for each configuration.

4.5.1.4 Metrics

In our experiments we focus on measuring two metrics. The average Processing Delay Cost and the Total Processing Cost. The former captures the processing latency as perceived by individual subscribers (R1.1. & R1.2.), whereas, the latter captures the processing redundancy overhead (R1.3.).

The average Processing Delay Cost (PDC) is measured by taking an event e and summing the execution cost of filters evaluated over e along various paths towards subscribers. Every path e is propagated along exhibits a different ordering and placement of filter executions based on the distribution of subscribers across brokers. Hence, we calculate the average PDC by taking the total processing cost perceived by all subscribers and dividing it by the total number of notifications delivered. A low average PDC indicates efficient content extraction (R1.1), high workload decomposability by ensuring the distribution of filter executions across brokers, and efficient routing decisions (R1.2).

On the other hand, for a notification set N, the total processing cost is the sum of the execution costs of all filters evaluated against all $n \in N$ along all paths towards subscribers. The goal is to capture the processing redundancy overhead resulting from offloading the execution of filters to intermediate brokers, which may lead to to executing the same filter on the same event multiple times along various paths. Measuring the total processing cost quantifies the impact of modifying the overlay topology in response to the variation of filter execution workload across brokers (R1.3.).

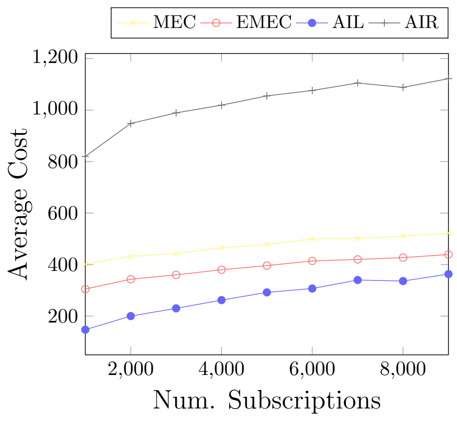


Figure 4.10: Shared filter ordering and placement under a Zipfian filter distribution

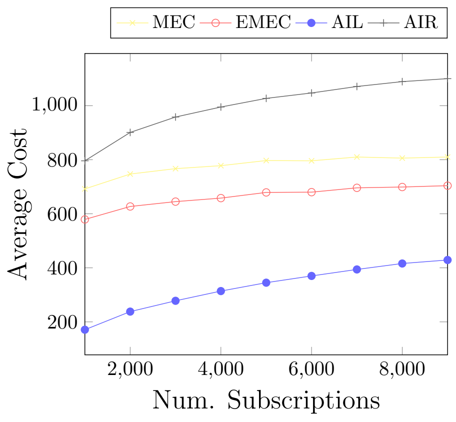


Figure 4.11: Shared filter ordering and placement under a Uniform filter distribution

4.5.2 Results

This section discusses the experimental results based on three main parameters: Number of subscriptions, number of filters, and topology structure.

4.5.2.1 Impact of Subscriptions

In the first experiment we vary the number of subscriptions while fixing the number of filters to 100. We measure the average processing delay cost along every path towards subscribers which accounts for the impact of the processing cost at each broker (i.e. filter executions) on the average perceived latency by subscribers while excluding communication costs. As shown in Fig. 4.10, the MEC and EMEC algorithms perform $50 - 70\%$ better than the AIR baseline when filter popularities follow a Zipfian distribution. The sharing degree in this experiment is highly skewed, which makes a few filters very popular and leads to resolving many subscriptions at an early stage by adaptively prioritizing filters with high edge-coverage. Fig. 4.11 shows the same experiment under a uniform distribution where we eliminate skew by randomly assigning every filter a weight between [0,1] and assigning filters to subscriptions based on a weighted probability. Results show a noticeable increase in the processing delay cost of all approaches because the adaptive ordering of filters requires more filter executions to resolve all subscriptions when the popularity of filters is less skewed.

EMEC consistently outperforms all other approaches due to short-cutting the residual graph earlier by incorporating the extended edge cover into the calculation of a filter's

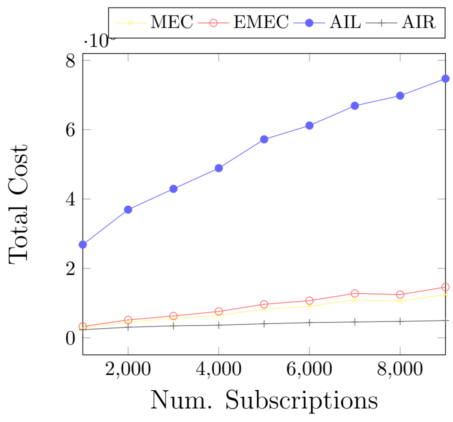


FIGURE 4.12: Total processing costs across the entire overlay - Redundancy Overhead

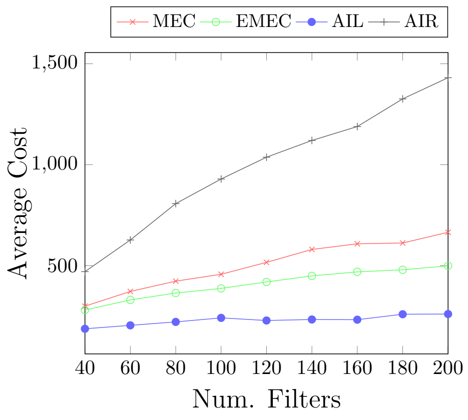


FIGURE 4.13: Impact of the total number of filters on the Processing Delay Cost

unit price. Furthermore, as the number of subscriptions increases, more possible combinations of the available filters is covered, resulting in an increase in the number of possible matches, as well as the chance of every filter getting evaluated as part of the adaptive ordering. This is in the benefit of our algorithms since finding a matching event as early as possible results in reducing the number of filters executed at brokers closer to publishers, and offloading filters to be executed downstream on the paths towards subscribers. This explains why the trends of MEC and EMEC are less steep than the baselines. Offloading more filters tapers down the number of filters executed on each path because as described earlier, as we go further in the tree, the popularity values of filters across subscriptions changes based on the view of each broker, which makes the algorithm take better decisions than upstream brokers. Ultimately, if filters were to be only executed at subscriber-edge brokers and ordered according to their local residual graph as in AIL, the results would be optimal with respect to the distribution of subscribers, but at the expense of (1) flooding the network with unmatched events, and (2) introducing a maximal redundancy overhead.

In the third experiment, we measure the redundancy overhead of all approaches as the total cost of filter executions across all broker in the overlay. This captures the additional processing cost imposed by two factors, (1) a poor ordering of filters and (2) executing the same filter at multiple brokers on different sub-trees. Results in Fig. 4.12 show how AIL leads to a linearly increasing total cost as the number of subscriptions increases. This is because sharing across subscriptions is only leveraged locally at each subscriber-end broker as opposed to sharing filters early on in the overlay. We can also see how AIR introduces the least overhead due to executing each filter only once at the publisher-end broker. Even though our algorithms can lead to some redundancy by executing the same

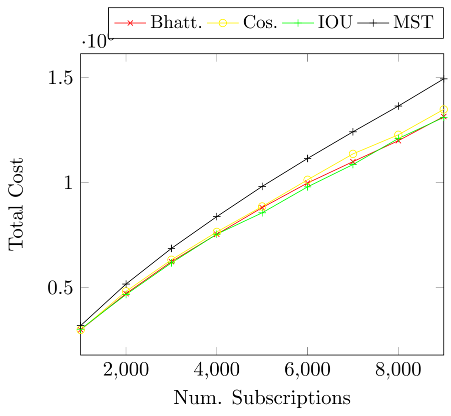


Figure 4.14: Topology reconfiguration under a Zipfian filter distribution

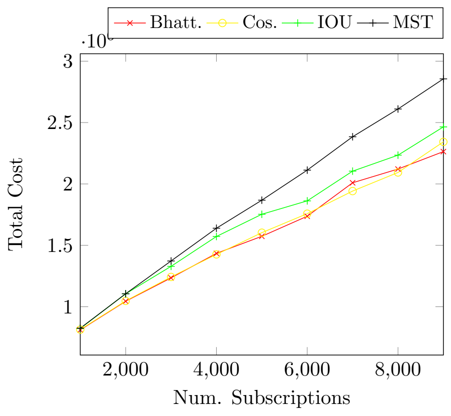


Figure 4.15: Topology reconfiguration under a Uniform filter distribution

filters at different paths due to offloading, this is balanced by decreasing the number of filters executed on each path, which explains the slight increase in processing cost as shown in Fig. 4.12. We also highlight how EMEC consistently leads to slightly more total cost than MEC because of offloading more filters to be executed downstream by eliminating more subscriptions earlier with the extended edge cover resulting in more redundant processing on separate paths.

4.5.2.2 Impact of Filters

In this experiment we fix the number of subscriptions to 3000 and vary the number of filters. We observe that the performance of AIR is more sensitive to the increase in the number of filters as compared to that of subscriptions. This is because in the latter, all algorithms use the same set of filters in the evaluation process. As the number of subscriptions increases, each filter will have a larger chance of being evaluated but the overall cost is bounded by the total cost of executing all filters available in the system. In Fig. 4.13 we show how the increase in the numbers of filters increases the processing delay cost of all approaches. However, it is also clear how the performance when applying MEC or EMEC degrades slower because distributing the execution of filters help reduce the impact of their increase as opposed to only resolving subscriptions at the publisher-end broker as in AIR.

4.5.2.3 Impact of Topology

In this experiment, we evaluate our overlay construction algorithm using the two proposed similarity measures based on the Bhattacharyya distance between filter popularity vectors (Bhatt.) and the cosine similarity between PRG impact vectors (Cos.). We also implement the Intersection over Unoin (IOU) similarity measure where we record matched events at every broker and measure the cardinality of their intersection squared divided by the cardinality of their union. Under a Zipfian distribution all similarity measure decrease the total processing cost almost equally as seen in 4.14. However, at a more balanced distribution of filters (Fig. 4.15) both our distance measures appear to outperform IOU which only utilizes traffic similarity as opposed to workload similarity.

4.5.3 Conclusion

In conclusion, experiments with varying toplogies, subscription workloads, and filter distributions, show how our Match-and-forward Edge-Coverage-based (MEC) algorithm outperforms both the All-in-Root (AIR) and the All-in-Leaves (AIL) baseline algorithms. In particular, intelligently distributing and ordering the execution of filters along various paths towards subscribers decreases the average Processing Delay Cost by 50% - 70% as compared to AIR (R1.1 & R1.2). The adaptive and greedy nature of the MEC and Extended-MEC heuristics helps tailor the ordering of filters on each path to the distribution of subscribers across brokers and the variations in filter distributions (i.e. popularity) across the subscriptions down each path. The reorganization of brokers in the overlay on the basis of PRG similarity between brokers shows a 20% decrease in the Total Processing Cost as compared to alternative approaches (e.g., shared event traffic IOU and static topologies) (R1.3.). The PRG similarity metric aims to capture shared filter executions between brokers along with the impact each shared filter has on the PRG of each broker in terms of expected edge coverage.

4.6 Summary

This chapter tackles research question (Q1.) concerned with supporting multimedia content as native event types in distributed publish/subscribe systems comprising federated broker overlays. It proposes the use of the publish/subscribe communication model as a medium for an asynchronous and decoupled interaction between multimedia producers and consumers. More precisely, It introduces fast binary classifiers (i.e. filters) into the forwarding function of brokers in an overlay network, it proposes an adaptive greedy algorithm inspired by previous approximation algorithms for the ordering and placement of filter evaluations across paths towards subscribers. The chapter also proposes two novel broker workload similarity metrics that facilitate the reconfiguration of the overlay topology subject to a skewed distribution of subscribers with similar interests. Experiments with varying publish/subscribe workloads show clear improvements in the average processing delay cost perceived by individual subscribers, a negligible increase in resource consumption across the overlay, and a low overall messaging overhead.

Chapter 5

SemanticPeer: A Distributional Semantic Peer-to-Peer Lookup Protocol for Large Content Spaces at Internet-scale

5.1 Introduction

This chapter tackles research question (Q2.) and proposes SemanticPeer, a DHT construction algorithm and lookup protocol that tackle the two technical requirements stated below.

Q2. How can loose semantic coupling be achieved within the exact-match lookup functionality offered by conventional DHTs underlying structured peer-to-peer networks while ensuring the dynamic organization, allocation, and retrieval, of data based on their shared semantic and syntactic properties.

Technical system requirements:

- **R2.1. Effective organization and mapping of semantic data.** This entails the distributed organization of data in a way that reflects the basic meaning of data items and the semantic relationships among them.
- **R2.2. Efficient and effective semantically decoupled lookup operations.** This entails departing from ID-centric file sharing to loose semantic coupling in DHTs. The retrieval of data items should be on the basis of a semantic routing

model that relaxes strict agreements on semantics and locates the target data items at a low cost.

This chapter is organised as follows. Section 5.2 provides an overview of SemanticPeer, it highlights the four main procedures that facilitate building a semantic DHT. Section 5.3 discusses related work. It elaborates on existing structured peer-to-peer networks and their underlying DHTs. It also discusses existing peer-to-peer publish/subscribe systems while emphasising on their matching model and degree of expressiveness. The section also discusses semantic overlay networks and categorises the main event matching models that exist in the literature, namely, the content-based approach, the concept-based approach, and the approximate semantic matching approach. Section 5.4 elaborates on the proposed solution. In particular, section 5.4.1 discusses the distributional hypothesis which constitutes the fundamental foundation for learning the semantic vector space underlying SemanticPeer. Section 5.4.2 explains the intuition behind using hierarchical clustering for partitioning the vector space and details the inner workings of the clustering algorithm. Section 5.4.3 discusses how the network is constructed via bootstrapping and how tight groupings of related semantic concepts are dynamically assigned to nodes in a structured peer-to-peer overlay. Section 5.4.4 discusses how a new node joining the network finds a point of entry and populates its own routing table. Section 5.4.5 presents the proposed semantically decoupled lookup protocol. Section 5.4.6 elaborates on how rendezvous routing can be utilised to map two semantically related terms to the same node atop SemanticPeer. Finally, section 5.5 provides an empirical evaluation of the different functionalities offered by the network and section 5.6 summaries the chapter and highlights the contributions of this work.

5.2 SemanticPeer Overview

SemanticPeer [172] is a DHT construction algorithm and lookup protocol that aims to elevate the tight semantic coupling between its participants. It takes a vector space consisting of N vectors, each representing a distinct semantic concept, and dynamically assigns groupings of vectors that fall in semantic proximity to each other to nodes in a structured peer-to-peer network (R2.1.). The proximity notion is defined according to a closeness metric (i.e. Cosine distance) that allows for performing lookup operations that converge to nodes maintaining the target or highly similar vectors (R2.2.). The intuition is that data items sharing semantic properties should be mapped to the same or neighbouring locations in the network. The overlay network is built on the basis of four main procedures as outlined below (Fig. 5.1):

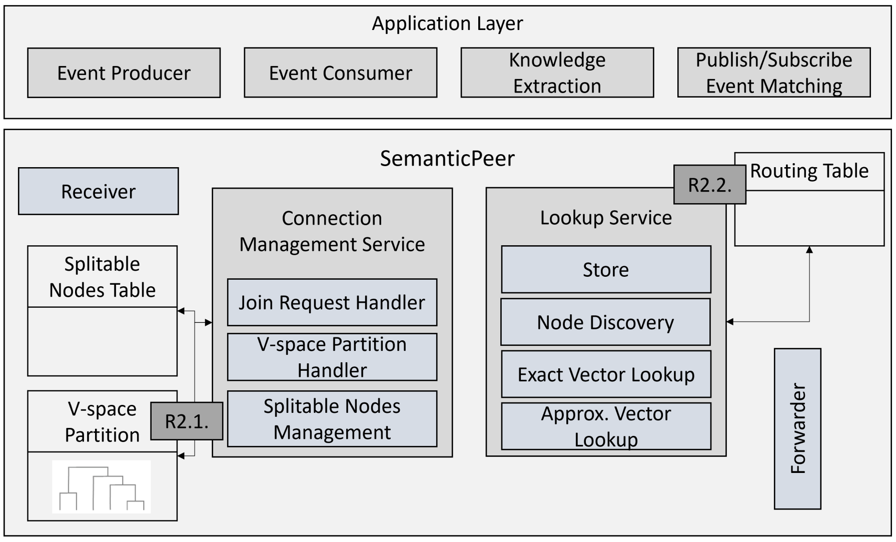


Figure 5.1: SemanticPeer Node Architecture

1. **Semantic agreement:** Participants (i.e. nodes) loosely agree on a large corpus of text comprising thousands of terms. The corpus is then mapped to a vector space by leveraging learning methods based on distributional semantics. In this work, we use pre-trained word embeddings obtained by training Word2Vec [158] on the GoogleNews corpus. Word2Vec is a 2-layer neural network that takes a large corpus of text as its input and produces a vector space. However, the methods introduced are generalizable in the sense that any corpus can be used and it can be paired with any learning model (e.g. BERT [173], Glove [174], etc.).

2. **Partitioning the space:** A vector space consisting of N vectors, each representing a distinct semantic concept, is partitioned via an unsupervised hierarchical clustering technique [160]. The result of the clustering is a dendrogram (tree-like) structure that links together the original vectors at multiple levels. The tree represents a taxonomy of clusters with parent-child and sibling relationships. Each cluster comprises a grouping of vectors that fall in semantic proximity to each other.

3. **Network construction (R2.1.):** The Connection Management Service is responsible for dynamically assigning the resulting partitions to nodes in a large structured peer-to-peer network. The network is constructed via bootstrapping, where, an entry node maintaining a partition of the space (i.e. sub-tree) passes along half of its local dendrogram to the new node and the two nodes exchange routing information. Two nodes are considered neighbours if they maintain partitions that share a sibling relationship in the original dendrogram.

4. **Content routing (R2.2.):** The Lookup Service provides low level communication abstractions by exposing a routing substrate that maps the vector representations of terms to a location in the space via an iterative lookup operation. The lookup operation leverages Cosine distance to map terms to nodes responsible for a partition of the space comprising terms that share semantic properties.

5.3 Related Work

This section is split into four segments. First I discuss existing structured peer-to-peer networks in terms of the properties and inner workings of their underlying DHTs. I then discuss existing decentralised publish/subscribe systems that operate atop structured peer-to-peer networks. Following that, I present approaches for constructing self-organizing semantic overlay networks based on structured and unstructured overlays. Finally, I categorise existing event matching approaches into three groups, namely, content-based, concept-based, and approximate semantic matching approaches, I highlight their shortcomings in light of the identified requirements.

5.3.1 Structured Peer-to-Peer Networks

Chord [117] is a distributed lookup protocol and algorithm for constructing, maintaining, and operating a DHT. Chord provides a distributed computation of a hash function mapping $m - bit$ keys to nodes using Consistent Hashing. Keys are ordered along an identifier circle modulo 2^m. Key k is assigned to the first node whose identifier is equal to or follows k in the identifiers circular space. To enable scalable key location, each node maintains a Finger Table. The ith entry in the table at node n contains the identity of the first node that succeeds n by at least 2^{i-1} on the identifier circle. This property ensures that by forwarding a key to the closest node at each step, the Chord protocol guarantees that the number of routing hops necessary will be $O(\log N)$.

Pastry [6] assigns each node a $128 - bit$ identifier within $[0, 2^{128} - 1]$. Node Ids and keys are thought of as sequences of digits with base 2^b. It organizes nodes in a ring structure so that a message can be routed from any node to any key in a logarithmic number of steps with respect to the total number of nodes. It implements a prefix-based routing protocol whereby a key is forwarded to the node whose identifier has the longest common prefix with the key's identifier. Each record in row i of the routing table of node n is a node identifier that shares i prefix bits with the identifier of n. Similar to Chord, the partitioning of the key space in Pastry along with each node's state guarantees reaching any key in $O(\log N)$ steps.

Kademlia [5] assigns each node a $160-bit$ opaque ID. It treats nodes as leaves to a binary tree, with each node's position determined by the shortest unique prefix of its ID. The protocol guarantees that each node knows at least one node in each of the sub-trees. Kademlia uses the XOR distance measure to identify how close two nodes are in the key space. To route messages in a logarithmic number of steps, for each $0 \leq i \leq 160$, each node maintains lists (i.e. k-buckets) of identifiers for nodes of distance 2^i and 2^{i+1} from itself.

CAN [7] maps keys as coordinates in a d-dimensional Cartesian Space. The space is dynamically partitioned among nodes such that every node is assigned a distinct zone. Each node maintains routing information to reach its immediate neighbors. Two nodes are considered neighbors if their coordinate zones overlap along $d-1$ dimensions and abut along one dimension. A node routes a message in a greedy manner by forwarding towards the neighbor with a zone closest to the destination based on Euclidean Distance. The average routing path that CAN guarantees is $\left(\frac{d}{4}\right)\left(n^{\frac{1}{d}}\right)$ hops.

Discussion: All the algorithms and protocols discussed above construct a structured peer-to-peer network from a purely logical key-space that bears no relation to any physical space. The space is partitioned arbitrarily where data items are mapped to partitions following a uniform random distribution. They all offer Key-based Routing primitives (KBRs) that route a message m with key x to the node responsible for the partition where x falls in a relatively small number of routing hops. The exact-match key or hash that is used for content-addressing has no meaningful relationship to the data it represents, it is merely a logical pointer. Attempting to match two semantically related terms would result in mapping their identifiers (i.e. keys) to two distant locations in the network. Conversely, SemanticPeer utilises a semantic vector space as its underlying key-space. The space can be conveniently used to map lexical items as a function of their distribution in linguistic contexts **(R2.1.)**. By that, SemanticPeer preserves the semantic and lexical relationships between data items and ensures that with a certain probability, semantically related data items will be mapped to the same or neighbouring nodes in the network **(R2.2.)**.

5.3.2 Peer-to-Peer Publish/Subscribe Systems

Scribe [18] is one of the earliest Topic-based Publish/Subscribe systems built over the Pastry DHT. It employs Rendezvous routing by applying two data mapping functions to associate subscriptions and events to nodes in a structured peer-to-peer network. The intuition is that publishers and subscribers routing towards the same key (i.e. topic identifier) will meet at the same node in the network. Scribe takes the union of the paths traversed from subscribers towards the rendezvous node to maintain and persist a group spanning tree that is then utilized for event dissemination.

Hermes [17] is a content-based publish/subscribe system built over the Pastry DHT. It employs rendezvous routing to associate event-types and subscriptions to nodes in the network. Here, an advertisement message that usually expresses the schematic structure of produced events should meet user subscriptions at the same node in the network. At each node n_i, it associates a filter with each link to a neighbor node n_j. As a subscription s is being routed towards its rendezvous node, whenever it comes across a node n_i that has seen a matching advertisement, s is also routed towards n_j, where n_j is the neighbor of n_i that sent the matching advertisement.

Meghdoot [35] supports subscriptions with numerical range predicates. It was build over the CAN DHT leveraging its d-dimensional Cartesian space. Subscriptions are defined over a schema of n attributes as key-value pairs. The intuition here is that the content space defines a $2n$ dimensional Cartesian space where each attribute is associated with a domain range $[L_i, H_i]$. A subscription defining a range predicate l_i, h_i, $1 < i < n$ occupies a point in the $2n$ dimensional Cartesian space.

DHTStrings [175] and PastryStrings [118] are approaches that focus on supporting subscription predicates on string attribute types. Matching operations may include suffix (e.g. ABC*), prefix (e.g., *ABC), and containment (e.g. A*B). DHTStrings uses the hash function native to the underlying DHT to map an event attribute holding value (ABCD) to nodes in the network. To map subscription predicates, they use the same hash function h to map all possible prefixes (e.g. h(A), h(AB), h(ABC), and h(ABCD)). The approach for supporting suffix and containment follows the same methodology. PastryStrings on the other hand is more comprehensive in the sense that it supports both numerical and string attributes. The proposed model relies on an indexing structure that organizes nodes in a hierarchy (i.e. string trees), where the number of constructed trees is equal to the number of letters in the alphabet. The key-based routing primitives native to the underlying DHT are used to map event attributes to the root (i.e. Tnode) of its respective tree. Relevant subscriptions are stored at child nodes where events traverse down the hierarchy based on the sequence of letters being matched and on the basis of whether the matching is based on numerical ranges or on suffix, prefix, or containment.

Tariq et al. [176] propose a broker-less peer-to-peer content-based publish/subscribe system that focuses on satisfying the individual message delay requirements of subscribers in the presence of bandwidth constraints. The authors distinguish between peer-level and user-level subscriptions in an attempt to, (1) minimise resource usage by avoiding false positives, and (2) ensure scalability by balancing the contributions of peers according to their available resources. They propose the use spatial indexing, e.g. DZ expressions [177, 178], which play an important role in efficient multiway recursive division of N-dimensional space. The event space is divided into regular sub-spaces which serve as enclosing approximations for user-level subscriptions. The intuition is to coarsen peer-level subscriptions so that additional events can occupy a share of its bandwidth.

Discussion: Plenty of structured peer-to-peer publish/subscribe systems exist in the literature. I refer the reader to the work by Kermarrec et al. [31] for an extensive analysis of existing solutions. The survey paper neatly categorises solutions along three dimensions, namely, expressiveness, DHT dependability, and state. However, all the solutions discussed are designed to work over structured textual and numerical content under the assumption of term-level full agreement between participants. The underlying logical key-spaces and exact-match KBR primitives do not suffice to overcome the semantic boundary between multimedia event producers and consumers. In addition, the use of spatial indexing by dividing the event space into an N-dimensional space is only applicable under the assumption that every dimension in the space is known, e.g. structured content space, which is turn challenged by the unstructured nature of multimedia events.

5.3.3 Semantic Overlay Networks

Chand et al. [67] propose event routing strategies that spread messages within a community of peers that share similar interests. Peers organise in a hierarchy that defines neighborhood relationships based on interest intersection, containment, or equivalence relationships. For instance, the containment hierarchy tree is defined as follows: A peer P that registered subscription S is connected in the tree to a parent peer P' that registered subscription S' if S' is the subscription in the system closest to S according to the containment proximity metric.

Voulgaris et al. [66] propose Sub2Sub, a collaborative self-organizing publish/subscribe system deploying an unstructured overlay network. The network utilises an epidemic-based routing strategy to organise nodes by continuously exchanging subscription information and assessing their semantic similarity. The idea is that every subscription defines an N-dimensional space, subscriptions are partitioned into M disjoint hyperspaces, whereby, relevant events fall into one of these partitions. This aspect simplifies the dissemination of events where publishers only need to locate any one matching subscriber for its potential event(s), and deliver the event(s) to it. From that point on, dissemination is taken care of by the matching subscribers themselves.

Tang et al. [179] propose pSearch, a decentralized non-flooding peer-to-peer information retrieval system that distributes document indices through the peer-to-peer network based on document semantics generated by Latent Semantic Indexing (LSI). They define the notion of a semantic overlay as a logical network where contents are organized around their semantics such that the distance (i.e. routing hops) between two documents in the network should be proportional to their dissimilarity in semantics. They use the CAN DHT [7], which organises nodes over a d-dimensional Cartesian space, to map the vector representation of a document (generated by LSI) as the key to store the document index in the CAN DHT.

Li et al. [180] propose the Semantic Small World (SSW) overlay network that facilitates efficient semantic search over peer-to-peer networks. The overlay dynamically clusters peers with semantically similar data closer to each other and maps these clusters in a high-dimensional semantic space into a one-dimensional small world network. The intuition is that structured data objects, usually represented by a collection of attribute values, can be projected as a point in a multi-dimensional semantic space. They propose a dynamic dimension reduction method to construct a one-dimensional SSW that operates in a high dimensional semantic space.

Penzo et al. [181] propose methods to cluster peers that are semantically related into Semantic Overlay Networks (SONs), while maintaining a high degree of autonomy. Choice of neighbors is supported through the introduction of two kinds of selections, one based on a semantic similarity threshold (range-based selection) and the other one is based on a maximum number of neighbors (k-NN selection). Semantic similarity is defined according to the WordNet thesaurus [157], in particular, they quantify the distance between two semantic concepts by computing the depths of the concepts in the WordNet hypernymy hierarchy and the length of the path connecting them.

Qian et al. [53] propose JTangCSPS, a semantic broker overlay built on the basis of an Ontology Class Weighted Tree (OCWT). The tree is mapped to a 1-dimensional space where each class is assigned a unique key. An Ontology Routing Table stores the IDs of ontology classes in different *sets* that reflect the taxonomy of classes in the OCWT. The system provides logarithmic bounds for the performance of the proposed semantic routing model as a function of the network size.

Tariq et al. [182] propose a content-based peer-to-peer publish/subscribe systems that leverages distributed spectral cluster management to, (1) reduce the cost for event dissemination by clustering subscribers exploiting the similarity of events, and (2) preserve the expressiveness of the subscription language. To cluster similar peers into a single dissemination structure they propose not only using the overlap between subscriptions, but also take into account the number of events recently matched by the subscriptions.

They use the Jaccard index to capture overlapping events between peers. One the similarity graph is computed, the subscription clustering is performed by partitioning the similarity graph into k disjoint sub-graphs.

Discussion: The approaches discusses above put forward various definitions of semantic similarity between peers. The various similarity metrics used include the following: (1) intersection, containment, and equivalence for subscription predicates that specify relational operators over string and numerical value types [66, 67] **(Content-based approach)**, (2) associations between concepts based on semantic and syntactic relationships stored using conceptual semantic hierarchies (e.g. ontologies) [53, 181] **(Concept-based approach)**, and (3) mapping of terms and documents to high dimensional spaces on the basis of multi-attribute structured data objects [180] or by leveraging methods based on computational linguistics (e.g. vector space semantics) [179] **(Approximate semantic matching approach)**. In the following subsection, I elaborate on this categorization of matching approaches while highlighting their shortcomings in light of the identified requirements **(R2.1. & R2.2.)**.

5.3.4 Current Event Matching Approaches

This section discusses the three main event matching models that exist in the literature. (See table 5.1)

5.3.4.1 Content-based Approach

In this approach, participants are tightly coupled by the structure and semantics of the data being disseminated. The system performs exact string comparisons between event attributes, and subscription predicates. In broker-based environments, explicit configurations are required to setup the routing of events and subscriptions across the system (e.g. SIENA [33] and PADRES [69]). The closed nature of the event processing environment guarantees high effectiveness in terms of event matching accuracy. However, scaling the system to cope with higher numbers of data producers and consumers comes at a high cost and usually requires cloud deployments. On the other hand, structured peer-to-peer networks organize nodes in a specific logical structure such as a tree, a ring, or a multidimensional space. The targeted expressiveness of content-based publish/-subscribe systems defines the requirements placed on the underlying structure [31]. The expressiveness of the system is dictated by the properties of the underlying key-space and the employed data mapping function (i.e. hash function). Content-based systems usually define events as tuples of ordered attributes as key-value pairs. Quite conveniently, this structure resembles a multidimensional space with one dimension per attribute [35].

		R1. Organization of semantic concepts			R2. Semantically decoupled lookup operations		
		Method	Cost	Effectiveness	Method	Efficiency	Effectiveness
Content -based Matching	Hermes [17], Meghdoot [35], PastryString [6]	Not explicit	Defining a large number of rules	Low	Exact-match KBR primitives	Low	Medium
	Sub2Sub [66], Chand et al. [67]	Semantic peer clustering	Defining a large number of rules	Medium	KBR primitives / Gossiping	Medium	Medium
Concept -based Matching	Kem et al. [52]	Not explicit	Creating Mapping Tables	Low	KBR primitives & Mapping Tables	Low	Medium
	Penzo et al. [181]	Ontological peer clustering	Establishing agreements on ontologies	Depends on the domains and size of ontologies	Semantic Routing Indices (SRIs) [183]	Medium to High	Medium to High
	JtangCSPS [53]	Ontology Class Weighted Trees	Establish agreements on ontologies	Depends on the domain and size of ontologies	Ontology Routing Tables (ORTs)	Medium to High	Medium to High
Approximate Semantic Matching	pSearch [179]	Latent Semantic Indexing (LSI)	Establishing agreements on large corpora of texts	Medium to High	CAN KBR primitive	Medium to High	Medium to High
	SemanticPeer	Word2Vec Embeddings & Hierarchical Clustering	Establishing agreements on large corpora of texts	Medium to High	Iterative Dendrogram Traversal	Medium to High	Medium to High

TABLE 5.1: Current Approaches

To this end, the most fitting peer-to-peer infrastructure is the one based on a multidimensional spaces (i.e. Meghdoot [35] over CAN [7]). Moreover, a plethora of indexing structures have been proposed to support range queries and queries that define predicates on string attribute types such as prefix, suffix, and containment (i.e. DHTStrings [175] and PastryStrings [118]). Approaches such as Scribe [93] and Hermes [17] rely on creating and persisting dissemination structures that are derived from rendezvous routing strategies where event attributes and subscription predicates should satisfy an exact string match. Finally, content-based approaches that aim to cluster nodes based on their semantic similarity perform exact string comparisons between terms present in events and subscriptions or look for intersections between subscription predicates with constraints over numerical ranges [66, 67]. It is clear that under both environments, namely broker-based and peer-to-peer, the publish/subscribe system tends to be limited as far as scaling out to more heterogeneous environments (i.e. large semantic content spaces) due to the effort required to keep the agreement on shared schemata and semantics, and to develop a large number of rules [55].

5.3.4.2 Concept-based Approach

In this approach, participants may use different terms to describe the same concept given that both parties agree on a hierarchical structure that preserves semantic and lexical relationships between terms. This approach was originally employed in centralized publish/subscribe systems (i.e S-TOPSS [184] & G-TOPSS [72]). The proposed semantic matching model translates the content of incoming events to all possible synonyms and semantically related concepts before matching. In the context of peer-to-peer file sharing systems, Kementsietsidis et al. [52] propose the use of mapping tables to overcome the exact-match limitation of existing key-based routing primitives offered by various DHTs. At their simplest, mapping tables are populated by pairs of corresponding identifiers from two different sources. With such tables, when a participating peer attempts to retrieve a file called X, it first consults a shared or local mapping table to find the names of X in each of the target peers. This approach is complementary to peer-to-peer publish/subscribe systems and can serve as a rule-base to overcome the tight semantic coupling between participants. Mapping tables can also be replaced with conceptual semantic hierarchies which may serve as a much more dense data structure to store semantic information. However, as such structures grow in size, they can hinder the performance of the system by forcing peers to initiate many lookup operations **(R2.2. Efficiency)**. Other concept-based peer-to-peer approaches aim to organise nodes on the basis of a semantic space representing the various data items stored in the network [53, 181]. The notion of semantic association or similarity is derived from hierarchical semantic structures (i.e. ontologies). However, the conceptual model itself takes a lot of manual work and expert knowledge to build; the semantic coverage of the model in terms of the number of concepts and semantic relationships can be considered limited compared to the wide semantic space of recognition at the scale of human ability **(R2.2. Effectiveness)**.

5.3.4.3 Approximate Semantic Matching Approach

In this approach, participants may also use different terms to describe the same semantic concepts given that they establish a loose agreement on large corpora of texts. In the domain of Natural Language Processing (NLP), word meaning is mathematically modelled and learnt by mapping terms to high-dimensional vector spaces leveraging large amounts of contextual information presented in large corpora of texts [8]. The resulting mappings along with an efficient vector similarity metric (i.e. Cosine distance) have previously been employed to build a centralized approximate event processing model for addressing semantic coupling in event-based systems [55]. The authors propose an approximate event matching scheme that is rooted in uncertain schema matching where they create pairwise mappings between the vector representations of terms present in structured approximate events and subscriptions. In a peer-to-peer setting, relevant approaches tackle structured event types by mapping the attribute space of events to a multi-dimensional semantic space that is then used to organize peers in a way that reflects the semantic relationships between the data items stored in the network [180]. However, semantic similarity is defined over structured event types and is limited in terms of matching semantically related terms. An alternative approach for unstructured data items (e.g. documents) leverage vector space semantics and latent semantic indexing to map documents as indexes in a multi-dimensional semantic space [179]. However, this approach is limited to identifying semantic similarities between documents instead of lexemes or terms. The computational semantics method used (i.e. LSI) is outdated when compared with the recent neural network based approaches (e.g. Word2Vec [158] & Bert [173]).

5.4 SemanticPeer System Design

SemanticPeer is a DHT construction algorithm and lookup protocol that aims to overcome the tight semantic coupling between its participants. The system departs from ID-centric file sharing to loose semantic coupling in DHTs. This is facilitated via the construction of a structured peer-to-peer network that organizes and maps data in a way that reflects the basic meaning of data items and the semantic relationships among

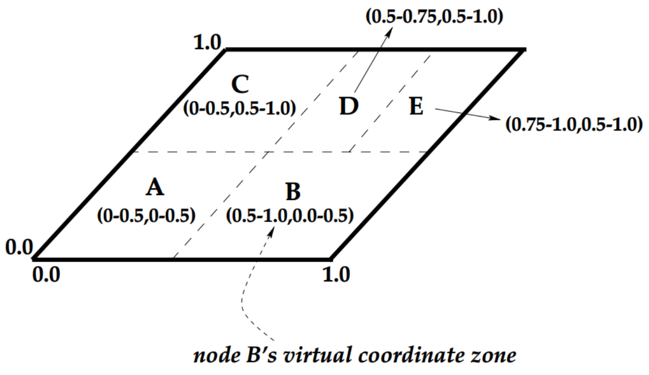


FIGURE 5.2: Example of a CAN 2-d space with 5 nodes

them. Storing and retrieval of data items is done on the basis of a semantic routing model that relaxes strict agreements on semantics between participants and locates the target nodes in a small number of routing hops.

The intuition behind our design was rooted in CAN [7]. As shown in Fig. 5.2, CAN is a DHT design that maps keys as coordinates in a virtual d-dimensional Cartesian space. The space is completely logical and bears no relation to any physical coordinate system. It is dynamically and arbitrarily partitioned among nodes such that every node is assigned a distinct zone. Two nodes are considered neighbors if their coordinate zones overlap along $d-1$ dimensions and abut along one dimension. If the space is partitioned into n equal zones, individual nodes maintain $2d$ neighbours.

In SemanticPeer, we replace CAN's Cartesian space with a high-dimensional semantic vector space. CAN's approach to overlay construction and content routing would not translate well to high-dimensional vector spaces due to scalability and space traversability constraints. On the one hand, as the value of d increases, the number of neighbours maintained by each node may increase to a degree that can hinder the performance of the system. On the other hand, the closeness metric utilised by CAN (i.e. Euclidean distance) is known to under-perform when applied to the retrieval of the most similar texts to a given document. Moreover, as the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse. This sparsity is problematic mainly because organizing and searching data often relies on detecting areas where objects form groups with similar properties; in high dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient.

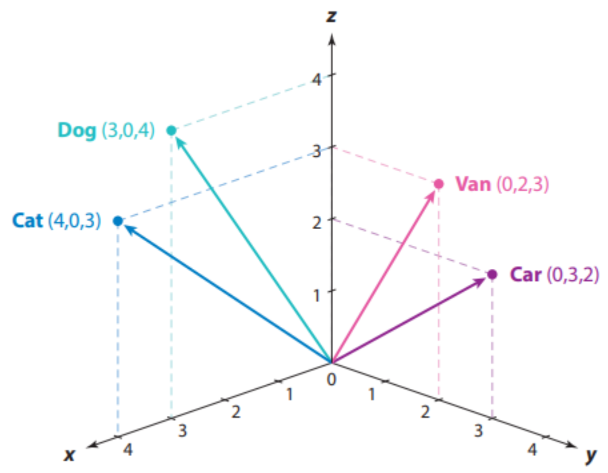


FIGURE 5.3: Distributional vectors of the lexemes car, cat, dog, and van. [8]

5.4.1 Distributional Hypothesis

The theoretical foundation of Distributional Semantics has become known as the Distributional Hypothesis: Lexemes with similar linguistic contexts have similar meanings [8]. Distributional approaches to word meaning acquisition originally leveraged statistical properties of linguistic entities as the building blocks of semantics. The hypothesis was premiered by Harris [185], who argued that "difference of meaning correlates with difference of distribution."

The aforementioned statistical properties initially identified as the frequencies of word co-occurrences in a corpus of documents have led to the conception of the vector space model in information retrieval [186]. The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering. Since its conception, the vector space model has also been used to identify semantically associated words by measuring the similarity of their corresponding vectors (Fig. 5.3). In particular, Cosine Similarity is a widely used measure of how similar two documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in multi-dimensional space. Cosine similarity is beneficial because even if the two similar documents are far apart by the Euclidean distance, chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

The availability of larger corpora and faster computers have recently resulted in the conception of highly accurate Machine Learning models based on Neural Networks such as Google's Word2Vec [187] and BERT [173], Stanfords's Glove [174], and Facebook's Fasttext [188]. Such models are used to produce word embeddings (i.e. vectors) by

Algorithm 1 Agglomerative Hierarchical Clustering

```
1: Given:
2: A set $V$ of vectors $\{v_1, ..., v_n\}$
3: A distance function $Dist(c_i, c_j)$
4: procedure AGGLOMERATIVE(V,DIST)
5:     $C \leftarrow \emptyset$                         ▷ Initialize empty forest of clusters
6:     for $i \leftarrow 1, n$ do
7:         $C \leftarrow C \cup \{\{v_i\}\}$
8:     $T \leftarrow C$                         ▷ Initialize Tree as sequences of Merges
9:     while $|T| > 1$ do
10:        $c_i, c_j = min(Dist(c_i, c_j))$
11:                $\forall c_i, c_j \in C$
12:        $C \leftarrow C \setminus c_i \setminus c_j$
13:        $C \leftarrow C \cup \{c_i \cup c_j\}$
14:        $T \leftarrow T \cup \{c_i \cup c_j\}$
15:    Return $T$
```

reconstructing linguistic contexts of words. The resulting vectors are typically comprised of several hundreds of dimensions. The produced semantic vector spaces render themselves a very promising candidate to serve as a key-space to construct semantically-aware DHTs. The challenge to be tackled is twofold. Mainly, the partitioning of the vector space is supposed to create compact semantic groupings of vectors as key-space partitions. In addition, each node in the network has to maintain enough state to be able to route a message m containing the vector v to the node responsible for vector v in a few number of hops relative to the network size.

5.4.2 Hierarchical Clustering

Hierarchical clustering [159] is an unsupervised clustering technique that aims to build a hierarchy of clusters (i.e. dendrogram structure). A dendrogram is a tree-like structure that visualises the relationship between data points. There are generally two ways to construct a dendrogram. The first method is bottom-up, where you start with the original observations and keep merging the individual data points and sub-clusters. The other alternative is top-down, where you start with the entire dataset as one cluster and divisively partition the space until you reach individual data points. The limitation with the latter is that you need to determine at each step how many clusters to create. Whereas, in the former, the branching reflects the distance between individual data points and formed clusters at dendrogram construction time.

In this work, we apply agglomerative clustering (i.e. bottom-up hierarchical clustering) based on Cosine distance to obtain tight groupings of word embeddings. In this technique, initially each data point (i.e. vector) is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster is formed. The basic clustering algorithm is straight forward (Algorithm 2). We compute the proximity matrix based on Cosine distance, where, for a vector space containing N vectors, the proximity matrix is an $N * N$ symmetric matrix with the ith and jth entry representing the cosine distance between the ith and jth vector. The algorithm begins with a forest of clusters, i.e. original vectors that are yet to be used in the hierarchy being formed. When two clusters c_i and c_j are combined into a single cluster $\{c_i \cup c_j\}$, c_i and c_j are removed from the forest, and $\{c_i \cup c_j\}$ is added to the forest. When only one cluster remains, the algorithm stops, and this cluster becomes the root. We adopt the group-average method [160] to calculate the distance between two newly formed clusters u and v (Equation 5.1). For all points $i \in u$ and $j \in v$ where $|u|$ and $|v|$ are the cardinalities of clusters u and v, respectively, the distance between u and v, $d(u, v)$ is computed as follows.

$$d(u,v) = \sum_{ij} \frac{d(u[i], v[j])}{(|u| * |v|)} \tag{5.1}$$

As shown in Fig. 5.4 (left), The output of the clustering is a dendrogram structure with the final single cluster as the root cluster and the two sub-clusters that were merged to form the super-cluster as its children, and so on. The leaves of the tree are the original vectors.

The reason behind choosing a hierarchical clustering approach as opposed to other popular clustering techniques is twofold:

- **Number of clusters:** Hierarchical clustering offers a way of not having to declare the number of clusters needed beforehand. This property is desirable because it is challenging to provide an accurate and up-to-date estimation of the number of nodes participating in a given peer-to-peer network [**? ?**].
- **Space traversability:** The resulting tree-like structure provides a taxonomy of clusters with parent-child and sibling relationships. The dendrogram structure allows us to successively partition the space into sub-trees that are addressable using Cosine distance, in a similar fashion to Kademlia's binary tree [5] which is in turn paired with the XOR distance metric.

5.4.3 Network Construction

To construct the network, we start with one node (i.e. bootstrap). The first node is passed the entire vector space after clustering as a dendrogram structure. When a node

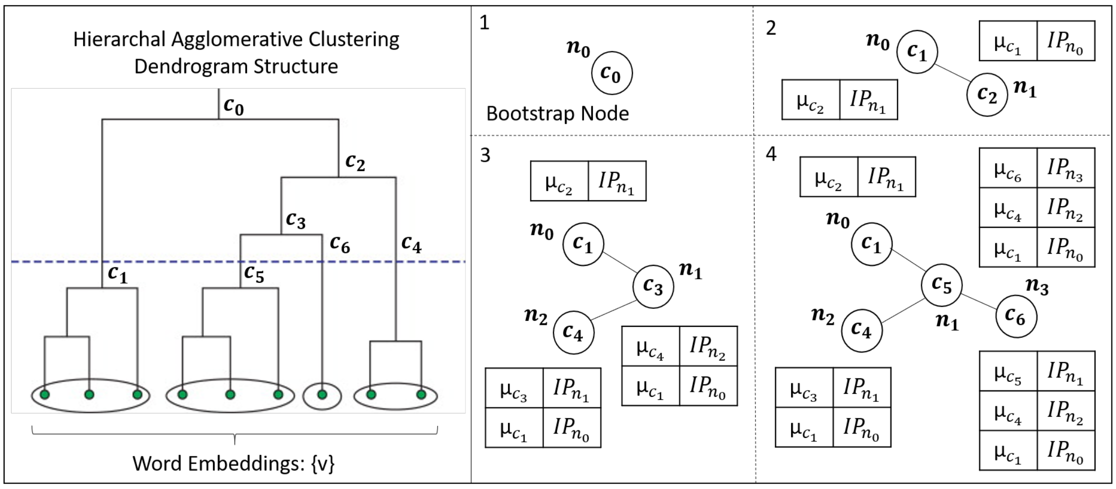


FIGURE 5.4: Dendrogram structure resulting from Hierarchical clustering (left). Network construction example (right).

attempts to join, the bootstrap node randomly chooses a node from the network for the new node to join through. Initially, no other nodes will be present. The bootstrap proceeds to split its vector-space partition with the new node by dividing the dendrogram it maintains into two clusters (i.e. sub-trees) and passing one cluster to the new node. The bootstrap node keeps reference to the mean vector of all the vectors that fall in the cluster it passed to the new node along with its IP address as a new entry in its routing table. Similarly, the new node keeps reference to the mean vector of the vectors that lie in the other cluster along with the IP Address of the entry node. The mean vector of the cluster maintained by a node in the network serves as its identifier. In the case of Word2Vec embeddings, a vector happens to be of magnitude 300. Here we note that the magnitude of vectors is subject to the method used for learning. As more nodes attempt to join, the bootstrap either selects itself, or some other random node from the network to split its partition with the new node. Once a node n is chosen, it splits is local cluster into its children clusters, and proceeds to exchange references with the new node, it also passes along all the references (i.e. Mean Vector and IP Address pairs) in its local routing table. The exchange of routing information guarantees that the joining node knows at least one node it each of its sub-trees. As the network grows, the original dendrogram will continue to be successively split into its subsequent sub-trees which serve as stand-alone dendrograms at the nodes that maintain them.

Fig. 5.4 shows an example of the construction algorithm given the dendrogram on the left. Node n_0 serves as the bootstrap and is initially passed c_0 representing the root cluster. When n_1 attempts to join, n_0 splits c_0 into its two children clusters c_1 and c_2. The two nodes exchange the mean vectors of both clusters to populate their routing tables. As shown in the dendrogram, c_1 is considered to be a compact cluster that should no longer be split. When node n_2 attempts to join by sending a Join message to the bootstrap node, n_0 defers the message to n_1. Node n_1 proceeds to split the sub-tree it maintains into c_3 and c_4. The two nodes exchange the mean vectors of both clusters and n_1 also forwards the mean vector of cluster c_1 along with the IP address of node n_0, so that node n_2 knows at least one node in each of its sub-trees. Similar to c_1, c_4 is no longer splittable, hence, when node n_3 attempts to join, the Join message is first deferred to n_1. Node n_1 then checks whether any of its subsequent neighbours can further split their partition. No other nodes can further split, so n_1 proceeds to split c_3 into its children clusters c_5 and c_6, passes along c_6 to n_3 and retains c_5. The two nodes exchange references so that n_3 knows the mean vectors and IP addresses of (c_1, n_0), (c_5, n_1), and (c_4, n_2) respectively. At this point, all four nodes in the network can no longer split their partitions making the network saturated. As it is clear, the potential size the network is subject to the initial size of the vector space, and the maximum number of flat clusters that can be generated from the given dendrogram. Furthermore, note that when n_1 split c_2 into c_3 and c_4, the node identifier of n_1 changed from the mean vector of c_2 to the mean vector of c_3. However, n_0 was not informed of that change, nor was it passed the mean vector of c_4 which represents the identifier of n_2. The intuition to keep n_1 addressable by the mean vector of c_2 from the perspective of n_0 is the following: If n_0 wants to address a vector v that is stored in n_2, the closeness metric should identify that the closest partition to v is c_2 which leads to forwarding the message to n_1. On reception of the message, n_1 had already split c_2. If the vector belongs to c_4, n_1 will identify that the vector is closer to the mean vector of c_4 than it is to the mean vector of its current local partition c_3, hence, n_1 should then forward v to n_2. This process mimics the traversal of the original dendrogram but in a distributed fashion. Also, after c_3 was split by n_1 into c_5 and c_6, n_0 and n_2 where not informed about the new node n_3. Not informing parent nodes about new partitions of the dendrogram guarantees that the routing table does not grow beyond $log(N)$ where N is the number of nodes in the network.

In the following we fill in the details on how the bootstrap chooses other nodes from the network to act as entry nodes. We define the notion of splittable nodes and how nodes keep track of them. We then provide a lookup protocol that allows any node in the network, given a vector v, to locate the node retaining the partition where v belongs in a logarithmic number of hops.

5.4.4 Finding an Entry Node

Each node maintains a table to keep track of what neighbors are still able to split their vector-space partition. A neighbor is removed from the table when it satisfies two conditions: On the one hand, the partition maintained by the neighbour (i.e. cluster)

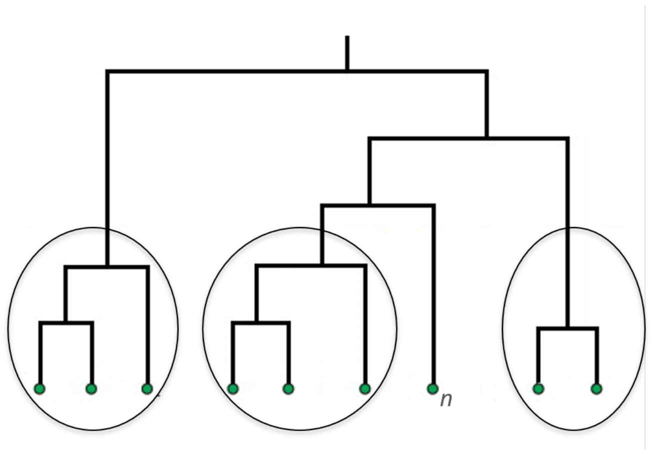


FIGURE 5.5: Node n maintains a partition of the vector space containing one original observation (i.e. vector), the ovals show sub-trees in which node n should know at least one contact

should be considered compact enough to not be further split into its subsequent child clusters. On the other hand, the neighbour should have no subsequent neighbors that can further split their vector-space partition. Keeping track of splittable neighbors allows any node to defer a Join message to a random node from its neighbors. Upon reception of a Join message, the bootstrap node either chooses itself as the entry node or forwards the Join message to one of its splittable neighbors. Under the latter case, once the neighbour receives the Join message, it either chooses itself or proceeds to recursively forward the message to one of its splittable neighbours until an entry node is chosen.

To keep the table of splittable neighbours up-to-date, once a node n splits its partition with a new joining node n', it checks if the partition it just passed to n' can further be split. If so, it adds the IP address of n' as an entry in its local splittable nodes table. Otherwise, n should check whether the partition it was left with could further be split or is compact enough. If its compact enough, the node then consults its splittable nodes table to check whether any of the other neighbouring nodes maintain splittable partitions. If the table is empty, n notifies its parent node that no nodes down this path can further be split. The parent node proceeds to remove n from its local splittable nodes table and repeats the same process by checking whether its local partition can further be split and whether the local splittable nodes table is empty. In which case, the parent node back-propagates the message to its parent node, and so on. Following the example in Fig. 5.4, when n_1 joins, n_0 identifies that c_2 is still splittable, so it adds n_1 to the splittable nodes table. Once n_2 attempts to join, n_0 identifies that its local partition c_1 is no longer splittable, so it consults its local splittable nodes table, and forwards the Join message to n_1. Following that, when n_1 splits c_2 into c_3 and c_4, it identifies that c_4 is no longer splittable, hence, it does not update its local splittable nodes table. Finally, when n_3 joins through n_1, n_1 identifies that neither the partition

Lookup Operation	Procedure
Store	This operation halts once the K closest Nodes identified at each iteration converge to the same nodes. In other words, the K node IDs identified at the current iteration should match the Node IDs stored in the local buffer from the previous iteration. The Node then instructs each of the K nodes to store a {vector:value} pair for later retrieval.
Node Discovery	This operation halts once the node with an identifier that matches the Lookup vector is identified.
Exact Vector Lookup	This operation halts once the recipient of an Exact Vector Lookup message identifies that it stores a data item with the same identifier as v.
Approximate Vector Lookup	Similar to Store, This operation halts once the K closest Nodes identified at each iteration converge to the same nodes. The initiating node then sends a fetch vector message to each of the K nodes identified. The recipients of the fetch vector message return the Top-K closest values they store by measuring the Cosine distance between their identifiers and v.

Table 5.2: SemanticPeer Protocol Operations

it passed i.e. c_6, nor the partition it retained i.e. c_5 are splittable. At this point, n_1 has no splittable neighbours and its local partition is no longer splittable so it informs its parent node n_0 which in turn removes it from its local splittable nodes table.

Finally, to determine whether a partition is compact enough, we use the Group-average distance method. The Group-average distance between two clusters is calculated according to Equation 5.1. This method forms a flat cluster u so that the original observations between its two child clusters have no greater a group-average distance than a predefined threshold t. More precisely, when a node passes along a partition u and retains v, it computes the group-average distance d between the child clusters of u. If $d < t$, u is deemed no longer splittable. It then calculates the group-average distance d' between the child clusters of v to identify whether the partition it retained is further splittable or not.

5.4.5 SemanticPeer Protocol

The SemanticPeer protocol can be split into four main operations: Store, Node Discovery, Exact Vector Lookup, and Approximate Vector Lookup. All four operations are derived from a single Lookup function that initiates a Lookup operation once passed a vector and halts once one or more target nodes have been identified.

The Lookup function takes a vector v of magnitude 300 along with a unique vector identifier as its arguments and identifies the K closest nodes available in the local routing table. The routing table stores contact information as a list of Node identifiers i.e. the mean vectors of clusters maintained by contacts, paired with their IP addresses. To identify the K closest contacts, the node initiating the Lookup simply calculates the Cosine distances between v and the Node identifiers, then ranks the results in ascending order. The node proceeds to send a Store, Node Discovery, Exact Lookup, or Approximate Lookup message to the K nodes identified in the previous step. K is a constant that determines the size of the search space and can impact the performance and accuracy of the Lookup operation. The recipients of any of the four messages return the IP addresses, and Node identifiers of the K closest nodes available in their routing tables, along with the identifier of v. Once the initiating node receives K replies that match the identifier of v, it removes any duplicates from the returned node identifiers, identifies the K closest Nodes to v, then stores them in a temporary buffer, and repeats the same process until the target nodes are identified. The conditions upon which each of the four Lookup operations halts and the subsequent actions done by the node initiating the Lookup operation are outlined in Table 5.2.

5.4.6 Rendezvous Routing atop SemanticPeer

At this point, we can easily see how a peer-to-peer publish/subscribe system with loose semantic coupling can be achieved atop SemanticPeer. Just like Scribe [18], we employ rendezvous routing. The advantage here is that the hash or key that was previously used to reach the rendezvous node maintaining a certain topic can now be replaced with a vector (i.e. word embedding) and routed towards a node that maintains a compact grouping of vectors that are semantically related. So, instead of being limited to an exact string match, users can subscribe to semantic concepts by using related terms. Publishers and subscribers can loosely agree on the corpus used to learn the vector space underlying the structured peer-to-peer network. Fig. 5.6 shows a subscriber interested in receiving images of cats. Publishers may call the underlying Store operation to route an event containing an image of a cat and labelled with the term "kitty" instead. The vector representation of the term "cat" would be routed towards target nodes that maintain related concepts (e.g. "kitty", "wild cat", etc.). Just like ranked data retrieval, when a subscriber routes a subscription containing the term "cat", the rendezvous node(s) may return the Top-K most related semantic concepts registered by publishers where subscribers may then select what topic(s) to subscribe to.

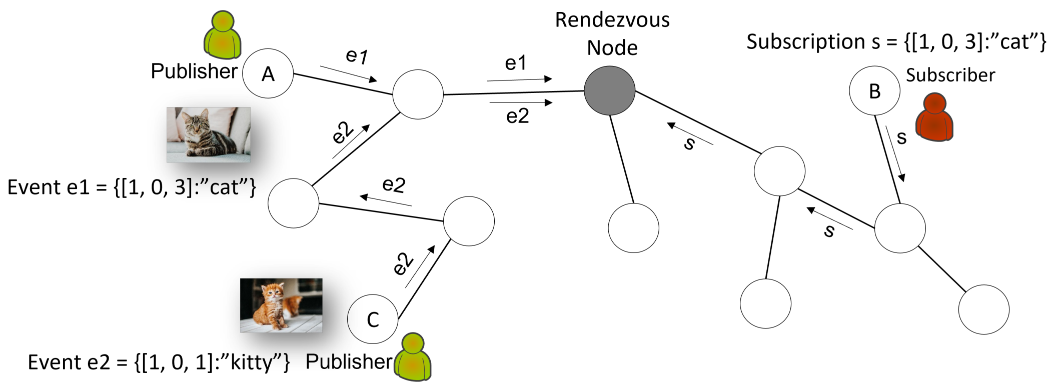

Rendezvous Node
Subscription s = {[1, 0, 3]:"cat"}
Publisher
A
e1
e1
e2
Subscriber
B
s
e2
s
Event e1 = {[1, 0, 3]:"cat"}
e2
s
e2
C
Event e2 = {[1, 0, 1]:"kitty"}
Publisher

FIGURE 5.6: Rendezvous Routing atop SemanticPeer. Participants may call the underlying Lookup Operations to route a value x into the network as a $\{vector : value\}$ pair.

5.4.7 Dendrogram Traversability Constraints

In agglomerative hierarchical clustering, during the construction of the dendrogram, we do not have any assumption on the number of clusters. After its construction, we may form flat clusters by slicing the dendrogram horizontally, whereby, all the resulting branches formed below the horizontal cut represent individual clusters. An alternative way is to cut the dendrogram inconsistently by utilizing some distance measure and a threshold to separate clusters individually. In SemanticPeer, the entire hierarchy of clusters is not known to all nodes due to scalability constraints, hence, the system uses the latter method to facilitate a distributed and dynamic formation of clusters. Every node successively divides the branch it maintains and passes along one of the resulting branches to the new joining node. If the node identifies that the branch it already maintains is compact enough and need not be further split, it ceases to act as an entry node and defers join messages to splittable neighbours.

Furthermore, all the members (i.e. vectors) of a branch belong to a certain class. However, to infer this class entity, one has to go through individual samples at each level within the formulated cluster and find out what feature is common in the resulting cluster. Such an approach is infeasible due the high-dimensionality of the space. Alternatively, we represent every class by the mean of all the vectors in its membership. The class identifier (i.e. mean vector) is then used to traverse the structure by assigning vectors to clusters based on Cosine distance. The closest class identifier to a vector v is regarded as the cluster (partition) it should be assigned to. Hence, a lookup operation for v iteratively finds closer class identifiers (acting as node identifiers) by consulting local routing tables at every node the lookup message is propagated to.

To populate its routing table, a node inherits all the references (node identifier and IP address pairs) from its parent node. These references include contacts that the parent node previously acquired by acting as an entry node (parent node) and dividing its own partition, in addition to contacts that it previously inherited from its parent node when it joined the network. Thus, the SemanticPeer protocol guarantees that every node knows at least one node in each of its adjacent branches (Fig. 5.5). With this guarantee, any node can locate any other node by its identifier with a certain probability. Unlike the Kademlia binary tree, the traversal of the hierarchy of clusters is probabilistic due to the following constraints.

- Hierarchical clustering is heavily driven by heuristics, which does not eliminate the possibility of forming sparse clusters or clusters that do not accurately reflect tight semantic groupings of terms. As a result, the traversal of the space becomes prone to arbitrary cluster assignments due to the greedy and iterative nature of the lookup protocol.
- The mean vector of all the member vectors in a cluster may not act as a precise representation of all the different terms it groups together. The mean here is merely a summarization of the vector representations of terms which would ultimately lead to loss of information. To mitigate this issue, one would incorporate additional parameters such as the numbers of data points in a cluster, their distribution, etc.
- The high dimensionality of the space coupled with an angular distance metric (i.e. cosine distance) opposes the unidirectionality of conventional distance measures (i.e. Euclidean distance over Chord's circular identifiers space or the XOR metric over Kademlia's binary tree). A unidirectional distance measure is where for any given point x and a distance $\delta > 0$, there is exactly one point y, such that $d(x, y) = \delta$. This property does not hold in SemanticPeer which might eventually lead to erroneous cluster assignments.

Nonetheless, under the assumption of an accurate assessment of the closest clusters at every iteration of the lookup protocol, and given that the original dendrogram is successively split in half during network construction, we can guarantee a logarithmic bound on the number of hops taken to reach a target node with respect to the number of nodes in the network.

5.5 Evaluation

In this section, we present our evaluation methodology and experimental results obtained from a prototype implementation of SemanticPeer. The prototype was implemented in Python mostly because of the support this language has for Machine Learning and Natural Language Processing with libraries such as Numpy, Scipy, and NLTK. Hierarchical agglomerative clustering has been implemented in standard numerical and statistical software such as R, MATLAB, Mathematica, and SciPy. The computational complexity of the algorithm arising from the need to continuously compute distances between data points has been reduced in various modern algorithms available in the Scipy Python library [160]. To perform experiments at the scale of thousands of nodes, we resort to an event-based peer-to-peer network simulation framework. We note that the size of the network is bounded by the size of the vocabulary (i.e. number of vector representations of terms used). We discuss how the simulation environment mimics the properties of a real world network environment. We also detail the methodology of our evaluation capturing evaluation metrics along two dimensions, namely, efficiency and effectiveness.

5.5.1 Simulation Environment

We use an event-based peer-to-peer network simulator implemented in Python using SimPy. SimPy is a process-based discrete-event simulation framework based on standard Python. The library is based on the concept of Generator Functions (GF). In summary, a GF returns a generator iterator by calling yield. The yield statement suspends the function's execution and sends a value back to the caller, but retains enough state to enable the function to resume where it left off. When resumed, the function continues execution immediately after the last yield run. This allows us to model nodes in a peer-to-peer communication network. We simulate bootstrapping and messaging, enabling network construction and content routing. The simulation framework is available on GitHub [189].

The simulated network comprises object instances from an OpenPubSub node class implemented in Python. If a node X wants to pass a message to node Y, it simply adds a data item, in the form of a JSON string, to a FIFO queue data structure maintained by node Y. The simulation is event-based, which essentially means that tasks executed at nodes, e.g., send, receive, etc., operate by taking events, one at a time, and propagating them throughout the system. Each operation is a process described by a Python Generator Function. The yield statement allows for the suspension of operations executed at nodes in response to event triggers. For instance, a send operation may wait for a *time-out* event representing delay. Events of this type are triggered after a certain amount of simulated time has passed. The suspension of operations via event triggers allows nodes in the overlay to execute operations asynchronously as in real-world implementations. Consequently, certain operations may be evaluated several times in a single cycle because of the different arrival times of the inputs and feedback events from downstream operations.

5.5.2 Evaluation Methodology and Metrics

Evaluation metrics fall into two categories: efficiency and effectiveness. Efficiency metrics measure the performance of the peer-to-peer network. This includes, network construction and routing overhead. On the other hand, effectiveness metrics measure the quality of event matching in a peer-to-peer publish/subscribe system. Such metrics capture the accuracy of our SemanticPeer Protocol including Store, Exact Vector Lookup, and Approximate Vector Lookup.

5.5.2.1 Evaluation Methodology of Efficiency

The evaluation methodology of efficiency is based on the methodologies of existing DHTs such as Chord, Pastry, CAN, and Kademlia. To construct the network, we used a high-dimensional vector space instead of conventional key-spaces. The performance of any network construction algorithm depends on the messaging overhead caused by individual node joins. We define the messaging overhead as the number of nodes traversed to find a random node in the network for the new node to join through. Moreover, the performance of any routing primitive depends on the length of the path between two arbitrary nodes in the network. We define the path length as the number of nodes traversed during a lookup operation.

5.5.2.2 Evaluation Methodology of Effectiveness

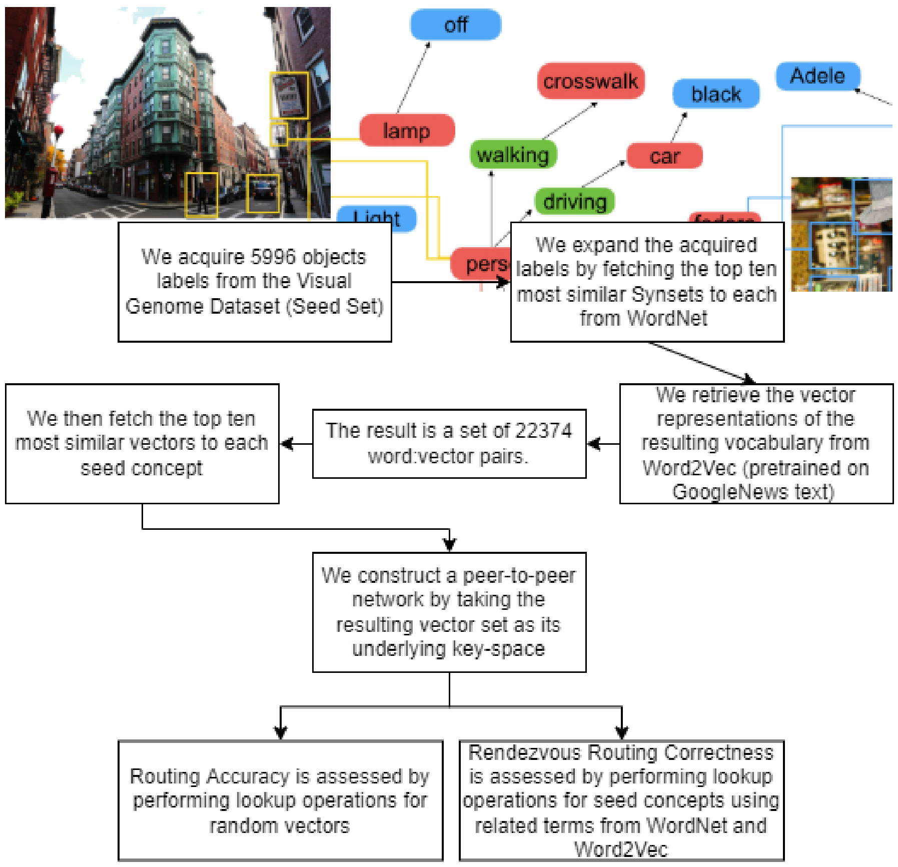
The evaluation methodology of effectiveness is based on Recall and Precision in the Information Retrieval community. In the context of Information Retrieval as manifested by publish/subscribe systems, we may have subscriptions and events as opposed to queries and documents. Matching a document to a query as reported in the human-labelled ground-truth set translates to matching a subscription to an event. We identify two important factors that decide the effectiveness of event matching over a DHT interface: Routing Accuracy and Rendezvous Lookup Correctness.

Routing Accuracy We can measure the accuracy of the routing algorithm given a known set of messages to be routed and the key-space partitions (i.e. nodes) they belong to. The set of pairs of messages (i.e. vectors) and their corresponding key-space partition ID is what we consider as our ground-truth set. It conveniently mimics a human-labelled set of documents and their associated single class assignments as in the standard text classification effectiveness evaluation.

Rendezvous Lookup Correctness The standard approach to matching in publish/subscribe systems atop a DHT interface is rendezvous routing. More precisely, for the system to undergo event matching, the event should meet the subscription at the same node in the network. The accuracy of this aspect of the system is what we refer to as rendezvous lookup correctness. In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top-K retrieved documents. For each such set, precision and recall values can be plotted to give a precision-recall curve. Precision means the percentage of results which are relevant. Whereas, recall refers to the percentage of total relevant results correctly returned by the Lookup function. In the context of Information Retrieval in a peer-to-peer network with an underlying high-dimensional vector space. The probabilistic nature of distributional models in vector space semantics result in probabilistic lookup operations. More precisely, a vector lookup operation targeting a rendezvous node may result in True Positives, False Positives, and False Negatives. A True Positive means that two related semantic concepts, as reported in the ground-truth set, actually coincide at the same node in the network. A False Negative on the contrary means that the two terms got routed to two distinct nodes in the network. Finally, a False Positive means that two terms that are not actually semantically related coincide at the same node and are returned as a match. We capture such measures by generating ground truth data comprising the correct results of lookup operations for a selected set of vectors (i.e. word embeddings).

5.5.3 Dataset Description

Visual Genome [138] is a dataset and knowledge-base for training Machine Learning models for various Computer Vision tasks (Fig. 5.7). The dataset connects image concepts to language by mapping object instances, attributes, and relationships to their equivalent WordNet Synsets (as shown in table 5.3). The dataset covers 5996 WordNet Synsets in total. Synset instances are the groupings of synonymous words that express the same concept. Each synset contains one or more lemmas, which represent a specific sense of a specific word.



We acquire 5996 objects labels from the Visual Genome Dataset (Seed Set)

We expand the acquired labels by fetching the top ten most similar Synsets to each from WordNet

We retrieve the vector representations of the resulting vocabulary from Word2Vec (pretrained on GoogleNews text)

The result is a set of 22374 word:vector pairs.

We then fetch the top ten most similar vectors to each seed concept

We construct a peer-to-peer network by taking the resulting vector set as its underlying key-space

Routing Accuracy is assessed by performing lookup operations for random vectors

Rendezvous Routing Correctness is assessed by performing lookup operations for seed concepts using related terms from WordNet and Word2Vec

FIGURE 5.7: Evaluation Workflow

Visual Genome Image Concept	WordNet Synset
"blue and white coat"	"coat.n.01"
"beautiful sight"	"sight.n.01"
"water is rough"	"water.n.01"
"moose"	"elk.n.01"

TABLE 5.3: Visual Genome - WordNet Mappings

Word	Most similar terms according to WordNet
"coat"	["coat", "dufflecoat", "furcoat", "overcoat", "topcoat", "jacket"]
"sight"	["sight", "percept", "perception", "field", "view", "aspect"]
"water"	["water", "tapwater", "bathwater", "groundwater", "springwater"]
"elk"	["elk", "moose", "deer", "fawn", "reddeer", "whitetaildeer"]

TABLE 5.4: Semantic Concept Expansion via WordNet

We expand the acquired Synsets by fetching the ten most similar Synsets to each based

Word	Most similar terms according to Word2Vec
"coat"	["jacket", "overcoat", "skiparka", "furcoat", "sweater", "trenchcoat"]
"sight"	["eyesight", "earshot", "eye", "nowhere", "spectacle", "stare"]
"water"	["sewage", "groundwater", "potable", "aquifer", "seawater"]
"elk"	["deer", "muledeer", "whitetaildeer", "antelope", "quail", "pheasant"]

TABLE 5.5: Semantic Concept Expansion via Word2Vec

on the shortest path that connects the senses in the is-a (hypernym/hypnoym) WordNet taxonomy. We unpack the fetched Synsets into their corresponding lemmas. We then apply a few pre-processing steps which include: removing punctuation, removing special characters (i.e. spaces, underscores, dashes, etc.), transforming all the words to lowercase, and removing all duplicates. A snippet of the resulting seed semantic concepts and their related terms according to WordNet is shown in table 5.4. We retrieve vector representations from Word2Vec by taking the intersection of the resulting set of extended semantic concepts and the Word2Vec vocabulary of 3 millions words acquired by training the Word2Vec model on GoogleNews Text. The result is a set of 22374 {word, vector} pairs. Finally, after retrieving the vector representations, for each seed semantic concept we determine the ten most similar concepts according to Word2Vec by taking the ten closest vectors to the vector representation of each seed concept using Cosine distance. A snippet of the result is shown in table 5.5.

In what follows, we construct a peer-to-peer network by taking the resulting vector space consisting of 22374 vectors as its underlying key-space. We cluster the vectors using the Agglomerative Hierarchical Clustering technique as discussed in section 5.4.2. We then use the SemanticPeer network construction algorithm and protocol to perform efficiency an effectiveness experiments. We show results that take into account three main parameters: The size of the network N, the group-average distance t used to identify splittable nodes, and the constant K used to configure the search space for Lookup operations.

5.5.4 Experimental Results

This section discusses the experimental results along two main dimensions, namely, efficiency and effectiveness.

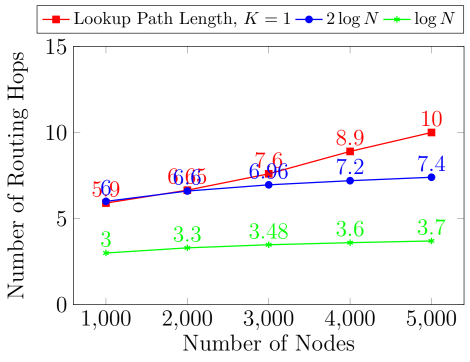


FIGURE 5.8: Average Lookup Path Length

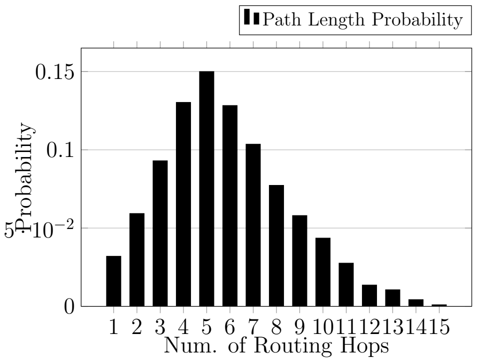


FIGURE 5.9: Average Path Length Probability Distribution

5.5.4.1 Network Performance

The first experiment shows the number of routing hops taken as a function of the network size. We vary the number of nodes from 1000 to 5000 in a network where each node maintains a vector-space partition. We carry out 3000 trials (i.e. lookup operations) where two arbitrary nodes are selected at random and a message is routed between the two. A lookup operation for vector v is initiated by one node and we measure the number of hops until it reaches the second node responsible for vector v.

Fig. 5.8 shows the average number of routing hops taken as a function of the network size. For reference we plot the values of $\log N$ and $2 \log N$ where N is the number of Nodes. This shows how SemanticPeer performs in comparison with Pastry, Chord and Kademlia. All three systems present mathematical proofs for Lookup operations taking no more than $\log N$ steps due to the deterministic nature of traversing their ordered circular identifier spaces and Binary tree respectively. The plot shows that the number of routing hops exceeds $2 \log N$ by an average of 1 to 3 Hops when the size of the network is between 3000 to 5000. We attribute the overall increase in routing hops to the Dendrogram structure. More precisely, network construction is based on partitioning the Dendrogram in half with no guarantees that the tree is balanced. The split may not result in partitions of the same size. Moreover, as more nodes join, we go further down the hierarchy which impacts the accuracy of its traversal given the high dimensionality of the space. In section 5.5.4.2 we show how the size of the network impacts the accuracy of the lookup. Erroneous lookup operations may result in longer paths. This aspect of the system can be mitigated by improving the clustering method, increasing the search space similar to Kademlia, and/or acquiring better vector representations.

Fig. 5.9 shows the probability distribution of the number of hops taken for a network of size 1000 in the same experiment. The results show that the path lengths with the

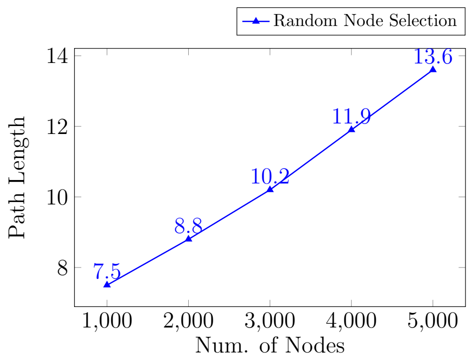


FIGURE 5.10: Finding Entry Node Path Length

highest probability for 3000 trails are 4, 5, and 6 which exceeds log N by a maximum of 3 hops.

In the second experiment, we measure the messaging overhead resulting from a node join operation. As mentioned earlier, upon reception of a join request, the bootstrap initiates a recursive random lookup for a node to act as an entry point for the joining node. Fig. 5.10 shows the average number of hops this operation takes as a function of the network size. The plot shows that the number of hops varies between ≈ 7 and ≈ 13 hops for a network of size 1000 to 5000 nodes respectively. The random nature of this process may result in longer paths when finding an entry node as the network grows. However, this process ensures that not only close nodes to the bootstrap are always chosen which in turn improves load-balance. Furthermore, by maintaining splittable nodes tables at nodes, we ensure that the entry node lookup operations don't go down the wrong path, and that they will always result in choosing an entry node unless all nodes in the network are saturated meaning that they reached a point where the partition they maintain can no longer be split.

5.5.4.2 Routing Accuracy

Routing Accuracy measures the effectiveness of the routing algorithm employed by SemanticPeer. The Publish/Subscribe implementation calls the native lookup function provided by the underlying DHT when attempting to route a subscription or an event to its corresponding rendezvous node. In the following, we evaluate the accuracy of Lookup operations. More precisely, routing a message m containing vector v should reach the node responsible for vector v. While excluding Churn and possible network partitions, DHTs such as Chord, Kademlia, or Pastry guarantee high-probability in routing a data item with an identifier k to the node responsible for the key-space partition where k falls.

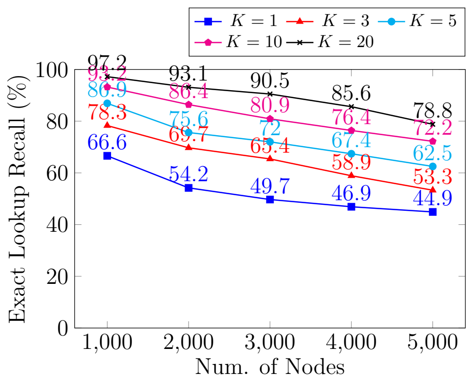


Figure 5.11: Routing accuracy with respect to the number of nodes

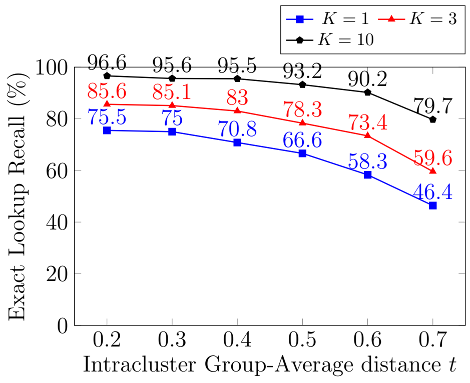


Figure 5.12: Routing accuracy with respect to the Group-Avg Distance t

In the context of lookup operations in SemanticPeer. Recall measures the proportion of correct Lookups to all the known correct Lookups in the ground-truth set such that $Recall = TP/(TP + FN)$. Furthermore, to generate the ground-truth data we select 3000 vectors at random from the existing vector space partitions. For every vector v_i used, we identify what node is responsible for v_i as part of its vector space partition. Then at random we select nodes to initiate Lookup operations for these vectors and compare the resulting destinations of the Lookup operations to the ground-truth data.

Fig. 5.11 shows the average *Recall* of 3000 Lookup operation trials as a function of the number of nodes. We vary the constant K which represents the number of the closest nodes chosen at each iteration of the lookup process. Recall varies between 66% and 45% when the number of nodes ranges from 1000 to 5000 where $K = 1$. However, as we increase the value of K we can see how the Lookup operation achieves 97% recall for 1000 nodes at $K = 20$. This comes at the expense of more messages sent which may reduce the efficiency but can be explained for networks with larger sizes where high effectiveness is a requirement. Moreover, Fig. 5.12 shows the effectiveness of the lookup operation while varying the threshold t chosen at network construction, where t represents the group-average distance between two child clusters which the node uses to determine whether it should further split its vector space partition. Varying this threshold can impact the quality of clusters. The default value chosen for all the previous experiments is 0.5. For a small value of t the number of Singleton clusters increases considerably which explains the high routing accuracy because locating a partition with fewer vectors by its mean vector would be much more accurate than larger partitions. On the other hand, for a high value of t the quality of generated clusters diminishes making the vector distribution a lot more sparse. Hence the accuracy of the lookup operation decreases because the mean vector doesn't offer a good representation of larger clusters with sparse vectors.

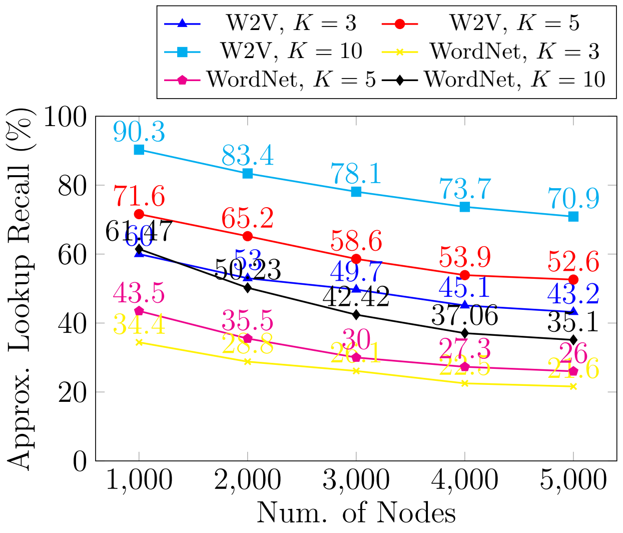


FIGURE 5.13: Rendezvous Lookup Correctness for related search terms from the WordNet taxonomy and Word2Vec embeddings

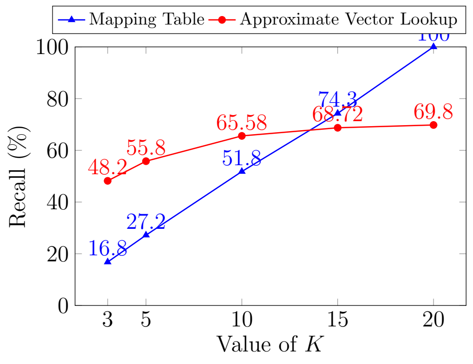


FIGURE 5.14: Semantic matching of related search terms via the native approximate lookup operations vs. the utilization of mapping tables for query rewriting

5.5.4.3 Rendezvous Lookup Correctness

In this experiment we measure the accuracy of approximately matching two related semantic concepts on top of SemanticPeer. We have two ground-truth sets for this experiment as discussed in section 5.5.3. We select 1000 random terms from the seed semantic concepts (i.e. Synset names) taken from the Visual Genome dataset and we consider them as events to be sent by publisher nodes. For each term we select 5 random related terms from each of the sets of related terms returned from Word2Vec and WordNet to act as subscriptions to be sent by subscriber nodes. For each of the 1000 events selected we end up with 10 approximate subscriptions. We then conduct two experiments, one for each ground-truth set.

The goal of choosing the related words according to Word2Vec is to measure the effectiveness of our vector space partitioning method. First, for each of the 5000 Word2Vec subscriptions we choose a random node from the network to send a subscription message by routing the vector representation of the term. The receiving nodes store the subscriptions in a subscription table. The constant K determines how many nodes the lookup function is looking for as target nodes, hence, for the value $K = 3$ the subscription gets stored in the three closest nodes to the vector representation of the semantic concept. Similarly, for each of the 1000 events, we choose a random node to publish the vector representation of the term to K nodes. Once a target node receives an event it matches it against registered subscriptions using Cosine distance and we compare the results with the ground-truth set. We then repeat the same experiment for the subscription acquired from WordNet. Fig. 5.13 shows the results of both experiments. As shown, SemanticPeer achieves 90% recall for a network of size 1000 where $K = 10$. Recall starts to decrease as the size of the network increases which can be attributed to the depth of the dendrogram as more partitions of successive sub-trees occur, hence, the routing getting less accurate. On the contrary, WordNet subscriptions don't perform as well. This can be attributed to the quality of the vectors and the training method. As can be seen in tables 5.4 and 5.5, the intersection of top related terms to the same semantic concept as reported by WordNet and Word2Vec is small. But, both sets can be considered relatively related to the semantic concept making the WordNet experiment very subjective.

Finally, the last experiment compares our approximate Lookup approach against using mapping tables as in conventional DHTs (Fig. 5.14). For this experiment we select 1000 seed semantic concepts at random and fetch the top 20 most related terms according to Word2Vec. For every seed semantic concept, we then select one of its 20 most related terms as an event to be sent by publisher nodes, and 5 other random terms as subscriptions to be sent by subscriber nodes. We vary the value of K between 3 an 20 for a network of size 1000 nodes. The Mapping Tables baseline takes the top K most related words and sends a Lookup message for each using conventional DHTs. We assume 100% lookup accuracy, hence, the goal of the baseline is to check whether any of the top K related words taken from the mapping table match the event. When K covers all the possible related words we should reach a matching accuracy of 100%. On the other end, the Approximate Lookup approach sends the vector representation of a subscription to the Top K closest nodes in the vector space. The results show that for a small value of K we outperform the baseline. Further, as the Mapping tables approach covers a bigger portion of the possible related terms for every seed concept, it starts to outperform the approximate Lookup Approach. However, the continuous increase in the value of K can hinder the performance of the system and lead to high messaging overhead.

5.5.5 Conclusion

In conclusion, we highlight that the proposed approximate semantic lookup protocol achieves over 97% in routing accuracy. This metric assesses the traversability of the dendrogram structure resulting from hierarchical agglomerative clustering and the viability of routing information acquired by nodes at network construction time **(R2.2.)**. However, this result comes at the expense of high messaging overhead due to the constant K used to parameterise the lookup operation. A high value of K leads to consulting up to K nodes for the Top-K closest node identifiers in their local routing tables at every iteration of the lookup operation, hence, more messages are propagated throughout the overlay network. In terms of matching related semantic concepts via rendezvous routing, the SemanticPeer protocol achieves over 90% accuracy in mapping the vector representations of two synonymous concepts to the same node in the network **(R2.1.)**.

5.6 Summary

This chapter presents a novel distributed lookup protocol and algorithm for approximate semantic content addressing in structured peer-to-peer networks based on DHTs. The proposed approach is based on distributional models of word meaning. Conventional keyspaces used to construct DHTs are replaced with a high-dimensional vector space that preserves the semantic properties of the data being mapped. We partition the space using agglomerative hierarchical clustering. This process is followed by a distributed algorithm that constructs a structured peer-to-peer network via bootstrapping. The overlay is augmented with an approximate semantic lookup protocol that, when passed a vector v, iteratively finds closer nodes by leveraging a proximity metric based on Cosine distance. Ultimately, it locates a target node that maintains a partition of the space containing data items that share semantic and syntactic properties. Experiments show comparable routing performance to existing DHTs in terms of the average path length for Lookup Operations. The proposed Lookup protocol achieves up to 97% in routing accuracy and up to 90% recall in matching two semantically related terms via rendezvous routing.

Chapter 6

OpenPubSub: Supporting Large Semantic Content Spaces in Peer-to-Peer Publish/Subscribe Systems

6.1 Introduction

This chapter tackles research question (Q3.) and proposes OpenPubSub, a decentralized content-based approximate semantic event matching model that addresses the technical requirements stated below.

Q3. How to support a conjunctive content-based subscription language over unstructured multimedia events spanning a large semantic content space in structured peer-to-peer networks?

Technical system requirements:

- **R3.1 Low cost in defining rules.** Low barrier to entry for participants in terms of establishing semantic agreements on terms present in exchanged events and subscriptions.
- **R3.2. Effectiveness in matching event and subscription semantics.** This entails the accurate assessment of the semantic relevance between events and subscriptions containing heterogeneous terms that might describe the same or highly related semantic concepts.

- **R3.3. Effectiveness in mapping event and subscription semantics.** This entails mapping arbitrary predicates and event attributes that share semantic properties to the same nodes in a large structured peer-to-peer overlay network.

This chapter is organized as follows. Section 6.2 provides an overview of OpenPubSub. Section 6.3 discusses related work. It discusses peer-to-peer publish/subscribe systems while highlighting the difference between having an underlying structured vs. an unstructured peer-to-peer overlay network infrastructure. It also discusses approaches on image and video analytics. Following that, section 6.4 presents the various services comprising an OpenPubSub node. It discusses how they interplay towards realizing the proposed decentralized approximate semantic event matching model. Section 6.4.1 presents the event and subscription language models. Section 6.4.2 presents the inner workings of the approximate semantic matching service. Sections 6.4.3 and 6.4.4 discuss the event management and subscription management services, respectively. Section 6.4.5 presents the member management service responsible for clustering rendezvous nodes with shared interests. Sections 6.4.6 and 6.4.7 discuss the connection management service responsible for building the network via bootstrapping and the Lookup service, respectively. Section 6.4.8 presents an improved hierarchical clustering algorithm based on shared nearest neighbours. The algorithm is utilized to partition the vector space underlying the overlay network. Following that, section 6.5 presents an empirical evaluation of OpenPubSub. And finally, section 6.6 summarises the chapter and highlights the contributions of this work.

6.2 OpenPubSub Overview

OpenPubSub is a layered, distributed, and approximate event matching model where participants loosely agree on a semantic space, i.e. finite set of terms, represented in the form of a large corpus of text **(R3.1.)**. The corpus is then mapped to a high-dimensional vector space by leveraging learning methods based on distributional semantics. The resulting vector representations are useful to quantify the semantic relatedness between terms. As shown in Fig. 6.1, this agreement is manifested by creating a structured peer-to-peer overlay network on the basis of a semantic vector space. Every node (i.e. participant) is assigned a vector as its unique identifier and maintains a partition of the space comprising the vector representations of terms that fall in semantic proximity to each other. The peer-to-peer infrastructure provides low level communication abstractions by exposing a routing substrate to the event processing layer to route events and subscriptions throughout the overlay. The underlying routing scheme maps the vector representations of terms present in events and subscriptions to a location in the space

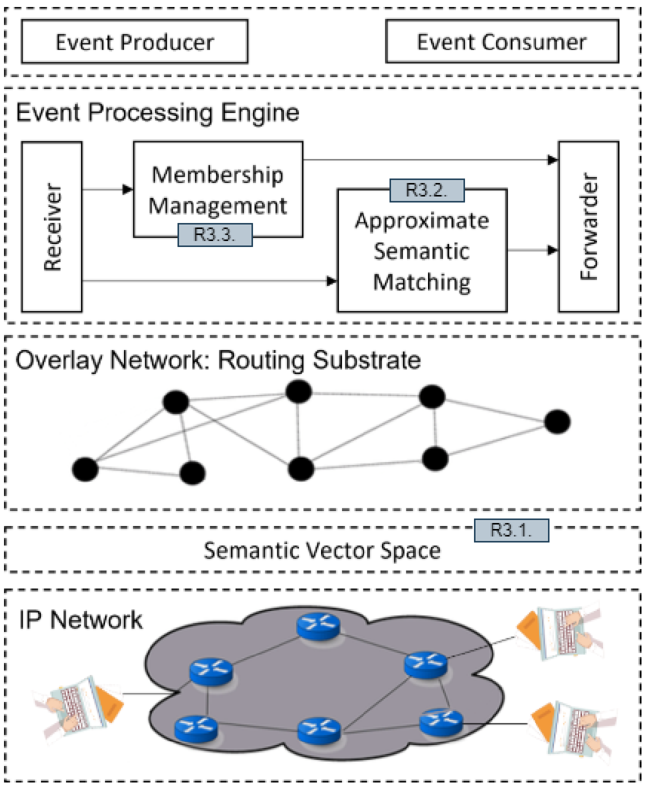


FIGURE 6.1: Layered Architecture of OpenPubSub's loosely semantically-coupled and decentralized event processing model

via an iterative lookup operation. The lookup operation leverages a proximity measure to map terms to nodes responsible for a partition of the space comprising terms that share semantic properties **(R3.3.)**.

The approximate matching scheme employs rendezvous routing by mapping conceptually related events and subscriptions to the same nodes in the network (i.e. rendezvous nodes). A membership management module comprising a gossiping service and an approximate subscription similarity estimation technique are used to cluster rendezvous nodes with shared interests **(R3.3.)**. Events are first mapped to their respective rendezvous nodes then propagated throughout a sub-overlay of nodes hosting similar subscriptions. This hybrid routing technique is followed by an approximate event matching model rooted in uncertain schema matching **(R3.2.)**.

6.3 Related Work

In the following, I discuss existing publish/subscribe systems over structured and unstructured peer-to-peer overlay networks. I also discuss related work on image understanding, action recognition, and accelerating image and video analytics in relational platforms (Table 6.1).

6.3.1 Peer-to-Peer Publish/Subscribe Systems (Structured)

Structured DHT-based peer-to-peer publish/subscribe systems employ rendezvous routing strategies to facilitate large-scale event matching and dissemination. I refer the reader to section 5.3.2 for a discussion on existing structured peer-to-peer publish/subscribe systems.

Discussion: The matching scheme in structured DHT-based solutions is dictated by the event schema and the underlying exact-match key-based routing primitives. In OpenPubSub, the event schema comprises a collection of terms of unknown length and no pre-defined attribute names. In a way, each event resembles a collection of values (topics) with no explicit semantic agreement between publishers and subscribers. The resulting semantic boundary and the lack of a well-defined fixed schema challenges conventional exact-match routing primitives and indexing structures. Instead, we resort to provisioning semantic routing **(R3.3.)** and matching strategies **(R3.2)** by replacing conventional logical key-spaces with a high-dimensional semantic vector space **(R3.1.)**. The system departs from exact string matching to approximate semantic matching by leveraging methods based on distributional semantics.

6.3.2 Peer-to-Peer Publish/Subscribe Systems (Unstructured)

Unstructured peer-to-peer publish/subscribe systems employ gossip-based routing strategies to facilitate large-scale event matching and dissemination. Gossip protocols rely on having all nodes in the network maintain contact information about a subset of all the other nodes. Such information is referred to as the partial view of a node and is used to disseminate messages throughout the system in an epidemic manner [92]. Popular research solutions, such as, Tera [94], Spidercast [125], Vitis [68], StAN [126], and Poldercast [19], propose topic-based event routing protocols over unstructured overlay networks. They use a combination of peer sampling and clustering. Peer sampling works by establishing random links between nodes through periodic randomized partial view exchanges. Peer clustering organizes nodes into topic-connected overlays by exploiting shared interests between nodes. The resulting structures vary based on preset requirements on node degrees, overlay diameter, absence or presence of churn, and the expected number of topics. The main goal behind peer selection is eliminating relay nodes which in turn reduces the messaging overhead.

Discussion: Nonetheless, proposed peer selection functions perform exact string comparisons between topics registered at nodes. Such an approach would not suffice when users use different terms to refer to the same semantic concepts. Conversely, in SemanticPeer [172], topics that share semantic properties are mapped to the same or already

		Content-based Matching Model		Concept-based Matching Model		Approximate Semantic Matching Model		
		Hermes [17], Meghdoot [35]	Sub2Sub [66], Chand et al. [67]	Mapping Tables [52]	S-ToPSS [56] G-ToPSS [72] JTangCSPS [53]	Hasan et al. [55]	SemanticPeer [172]	OpenPubSub
R3.1. Defining Rules	Method	Term-level full agreement	Term-level full agreement	Value Correspon -dance via Mapping Tables	Concept-level shared agreement via ontologies	Loose agreement on large Corpora of texts	Loose agreement on large Corpora of texts	Loose agreement on large Corpora of texts
	Cost	High	High	Medium	Medium	Low	Low	Low
R3.2. Semantic Matching	Method	Exact String Matching	Exact String Matching	Exact String Matching	Boolean Semantic Matching	Approx. Semantic Matching	Approx. Semantic Matching	Approx. Semantic Matching
	Effectiveness	Low	Low	Medium	Medium	Medium to High	Medium to High	Medium to High
R3.3. Semantic Routing	Method	Rendezvous Routing via exact-Match KBR primitives	Gossiping and Semantic Clustering	Query Rewriting followed by Rendezvous Routing	Rendezvous Routing via Ontology Class Weighted Trees (OCWTs)	NA	Rendezvous Routing via Approx. Semantic Lookup Operations	Rendezvous Routing and Gossiping
	Effectiveness	Low	Low	Medium	Medium	NA	Medium to High	Medium to High
	Efficiency	High	High	Medium to High	High	NA	Medium	Medium to High

Table 6.1: Current Event Matching Approaches

neighbouring nodes in the network **(R3.3.)**. However, the probabilistic nature of the proposed approximate lookup function leads to having some *stray* topics (semantic concepts) get mapped to distant nodes. To mitigate this issue, in OpenPubSub, we leverage peer clustering to link together rendezvous nodes that share similar subscriptions. We employ randomized peer sampling and peer clustering via a novel peer selection function that utilizes a subscription similarity estimation technique based on vector space semantics **(R3.3.)**.

6.3.3 Image and Video Analytics

I refer the reader to section 2.4.4 which discusses the field of Computer Vision and Image Understanding in addition to various Deep Neural Network architectures used for image classification and object detection tasks; and section 4.2.2 which discusses existing solutions that attempt to design light-weight and optimized neural network architectures to run on resource constrained devices.

Aside from still image classification and object detection tasks, there has been a number of approaches that build systems for video event processing, e.g., human action recognition, motion detection, or collision detection tasks. Yadav et al. [131] propose VID-WIN, a video complex event processing engine that utilizes an expressive video event query language, a Knowledge Graph based approach to represent video streams, and various query-aware window optimization techniques, to facilitate fast spatio-temporal event pattern matching. Gao et al. [45], propose an adaptive fusion and category-level dictionary learning model for multi-view human action recognition. The system processes video streams from multiple cameras to overcome viewpoint variations and to optimize the reconstruction of samples toward the action recognition task.

Discussion: We consider the works discussed above as complementary to OpenPubSub. Ambient Intelligent services in IoT smart environments comprise intelligent software agents that help orchestrate all aspects of an environment. Agents with image and video analytics capabilities may reside in resource constrained edge nodes or more resourceful fog or cloud nodes (see section 2.4.3). They cooperate and work together with a semantic publish/subscribe event processing layer to facilitate the dissemination of image and video event types to a plethora of users and applications.

6.4 OpenPubSub

This section presents the various services comprising an OpenPubSub node and how they interplay towards realizing its decentralized approximate semantic event matching

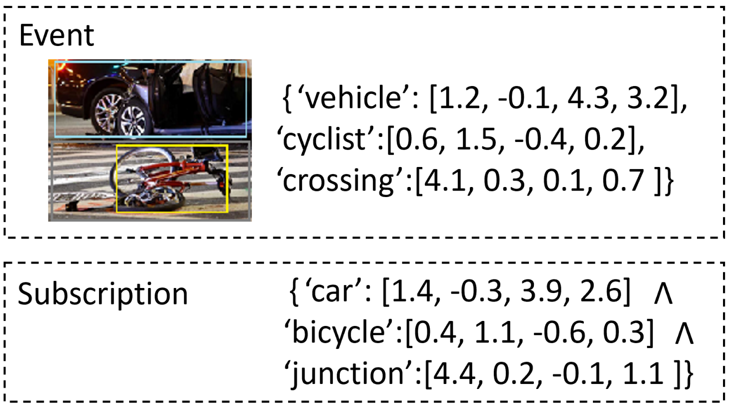

Event
{'vehicle': [1.2, -0.1, 4.3, 3.2],
'cyclist':[0.6, 1.5, -0.4, 0.2],
'crossing':[4.1, 0.3, 0.1, 0.7]}
Subscription
{'car': [1.4, -0.3, 3.9, 2.6] ∧
'bicycle':[0.4, 1.1, -0.6, 0.3] ∧
'junction':[4.4, 0.2, -0.1, 1.1]}

FIGURE 6.2: Illustrative example of an IoMT event and subscription describing an accident involving a cyclist and a car

model, it also delves into the implementation details of each of its components.

6.4.1 Event & Language Model

The event model used in this work evolves throughout a two stage process. Initially, an unstructured multimedia event such as a raw image or a video stream is received by an AI agent that employs a knowledge extraction task and adds descriptive labels prior to publishing. In the next stage, we have a semi-structured event consisting of the payload (i.e. unstructured multimedia content) and meta-data comprising a set of textual labels. Following that, the same agent, or another participating agent incorporates the vector representations of the present labels by transforming the meta-data into a set of tuples. Each tuple consists of a word:vector pair. A word:vector pair collectively represent a semantic concept that was extracted from the multimedia content being disseminated. On the other end, a subscription follows a conjunctive query model and comprises a set of predicates stringed together with the conjunction symbol '$\wedge$', each predicate also consists of a word:vector pair (Fig. 6.2).

The intuition behind adopting vector space semantics to represent terms present in events and subscriptions is to guarantee a low barrier to entry for event producers and consumers in heterogeneous environments such as the IoT/IoMT **(R3.1.)**. This model allows participants to loosely agree on large corpora of texts which may contain tens of thousands of terms.

6.4.2 Approximate Semantic Matching

Assuming that E is the set of all events, and S is the set of all subscriptions. W & V represent the sets of possible words that describe human-recognisable content, and

Algorithm 2 Approximate Semantic Matching

```
1: Given:
2: A set $V$ of vectors $\{v_1, ..., v_n\}$ representing a subscription
3: A set $V'$ of vectors $\{v'_1, ..., v'_m\}$ representing an event
4: procedure ApproxMatching(V,V')
5:     $A \leftarrow getMapping(V, V')$
6:     for $i \leftarrow 1, n$ do
7:         if $A_i < t$ then
8:             Return $False$
9:     Return $True$
```

Algorithm 3 *top* − 1 Mapping

```
1: Given:
2: A set $V$ of vectors $\{v_1, ..., v_n\}$
3: A set $V'$ of vectors $\{v'_1, ..., v'_m\}$
4: procedure getMapping(V,V')
5:     $M \leftarrow \emptyset$                          ▷ Initialize empty n*m matrix
6:     for $i \leftarrow 1, n$ do
7:         for $j \leftarrow 1, m$ do
8:             $M_{ij} \leftarrow CosineSimilarity(v_i, v'_j)$
9:     $A \leftarrow \{\emptyset\}$                     ▷ Initialize Mapping as an empty set
10:    for $i \leftarrow 1, n$ do
11:        $A_i \leftarrow argmax M_{ij}$
12:    Return $A$
```

their vector representations according to the distributional model used for learning, respectively. We define a function $\{F(w \rightarrow v) : w \in W \wedge v \in V\}$ that maps words to their vector representations, such that for all $w \in W \wedge v \in V$ there exists a unique mapping $w \rightarrow v$. For all $e \in E$ and $s \in S$, the same word:vector mapping in e or s cannot occur twice.

The inner workings of our matching scheme are inspired by the *top* − 1 approximate semantic matcher presented by Hasan et al. [55]. The goal of matching is to effectively assess the semantic relevance between a subscription $s \in S$ and an event $e \in E$ **(R3.2.)**. A relevance score is computed by obtaining pairwise mappings between the word:vector pairs in s and e. Given that the words on both ends are unlikely to result in an exact string match, the pairwise comparison is done by measuring the Cosine similarity of their vector representations resulting in a value between 0 and 1. The matching scheme works as follows (Algorithms 2 and 3): Initially an $n*m$ similarity matrix M is computed by measuring the semantic relatedness between each word-vector pair in s and all the word-vector pairs in e. Where n is the number of predicates in s and m is the number of tuples in e. Secondly, the most likely mapping is chosen by taking the tuple M_j with the *top* − 1 similarity measure for every predicate M_i. Finally, to reach a final outcome

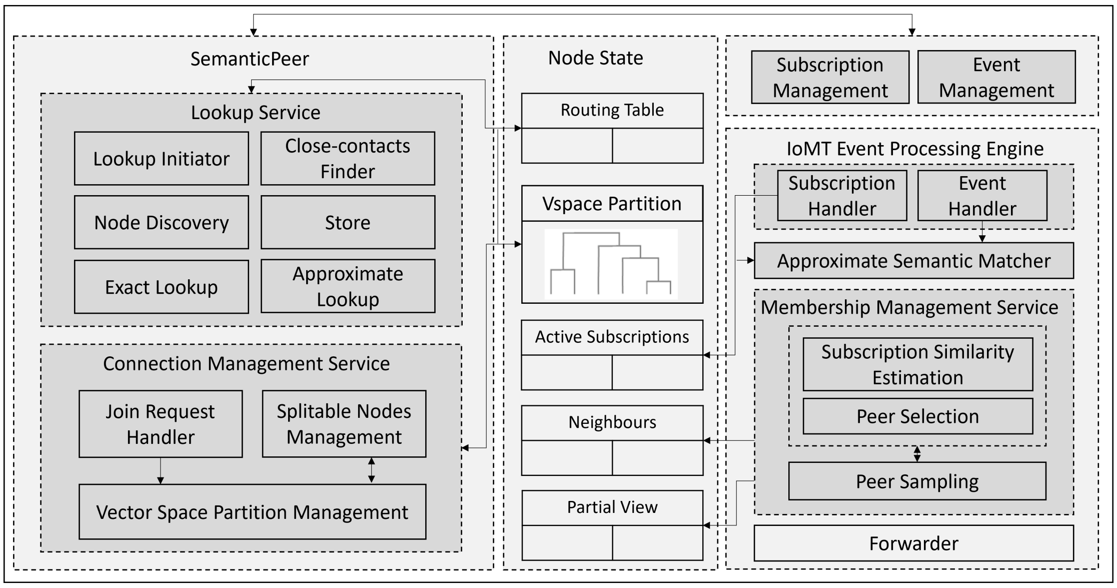


FIGURE 6.3: OpenPubSub Node Architecture

on whether the most likely mapping amounts to a match or not, all the final pairwise mappings chosen should have a similarity measure greater-than a preset threshold t.

6.4.3 Event Management

This component is responsible for disseminating events throughout the overlay network. It interacts with the underlying Lookup service at the network layer to map the word:vector pairs present in an event e to nodes in the network. For every tuple $w \to v$ in e, an approximate lookup operation is initiated. This operation recursively finds closer nodes to v and is parameterized by a constant k which allows for increasing the search space by choosing the $top - k$ closest nodes at every iteration. Once $l = k * m$ nodes are identified, where m is the number of tuples in e, the event management component calls a publish method to route e to each of the l nodes.

On the receiving end, an event handler is in charge of delegating e to the Approximate Event Matcher component which carries out the matching process against subscriptions registered in the active subscriptions table. It also forwards e to all its neighbours by consulting the local neighbour table which is in turn populated by the membership management service as we discuss in section 6.4.5.

6.4.4 Subscription Management

This component is responsible for registering subscriptions in a local subscriptions table and mapping them to rendezvous nodes in the network. The mapping process works in a similar fashion to the event management component by calling the underlying lookup service and performing an approximate lookup operation for every predicate $w \rightarrow v$ in a subscription s. It results in routing s to $l = k * n$ nodes where n is the number of predicates in s. On the receiving end, a subscription handler is in charge of registering s in the local active subscriptions table.

6.4.5 Membership Management Service

The goal of the Membership Management service is to aid with the propagation of an event e targeting rendezvous node r to additional nodes that hold active subscriptions similar to the ones registered at r **(R3.3.)**. The intuition is to support the selective yet probabilistic lookup operation with a randomized gossip-based technique that maximizes the probability of a match while keeping the constant k used to parameterize the lookup service to a minimum. Gossiping techniques have previously been used to minimize messaging overhead by exploiting shared interests in topic-based publish/subscribe systems atop unstructured overlay networks [19, 68, 94, 125–127]. Approaches dynamically construct topic-connected clusters of nodes by actively connecting nodes that subscribe to the same topics into relay-free sub-overlays. We repurpose random overlay construction and interest clustering techniques to work over a structured overlay due to the approximate nature of OpenPubSub.

This service is based on the work by Voulgaris et al. on Cyclon and Vicinity [54], it implements a two-layered framework comprising a peer sampling service at the bottom layer and a peer selection service at the top layer. The two services gossip periodically and asynchronously.

6.4.5.1 Peer Sampling

This service is in charge of periodically updating the partial view of a node via a basic swapping protocol [190]. Initially the partial view of a node n is a replica of its routing table which was populated upon joining the network by the Connection Management Service. The routing table stores references to partitions from a semantic vector space that are then assigned to nodes in the network. The peer sampling service starts by querying each of the nodes in its partial view and acquiring the vector identifiers of the subscriptions they have received from subscriber nodes. A vector identifier of a subscription s registered at rendezvous node r is simply the vector v used by the subscribing node to map s to r. Basic swapping simply works by exchanging a subset of node descriptors from a node's partial view with a random node within the same subset. A node

Algorithm 4 Peer Selection

```
1: Given:
2: Set P of Node descriptors (Partial View)
3: Set N of Node descriptors (Neighbour Table)
4: procedure SELECTNEIGHBOURS(P,N)
5:     $Q \leftarrow N.randomDescriptor()$
6:     $N.remove(Q)$
7:     $mergedView \leftarrow P \cup N \cup \{self\}$
8:     $QcandNeighbs \leftarrow SubSimEst(Q, mergedView)$
9:     $send(QcandNeighbs, Q)$
10:    $receive(candNeighbs, Q)$
11:    $mergedView \leftarrow P \cup N \cup candNeighbs$
12:    $N \leftarrow SubSimEst(self, mergedView)$
13:    Return
```

Algorithm 5 Subscription Similarity Estimation

```
1: Given:
2: Node descriptor Q (Target Node)
3: Set N of Node descriptors (Merged View)
4: procedure SUBSIMEST(Q,N)
5:     $S = \emptyset$                        ▷ Empty set of peer similarity scores
6:     $CN = \emptyset$                       ▷ Empty set of candidate neighbours
7:     $V \leftarrow Q.subVecIds$
8:     for $N_i \in N$ do
9:         $V' \leftarrow N_i.subVecIds$
10:        $A \leftarrow getmapping(V, V')$
11:        $card = 0$                          ▷ Cardinality of intersection
12:        for $A_i \in A$ do
13:            if $A_i \geq t$ then
14:                $card = card + 1$
15:        $S.append((N_i, card))$
16:    $S.sort(card)$
17:    for $i \leftarrow 1, G$ do
18:        $CN.append(S_i[0])$
19:    Return $CN$
```

descriptor includes its contact information (i.e. IP Address) and the vector identifiers of registered subscriptions.

6.4.5.2 Peer Selection

This service is in charge of periodically updating the neighbour table with a fixed number of node descriptors that share the most similar subscriptions with the current node. As shown in Algorithm 4, it selects a random neighbour Q and removes it from its neighbour table, it merges the node descriptors in its neighbour table and partial view and adds its own descriptor. Then, it selects a fixed number of descriptors G that carry subscription identifiers most similar to the ones registered in Q's active subscriptions table. G is the maximum pre-set fan out (i.e. degree) of a node. To obtain the descriptors, the peer selection service applies the subscription similarity estimation component on Q's active subscriptions identifiers and the ones in each of the node descriptors available in its merged view. It sends the $top - G$ descriptors with the highest similarity scores to Q, and receives not more than G descriptors from Q. Finally, the two nodes merge the received descriptors with their own partial view and neighbour table and rebuild the neighbour table by selecting the $top - G$ most similar descriptors to their own.

6.4.5.3 Subscriptions Similarity Estimation

This component identifies the candidate neighbours of a node Q by evaluating its subscription identifiers against the subscription identifiers from a set N of node descriptors (Algorithm 5). It obtains a similarity score for Q and each of the nodes in N, it then ranks the results in descending order, and returns the $top - G$ with the highest scores. To obtain a similarity score between two sets of subscriptions identifiers, similar to the approximate semantic matcher, we establish a pairwise mapping between the vector identifiers in the two sets based on cosine distance (Algorithm 3). Once the most likely mapping is chosen, the cardinality of their intersection is computed by simply counting the pairwise vector mappings that have a similarity measure greater-than a preset threshold t.

6.4.5.4 Discussion and Analysis

Voulgaris et al. [54] provide full proofs and extensive experimental results on various emerging properties exhibited by the overlay as a result of the swapping operation performed by the the peer sampling service. Analysed properties include, (1) guaranteed overlay *connectivity*, (2) convergence of the *average path length* and *clustering coefficient* to the exact values expected in a random graph, (3) a node *degree distribution* with low standard deviation, and (4) a *convergence* state that is independent from gossip length. The authors also highlight the role of the interaction between the peer sampling service and the peer selection service. Essentially, the peer sampling layer constantly supplies the peer selection layer with random links to nodes from all over the network. This aspect allows the peer selection service to add new joining nodes or nodes with recent emergent interest to the semantic cluster.

The emergent dissemination structure that clusters nodes with similar interests is conventionally referred to as a *semantic overlay network* [66, 67, 191, 192]. In OpenPubSub, the peer selection service identifies semantically close peers using an approximate semantic matching technique rooted in vector space semantics. The randomness introduced by the peer sampling service improves the traversability of the semantic vector space underlying SemanticPeer. In particular, the convergence properties of random peer sampling make up for the sparsity resulting from the high dimensionality of semantic vector spaces and the heuristics-driven unsupervised clustering techniques used to partition the space. It is important to note that assessing subscription similarity based on a pairwise vector identifier mapping and Cosine distance is one of many possible peer proximity measures that can leverage vector space semantics. For instance, utilizing shared event traffic between nodes has previously been employed in several approaches that explore the impact of overlay structure on the performance of publish/subscribe systems [62, 63]. Incorporating vector space semantics into the assessment of shared event traffic may induce the emergence of an enhanced semantic dissemination structure. Furthermore, utilizing both, subscription similarity and shared event traffic would result in clustering subscriber nodes with publisher nodes of interest, which would greatly improve event dissemination efficiency and effectiveness.

6.4.6 Connection Management Service

This component implements the SemanticPeer overlay construction algorithm [172]. We use bootstrapping to dynamically construct an overlay network where nodes constantly join and receive a partition of the vector space along with routing information. The connection management service is responsible for (1) processing Join requests, (2) finding an entry node, (3) partitioning the vector space and passing along one partition to the joining node, (4) setting up the received partition at the new joining node, and (5) populating the routing and splittable nodes tables. The notion of a splittable node essentially means that a node maintains a partition that may be further split into its child clusters. This table is used to randomly propagate Join messages to entry nodes. A node is removed from the table if the sub-tree it maintains (cluster) is compact enough and if it has no splittable neighbours.

6.4.7 Lookup Service

This service implements the SemanticPeer Lookup protocol. It facilitates four operations, namely, Store, Node Discovery, Exact Vector Lookup, and Approximate Vector Lookup. All operations are derived from the same lookup function. When passed a vector v, the Lookup Initiator consults a Close-Contacts Finder that measures the Cosine distance between v and all the node identifiers, i.e., mean vectors of maintained vector space partitions, in the local routing table. It then ranks the results and returns the $top - k$ closest identifiers. The Lookup Initiator proceeds to route v to the k closest nodes. Each of the recipients returns the $top - k$ closest node identifiers present in their local routing tables. Once the initiating node receives the results, it removes duplicate identifiers, finds the $top - k$ closest to v, and repeats the same process until the target nodes are identified.

The lookup function halts based on the type of operation initiated. A Node Discovery operation halts once one of the recipients of v identifies that v matches its own node identifier. An Exact Vector Lookup operation halts once one of the recipients identifies that it stores a data item with an identifier that matches v. The Store and Approximate Vector Lookup operations halt once the target nodes converge to the same nodes as the ones identified in the previous iteration or once a pre-set timeout has passed. The Store function then routes the data to be stored at the identified nodes, whereas, the Approximate Lookup function consults the identified nodes to return the closest data items they store by measuring the cosine distance between v and their identifiers.

6.4.8 Partitioning the Vector Space

In the following we improve on the agglomerative hierarchical clustering algorithm discussed in section 5.4.2 by adopting a distance measure based on shared nearest neighbours. The intuition is to tackle the sparsity of high-dimensional data which degrades the quality of clusters due to the locality of cluster composition stemming from the bottom-up nature of agglomerative clustering methods. More precisely, the successive merging of close clusters based solely on the cosine distance disregards the distribution of the entire set of vectors to be clustered. Our approach was inspired by the Jarvis-Patrick [161] clustering algorithm.

6.4.8.1 Jarvis-Patrick Clustering

The Jarvis-Patrick (JP) clustering method adopts a shared nearest neighbour approach for clustering. The key structure involved in this method is a shared nearest neighbour (SNN) graph. The SNN graph is based on creating links (edges) between vectors (nodes in the graph) based on the number of nearest neighbours they share. The algorithm starts with a set of n original observations i.e. vectors. Initially, we keep a record of the k nearest neighbours (k-NN) of each vector based on cosine distance. Now, let each vector represent a node in the SNN graph (see Fig. 6.4). Two nodes i and j will be joined by an edge in the graph if and only if i contains j in its k-NN list and vice versa. The strength (weight) of the edge corresponds to the number of nodes $n < k$ that are

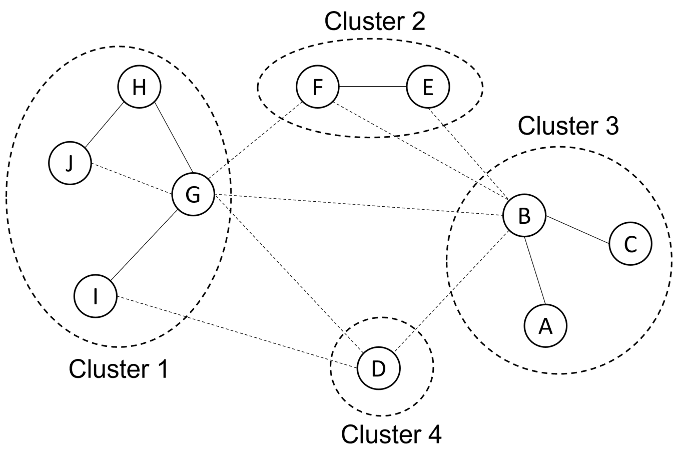


FIGURE 6.4: A toy example of an SNN graph resulting from Jarvis Patrick clustering.

common in both their k-NN list. The Jarvis-Patrick clustering method simply sets a threshold on the weight of the edges and deletes the edges with weights less than the threshold. The resulting connected components that remain form the final clusters.

Shared nearest neighbours provide an efficient way to handle the curse of dimensionality for high dimensional data. Both Houle et al. [193] and Steinbach et al. [194] illustrate the same fact. However, the issue with using JP clustering is that we cannot allocate clusters dynamically to nodes in a network while ensuring the traversability of the resulting partitions.

6.4.8.2 Agglomerative Clustering using k-NN

To handle high dimensionality and at the same time get the advantages of hierarchical clustering, we perform agglomerative clustering based on a distance metric stemmed from the shared nearest neighbours similarity measure used by the JP clustering algorithm. Initially, we compute the k-NNs of each vector based on cosine distance. Now, instead of creating an SNN graph and setting a threshold to remove edges, we populate a proximity matrix, where, for a vector space containing n vectors, the proximity matrix is an $n * n$ symmetric matrix with the ith and jth entry representing the cardinality of the intersection of the k nearest neighbours of the ith and jth vectors subtracted from k which results in a dissimilarity (distance) measure between the two vectors. The resulting matrix is then normalized to make sure all the distances obtained fall within 0 and 1. The proximity matrix is then fed to the average-linkage agglomerative clustering method which, in essence, resembles performing agglomerative clustering on the SNN graph, prior to determining a threshold to remove edges, by successively merging the nodes connected by edges with the highest weights. Consequently, The average-linkage

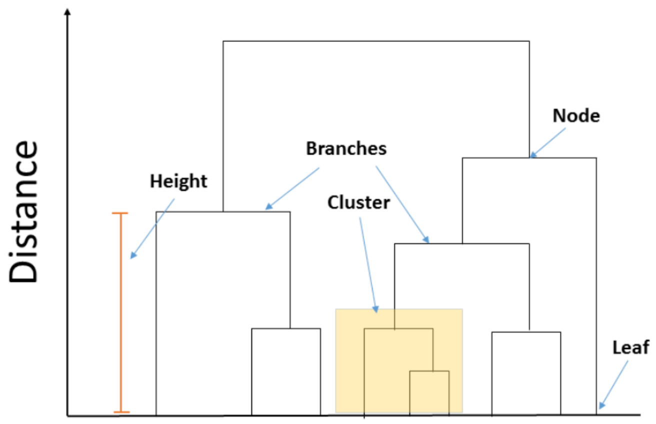


FIGURE 6.5: An illustration of a dendrogram [9]

method creates a hierarchy of clusters by successively merging closer clusters based on the shared k-NN distance measure.

The result of the clustering is a dendrogram structure (tree) that links together the original vectors at multiple levels (see Fig. 6.5). The height of the link between two clusters containing two original observations is known as the Cophenetic distance between the two observations. We use the Cophenetic Correlation Coefficient (CCC) [195] to verify the validity of the cluster tree. This technique compares the Cophenetic distances with the original distance data provided through the proximity matrix. The closer the value of the CCC is to 1, the more accurately the clustering reflects the original data. Our experiments show that applying cosine distance results in a CCC of 0.4, whereas applying the distance metric based on shared k-NN for multiple values of k results in a CCC around 0.8. The improvement in the quality of generated clusters is further validated in our rendezvous routing experiments which we discuss in detail in section 6.5.4.1.

6.5 Evaluation

This section provides an empirical evaluation of OpenPubSub.

6.5.1 Experiment Setup

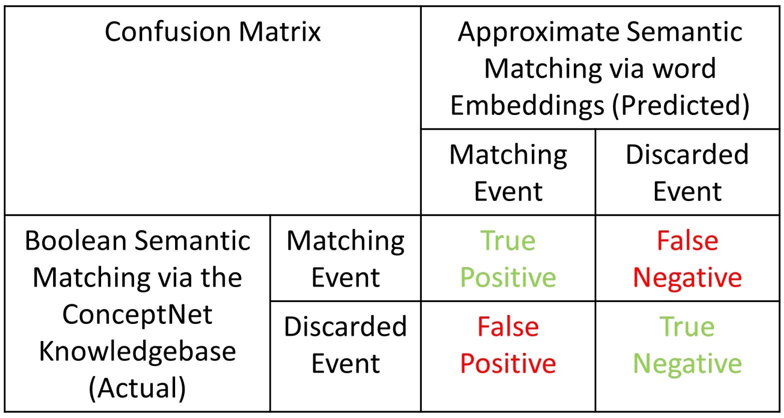
We implemented a prototype of OpenPubSub using SimPy. SimPy is a process based discrete-event simulation framework based on standard Python. With the help of Python generator functions and the yield statement, SimpPy allows us to model nodes in a peer-to-peer communication network and to test OpenPubSub on large simulated networks at the scale of thousands of nodes. The simulation framework is available on GitHub [189].

We simulate bootstrapping and messaging, enabling us to build an overlay network and to propagate events and subscriptions throughout the network. The peer sampling and peer selection protocols are cycle-based. They execute a view exchange between nodes at each cycle; a cycle represents the time unit between two executions.

6.5.2 Evaluation Methodology and Metrics

In our experiments, we construct an overlay network via bootstrapping, by allowing nodes to join, in order, receive space partitions, and exchange contact information. The vector space underlying the network comprises a set of 43060 word embeddings (i.e. vector representations of terms) that we acquire from the ConceptNet Numberbatch semantic vector set [196]. The set is clustered using an optimized implementation of the average-linkage hierarchical agglomerative clustering algorithm available in the Scipy Python library. The result of the clustering is a Dendrogram structure that is then passed to the Bootstrap node to initiate building the overlay. After the specified number of nodes join and populate their routing tables, we start by selecting nodes at random to propagate subscriptions to rendezvous nodes. We then select nodes at random to propagate events. We note that until this point, all routing operations are facilitated through the Lookup Service. We then measure the effectiveness of event matching via rendezvous routing. The variation in matching accuracy provides a measure for the effectiveness of the clustering algorithm and the iterative lookup operation used to traverse the space. To evaluate the Membership Management service, after the propagation of subscriptions throughout the overlay, the peer sampling and peer selection services are activated for a fixed number of cycles; nodes swap partial views and populate neighbour tables. Finally, we select nodes at random to propagate events and we measure the impact of gossiping on event matching.

We focus on measuring the effectiveness of our decentralized approximate event matching model. The evaluation is based on precision and recall as in Information Retrieval (IR) systems. In an IR system, precision is the fraction of the total number of relevant documents retrieved among the total number of retrieved documents, whereas recall is the fraction of the relevant documents retrieved among the number of known relevant documents. In the context of pub/sub systems, queries and documents are replaced by subscriptions and events. As shown in Fig. 6.6, we generate a ground-truth set by performing Boolean semantic matching over the events and subscriptions we generated. The matching engine uses a Knowledge-graph to perform query (subscription) translation prior to matching. The resulting matches form the known set of results to be compared with the matches resulting from running our approximate semantic matching experiments over a large peer-to-peer network. A True Positive (TP) result



Confusion Matrix		Approximate Semantic Matching via word Embeddings (Predicted)	
		Matching Event	Discarded Event
Boolean Semantic Matching via the ConceptNet Knowledgebase (Actual)	Matching Event	True Positive	False Negative
	Discarded Event	False Positive	True Negative

FIGURE 6.6: Precision & Recall. The outcome of approximate event matching as compared to query translation

refers to an approximate event match that is also present as a match to the same subscription in the known set, whereas a True Negative (TN) result refers to a discarded event that is not present as a match in the known set. On the other hand, A False Positive (FP) result refers to an approximate event match that is not present in the known set, and finally a False Negative (FN) result refers to a discarded event that is present as a match in the known set. In our experiments, recall measures the proportion of correct matches to all known matches such that $Recall = TP/(TP + FN)$, whereas precision measures the proportion of correct matches to all outcomes such that $Precision = TP/TP + FP$. The F-measure also known as the F-score is used to measure the overall accuracy of the matching process, it combines both recall and precision such that $F - score = 2 * Precision * Recall/Precision + Recall$.

6.5.3 Workload & Dataset

Constructing the overlay requires an underlying vector space with a finite number of vectors, each representing a unique semantic concept. To acquire our vocabulary we turn to Visual Genome [138] and ConceptNet [197]. Visual Genome is a dataset that provides annotated images to train Computer Vision models for tasks such as classification and object detection. It links visual concepts to language by mapping object labels, attributes and relationships to WordNet [157]. WordNet is a lexical database (ontology) that groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms (synsets), each expressing a distinct concept. Visual Genome provides a total of 108077 images with annotations linked to 7605 WordNet Synsets. The Synset names, e.g. 'pyramid.n.01', 'airport.n.01', 'flute.n.01', represent the seed representations of semantic concepts in our experiments. On the other hand, the ConceptNet knowledge base is a semantic network that captures a wide range of concepts and relations. Just like

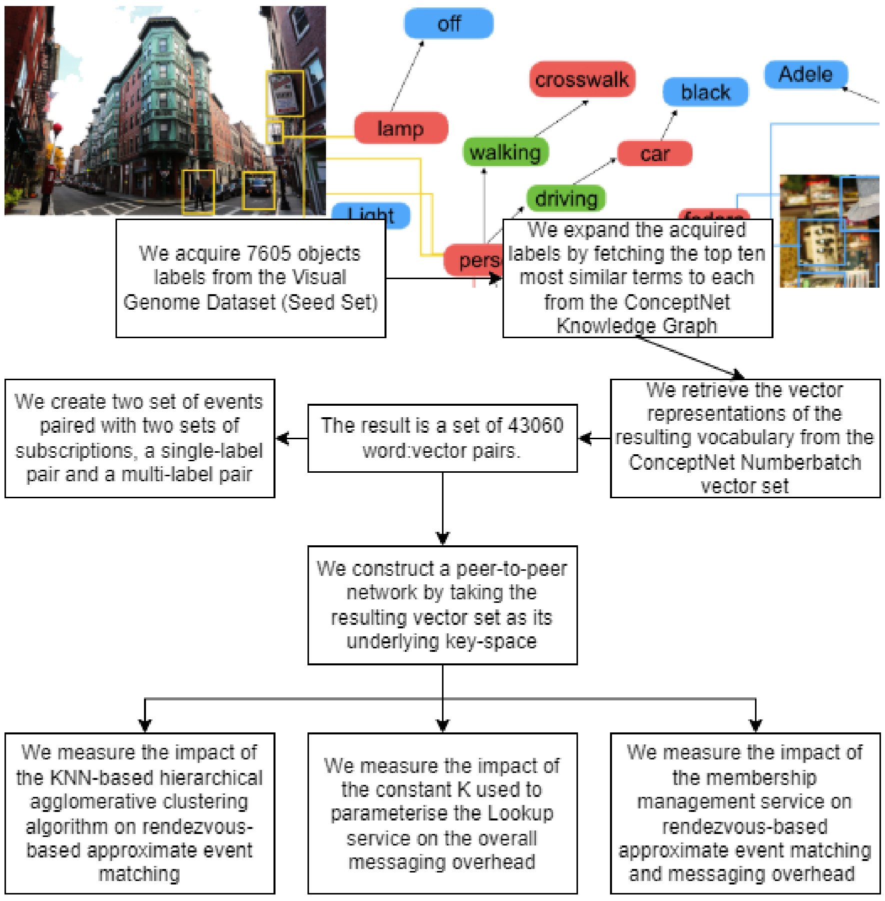

off
crosswalk
black
Adele
lamp
walking
car
driving
Light
pers
We acquire 7605 objects labels from the Visual Genome Dataset (Seed Set)
We expand the acquired labels by fetching the top ten most similar terms to each from the ConceptNet Knowledge Graph
We retrieve the vector representations of the resulting vocabulary from the ConceptNet Numberbatch vector set
The result is a set of 43060 word:vector pairs.
We create two set of events paired with two sets of subscriptions, a single-label pair and a multi-label pair
We construct a peer-to-peer network by taking the resulting vector set as its underlying key-space
We measure the impact of the KNN-based hierarchical agglomerative clustering algorithm on rendezvous-based approximate event matching
We measure the impact of the constant K used to parameterise the Lookup service on the overall messaging overhead
We measure the impact of the membership management service on rendezvous-based approximate event matching and messaging overhead

Figure 6.7: Semantic Expansion and Evaluation Workflow

WordNet, it can be used for tasks such as query expansion and identifying semantic similarity, but it focuses on linking concepts rather than words. It emphasises on informal conceptual connectedness for a more practical and context-oriented use over real-world texts such as the corpora commonly used to train distributional semantic models. The intuition behind using ConceptNet instead of WordNet is that Synsets or terms in WordNet are linked by a small set of semantic relations such as ‘is-a’ hierarchical relations and ‘part-of’ relations, whereas, ConceptNet covers pieces of commonsense knowledge to describe the real world, there are 20 kinds of relations categorized as causal, spatial, functional, etc [198].

6.5.3.1 Semantic Expansion

The goal of semantic expansion is to create a large vocabulary comprising collections of related semantic concepts whose vector representations can be used to construct a large structured peer-to-peer network. In a previous work we resorted to WordNet by fetching the most similar synsets to each seed synset based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) WordNet taxonomy [172]. However,

Word	Most related terms according to ConceptNet
"pyramid"	["pyramids", "pyramis", "pyramidal", "pyramid_scheme", "ponzi_scheme"]
"airport"	["aeroport", "international_airport", "airside", "air_station", "airfield"]
"flute"	["flutes", "flautist", "flutist", "tin_whistle", "clarinet"]
"elk"	["elk", "moose", "wapiti", "alces", "red_deer", "deer"]

Table 6.2: Semantic Concept Expansion via ConceptNet

our experiments showed that current pre-trained word embeddings, namely, Word2Vec [158] trained on Google News Text, and Fasttext [188] trained on the Wikipedia 2017 UMBC webbase corpus, did not reflect the formal linguistic relationships provided by WordNet. In this work, we use ConceptNet to expand the seed synsets we extracted from WordNet by fetching the top 10 most related concepts to each via the ConceptNet WebAPI. The result is a set of 43060 distinct semantic concepts. A snippet of the results is shown in table 6.2. The main reason behind choosing ConceptNet is the availability of ConceptNet Numberbatch [196], which is a set of semantic vectors that have semi-structured, common sense knowledge from ConceptNet itself. We obtain the vector representations of each seed and expanded concept resulting in a set of 43060 vectors.

6.5.3.2 Events & Subscriptions

We create two sets of events paired with two sets of subscriptions. A single-label pair (topic-based) and a multi-label pair (content-based). Here we note that the Approximate Semantic Matcher presented in section 6.4.2 behaves as a topic-based matcher when the number of labels specified in events and subscriptions is set to 1 and as a content-based matcher otherwise. The methodology for workload generation is similar to that by Hasan et al. [55].

- *Single-label:* For the topic-based workload, we select 5000 seed concepts as our event set (topics). We then replicate this set by creating 5000 subscriptions that satisfy an exact string match for every topic. Then, for every event e we select 3 related concepts at random as expanded events from e. This results in a set of 15000 expanded events. To generate the ground-truth set, we run a matching function that translates all the subscriptions to their related terms (using ConceptNet) before performing exact-string matching against the generated expanded events and we record the results. An example subscription would be s_1: **"airport"**, relevant events would be e_1: **"aeroport"**, e_2: **"air_station"**, and e_3: **"airfield"**.

- *Multi-label:* For the content-based workload, we select 1000 annotated images at random from the Visual Genome dataset as our seed event set. Images are annotated with 15 labels on average. For every event e, we generate 5 subscriptions by choosing up to 5 labels at random from e for every subscription. This results in a set of 5000 exact subscriptions. We then perform semantic expansion on the seed event set, where for every annotated image, we generate 5 expanded events by replacing a random number of its labels with related concepts. This also results in a set of 5000 semantically expanded events. An example seed event would be, e_1: ["spectacles", **"building"**, "sign", **"tree"**, "back", "trouser"]. An example subscription to e_1 would be s_1: ["building" $\wedge$ "tree"]. Finally, e_2: ["spectacles", **"constructing"**, "sign", **"pear_tree"**, "back", "trouser"] would be an expanded event from e_1. We generate the ground-truth set via translating labels in subscriptions to all their related terms and performing exact-string matching against the expanded event set.

To perform our experiments, we pair all labels with their vector representation which we retrieve from the ConceptNet Numberbatch semantic vector set. The evaluation workflow is outline in Fig. 6.7.

6.5.4 Experimental Results

In the following, we evaluate the approximate semantic matcher in a centralized setting over the single-label and multi-label workloads. The result is considered to be the upper-bound accuracy for running the same inside the network. The peer-to-peer approximate event matching process is facilitated via rendezvous routing and gossiping. The goal of the remaining experiments is to measure, to what extent, events and subscriptions are mapped to the same nodes in the network for the matching process to take place. The mapping can be impacted by several factors which include (1) the quality of generated partitions at network construction time, which is in turn impacted by the clustering method, (2) the performance of the Lookup function, which is impacted by the size of the network and the value of the constant k, and (3) the subscription similarity method which dictates the structure that emerges from running the membership management service.

6.5.4.1 Agglomerative clustering via shared K-NN

In this first experiment, we run the approximate semantic matcher, discussed in section 6.4.2, on our single-label workload in a centralized setting where events and subscriptions

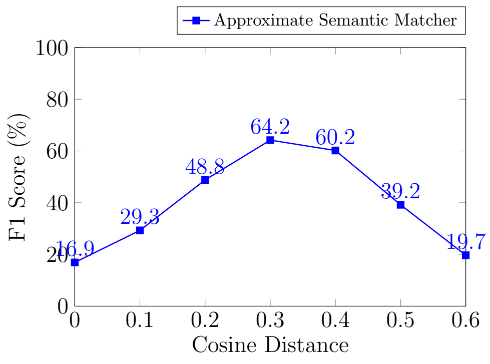


Figure 6.8: Centralized Topic-based Approximate Matching

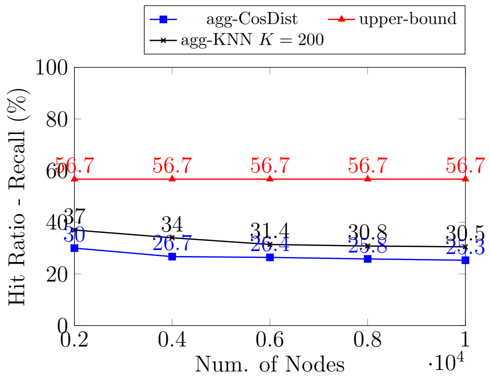


Figure 6.9: Topic-based Rendezvous Lookup Correctness

are sent to a single broker in charge of event matching and dissemination. We vary the threshold t (i.e. cosine distance) that the top-1 pairwise mapping should satisfy, i.e., be less than or equal to, for the event to be considered a match. In the case of single-label approximate matching, there will be a single one to one mapping between the vector representation of the subscription label and the vector representation of the event label, which will in turn have a Cosine distance value between 0 and 1. We calculate recall, precision, and the subsequent F1-score for each value of t. As shown in Fig. 6.8, the optimal value of t is 0.3, which results in the highest F1-score at 64.2%. For this same F1-score, recall is 57% and precision is 73%. In the next experiment we set the matching threshold t to 0.3, and we focus on measuring recall (i.e. hit ratio) under a decentralized setting. In other words, we run the approximate matching experiment over a structured peer-to-peer network. We run two experiments by employing average-linkage agglomerative clustering based on Cosine distance (agg-CosDist) for the first and on the shared $K-NN$ distance measure (agg-KNN) for the second. We vary the network size between 2000 and 10000 nodes. The matching scheme is based on rendezvous routing, where for each of the 5000 subscriptions, we pick a random node to issue a subscription by mapping the vector representation of the subscription label to a node in the network. On the other end, for each of the 15000 events (i.e. Topics) we pick a random node to publish an event. The higher the recall means that more matching events and subscriptions coincided at or were mapped to the same nodes in the network. We set k, which is used to parameterize the Lookup service to 1. As can be observed in Fig. 6.9, agglomerative clustering based of shared nearest neighbours consistently outperforms the one based on Cosine distance. This result can be attributed to the formation of better groupings of vectors that share semantic properties. Having higher quality clusters leads to nodes being assigned better partitions and more related terms being mapped to the same nodes. Accurately organising vector representations in a

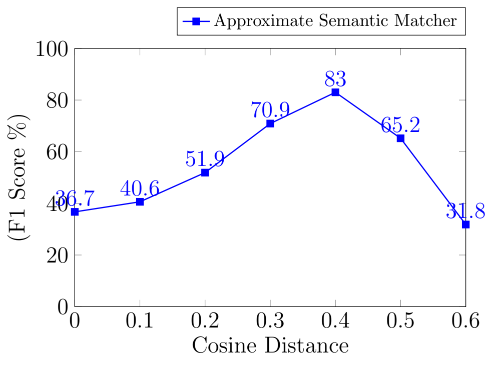


Figure 6.10: Centralized Content-based Approximate Matching

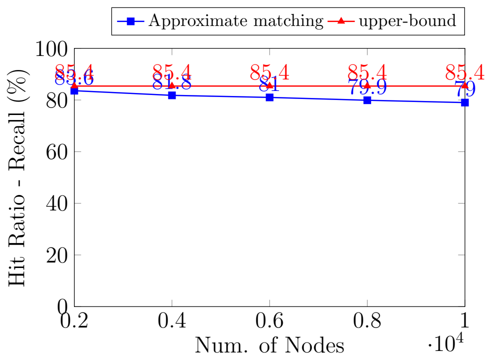


Figure 6.11: Content-based Rendezvous Lookup Correctness

hierarchy (i.e. Dendrogram structure) that reflects semantic relatedness enhances the effectiveness of rendezvous routing. For the remaining experiments we construct the network using the Dendrogram resulting from average-linkage agglomerative clustering using the shared nearest neighbour method.

6.5.4.2 Content-based Approximate Semantic Matching

In this experiment, similar to the single-label workload, we first run the approximate semantic matcher on the multi-label workload in a centralized setting while varying the value of t. As shown in Fig. 6.10, the optimal value of t is 0.4, which results in the highest F1-score at 83%. For the same value of t, recall is 85.4%, and precision is 80.4%. We then perform the same experiment over the overlay network by mapping multi-label events and subscriptions to nodes in the network using the event management service and the subscription management service respectively (as discussed in sections 6.4.3 and 6.4.4). We measure recall while varying the number of nodes in the network between 2000 and 10000. The matching scheme is also based solely on rendezvous routing with the gossiping service switched off. The Lookup service parameter k is set to 1. Fig. 6.11 shows that recall varies between 83% and 79% which is close to the upper-bound recall of 85.4 %. We attribute the high hit-ratio to two reasons, (1) enhanced Dendrogram structure and traversability due to utilizing shared $K-NN$ as a distance measure for agglomerative clustering (**R3.2. & R3.3.** Effectiveness), and (2) the high number of labels present in events (15 on average), which results in mapping the event to a relatively high number of nodes. Even further, each subscription contains up to 5 labels which also results in mapping each subscription to up to 5 nodes in the network. The trade-off here is a higher messaging overhead in addition to an increase in the number of duplicate matches.

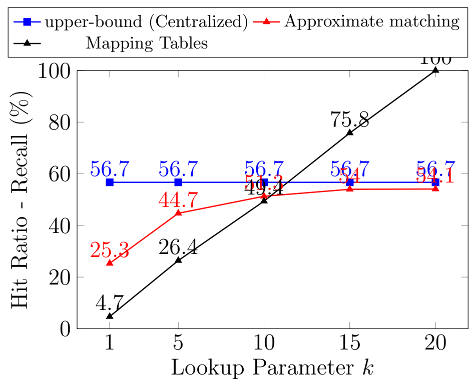


FIGURE 6.12: Topic-based Rendezvous Lookup Correctness via Rendezvous Routing

6.5.4.3 Impact of k on Rendezvous Lookup Correctness

In this experiment, we use the single-label workload to perform approximate matching over the OpenPubSub overlay network. The matching scheme is also based on rendezvous routing, however, we fix the network size to 10000 nodes and vary the Lookup Service parameter k between 1 and 20. This results in mapping events and subscriptions to the $top - k$ closest nodes, at each iteration of the lookup operation, as a result, each event and subscription would be delivered to the $top - k$ closest nodes as identified by the lookup service, which would ultimately increase the probability of a hit. As can be observed in Fig. 6.12, as the value of k increases, matching recall follows and nears the upper-bound for $k = 20$. We compare our result with an implementation of the Mapping Tables approach on the basis of theoretical guarantees on Lookup accuracy as presented in the Pastry DHT [6, 53]. We perform exact match key-based Lookup operations for the $top - k$ most related terms as reported in a local Mapping Table. Results show that the rendezvous lookup accuracy of mapping tables increases linearly with the value of k. For small values of k, the approximate lookup approach over OpenPubSub outperforms the baseline, whereas as k goes beyond half the amount of possible related terms as reported in the mapping table, the baseline approach outperforms approximate matching. We note that the messaging overhead can be considerably impacted due to mapping every event or subscription to multiple nodes in the network (Further discussed in section **??**). This increase in messaging overhead is a trade-off for achieving higher recall. We also note that the accuracy of our rendezvous routing approach is also bounded by the performance of the vector space model in a centralized setting. Further training and fine-tuning the distributional learning method may considerably enhance the results.

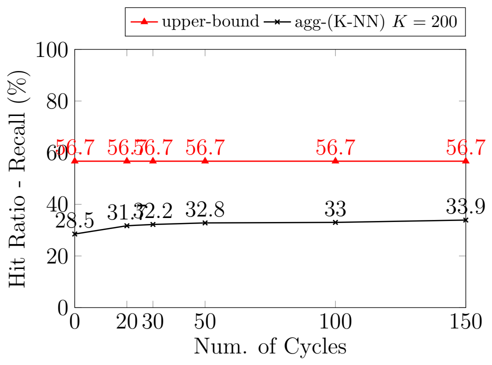


FIGURE 6.13: Membership management while varying the number of cycles

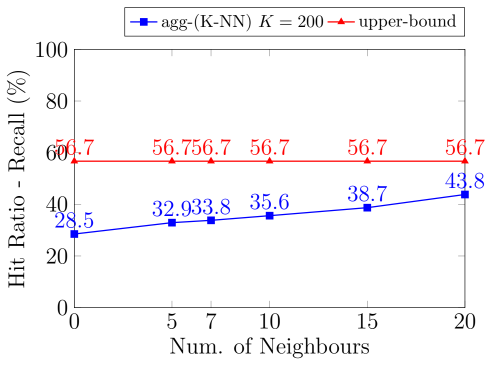


FIGURE 6.14: Membership management while varying the number of neighbours

6.5.4.4 Impact of the membership management service on matching recall

In the following we evaluate the impact of peer sampling and clustering on event matching recall. We fix the network size to 10000 nodes, we then use the single-label workload by choosing nodes at random to send each of the 5000 subscriptions to their respective rendezvous nodes. following that we run the membership management service as discussed in section 6.4.5. Finally we choose nodes at random to publish each of the 15000 expanded events (topics). In the first experiment we vary the number of cycles which essentially dictates how many times the peer sampling service and the peer clustering service perform view exchanges and neighbour table updates prior to event publishing. The number of neighbours (i.e. fan out) is set to 5. As can be observed in Fig. 6.13, when the number of cycles is set to zero, meaning that the membership management service is turned of, recall is at 30.2%. This is the result of relying solely on rendezvous-routing with the Lookup service parameter k set to 1. As the number of cycles increases, rendezvous nodes start to populate their neighbour tables with nodes sharing similar subscriptions. This is reflected by the consistent increase in recall until the number of cycles reaches 150. However, we can also see that recall starts to stabilise as the number of cycles goes above 50 which shows how periodic gossiping is starting to converge towards a stable state. In the next experiment, we fix the number of cycles to 100 and vary the number of neighbours to be selected by the peer selection service. As shown in Fig. 6.14, as the number of neighbours increases from 5 to 20, the matching recall consistently increases from 32.8% to 43.8%. This increase in recall is at the expense of additional overhead due to forwarding events to more nodes after reaching their respective rendezvous node.

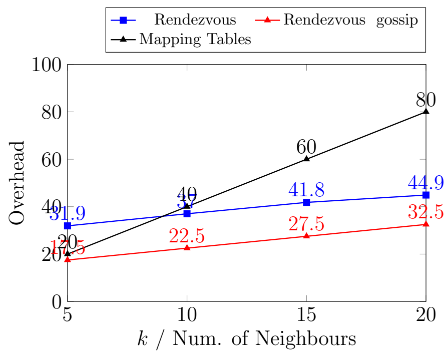


FIGURE 6.15: Impact of the event matching and routing techniques on messaging overhead

6.5.4.5 Messaging Overhead

In this experiment the messaging overhead is defined as the average number of nodes visited upon publishing an event and reaching a matching outcome. We compare the messaging overhead when applying rendezvous-based routing while varying the value of k between 5 and 20 versus applying our hybrid approach which comprises rendezvous-based routing with k set to one in addition to the membership management service activated. The latter approach allows rendezvous nodes to actively look for other nodes (neighbours) that share similar subscriptions. For the hybrid approach, upon reception of an event e, in addition to matching e against hosted subscriptions, the rendezvous node also forwards e to all its neighbours. We vary the number of neighbours between 5 and 20. As can be observed in Fig. 6.15, the rendezvous-based approach results in considerably more overhead due to the recursive nature of the underlying approximate Lookup operation. However, as discussed earlier and shown in Figs. 6.12 and 6.14, the rendezvous-based routing approach achieves more recall than the hybrid approach for higher values of k. Here we note that the goal of the hybrid approach is to benefit from node clustering to increase recall while keeping the value of k to a minimum. Hence the reason why k is set to 1 for the hybrid approach. This result reemphasises the trade-off between achieving high recall and minimising messaging overhead (**R3.2.** Effectiveness and **R3.3.** Efficiency).

We note that for conventional approaches, i.e., content-based and concept-based, the total number of lookup operations performed for the same workload would be equivalent to the number of rules that exist in the system. Given that we acquire the top 10 most similar concepts to each seed concept. And given that we expand our event set by picking random related concepts. The number of rules in the content-based model or the number of entries in the Mapping Tables approach (Concept-based model) would be equivalent to the size of the entire vocabulary. Hence, the number of Lookups would be tenfold that of OpenPubSub's (**R3.1.** Number of Rules). This is further exacerbated by the iterative nature of Lookup protocols which are often parameterized by a constant k that increases the number of nodes targeted at every iteration, e.g., Kademlia. The theoretical logarithmic bounds on the number of hops in conventional DHTs, i.e., Pastry, have been shown to be very close to experimental simulation results based on PeerSim [53**?**]. On the basis of such bounds and as shown in Fig. 6.15, the iterative nature of lookup protocols ultimately leads to a linear increase in messaging overhead (**R3.3.** Efficiency). At low values of k (1 and 5), Mapping Tables would result in lower messaging overhead at the expense of much lower event matching recall (Fig. 6.12). However, as the value of k increases, matching recall follows but so does the messaging overhead. We observe that employing the hybrid routing approach over OpenPubSub creates the least messaging overhead at the expense of moderate matching recall. Conversely, the rendezvous routing approach results in high matching recall at the expense of moderate messaging overhead. The computational and generalizable nature of vector space semantics allows for expanding the vocabulary to very large semantic spaces, whereas, mapping tables or ontologies, would require time and expert knowledge to create. Most semantic hierarchies are limited to specific domains, e.g. Animals, Medical Equipment, Sports, etc., which ultimately limits the semantic coverage of the system. Finally, in an attempt to cater for different constraints posed by subscribers on event matching recall or messaging overhead, one can expose or prompt Quality of Service (QoS) or Quality of Experience (QoE) parameters that can be set by subscribers as part of the language model. This feature would allow the system to choose between the different techniques proposed, e.g., the rendezvous-based routing approach, the hybrid routing approach, or the mapping tables approach. Having control over the choice of event routing and matching methods would aid with prioritizing matching recall over messaging overhead or vice versa and ultimately cater for user needs or present overlay network capabilities and constraints.

6.5.5 Conclusion

In conclusion, we propose OpenPubSub, a content-based approximate semantic peer-to-peer publish/subscribe system. Participants (i.e. peers) in an overlay network loosely agree on a large corpus of text **(R3.1.)**. The corpus is then mapped to a vector space by leveraging learning methods based on distributional semantics. The resulting vector space is then clustered into groupings of vectors that fall in semantic proximity to each other. Each cluster is assigned to a node in the overlay. The peer-to-peer infrastructure provides low level communication abstractions by exposing a routing substrate that maps the vector representations of terms present in events and subscriptions to a location in the space via an iterative lookup operation **(R3.3.)**. We augment the overlay with a content-based approximate semantic event matcher **(R3.2.)** and a membership management protocol that groups nodes with shared interests into semantic sub-overlays **(R3.3.)**. Experiments with approximate events and subscriptions from image annotations and knowledge graphs show a 15% improvement in event matching recall as a result of utilizing shared $K - NN$ as a distance measure for hierarchical agglomerative clustering **(R3.2.)**. Furthermore, the rendezvous routing approach reduces overall messaging overhead by up to 44% and increases event matching recall by up to 53%. Finally, the utilization of Gossiping reduces the overall messaging overhead by up to 59% and increases event matching recall by up to 35% **(R3.3.)**.

6.6 Summary

This chapter tackles research question (Q3.) and proposes OpenPubSub, a decentalized approximate semantic matching model that addresses the emergence of multimedia-based services and applications in the Internet of Multimedia Things. More precisely, the wide semantic content space of human recognisable concepts that can be stemmed from video and audio content introduces challenges in wide scale data integration and dissemination. This chapter proposes a content-based peer-to-peer publish/subscribe system that offers loose semantic coupling between its participants and facilitates the exchange of information between geographically dispersed, heterogeneous, and autonomous data producers and consumers. OpenPubSub addresses an inherent limitation in the current implementations of DHTs arising from their underlying logical key-spaces and exact-match key-based routing primitives. The scale of the content space limits the applicability of exact string matching when participants use different terms to address the same semantic concepts. It replaces logical key-spaces with a high-dimensional vector space. We further augment the overlay with an approximate semantic Lookup function coupled with a membership management protocol based on gossiping. When combined, the two routing strategies facilitate mapping related terms to the same nodes with high probability and by that enable us to apply rendezvous routing strategies followed by approximate semantic event matching rooted in uncertain schema matching.

Chapter 7

Conclusions and Future Work

7.1 Thesis Summary

This thesis addresses research problems that arise when attempting to support multimedia content as native event types in distributed publish/subscribe systems. The three main research questions tackled are outlined below.

Q1. How to integrate fast and resource-aware knowledge extractors into the forwarding function of distributed publish/subscribe systems comprising federated broker overlays?

Q2. How can loose semantic coupling be achieved within the exact-match lookup functionality offered by conventional DHTs underlying structured peer-to-peer networks while ensuring the dynamic organization, allocation, and retrieval, of data based on their shared semantic and syntactic properties?

Q3. How to support a conjunctive content-based subscription language over unstructured multimedia events spanning a large semantic content space in structured peer-to-peer networks?

The recent emergence of multimedia-based services and applications in the Internet of Things (IoT) has led to a noticeable shift in the type of data traffic generated by sensing devices and the data management requirements they demand. More precisely, the wide semantic spectrum of human recognisable concepts that can be stemmed from video and audio data introduces a very large semantic content space. The Internet of Multimedia Things (IoMT) is an emerging paradigm that concerns the processing and delivery of multimedia content, e.g. audio, video, acquired from the physical environment. The enormous growth of multimedia sensing devices in the Internet of Things (IoT) has emphasised the need for bringing additional functionality that deals with the complex nature of multimedia content. Connecting sensing devices to the internet is at the core of many IoMT applications and plays an immediate role in facilitating the acquisition and on-the-fly processing of data for real-time applications such as Autonomous Driving, Traffic Management, and Smart Agriculture. The scalability and interoperability offered by the publish/subscribe communication pattern has proved beneficial for connecting heterogeneous sensing devices and actuators to each other and to the internet. There are many publish/subscribe protocols being used in IoT applications today, perhaps most notably MQTT and DDS but there are numerous others.

Distributed publish/subscribe systems often comprise an application-level overlay of nodes that organize on top of an underlying physical network. The overlay infrastructure has direct implications on the performance of the system. It dictates the employed event matching and routing strategies. Overlay networks, as seen in the literature, take the form of semi-centralized federated broker overlays or fully decentralized peer-to-peer overlays. The former scales to dozens of nodes and is often managed by a single administrative entity, whereas, the latter often scale to thousands of nodes and is widely known to be self-organizing and permission-less where nodes join and leave at any time.

The problem at hand entails integrating knowledge extractors, i.e. Operators, into the event processing pipeline within the event matching engine of publish/subscribe systems. These operators are in charge of uncovering meaningful semantic concepts from the multimedia data being disseminated. Extracted concepts can then be evaluated against expressions submitted by subscribers as in conventional content-based event matching approaches. In the case of federated broker overlays, the closed nature of the system, and the availability of a single entity with explicit control over nodes in the overlay allows for the integration and management of such operators. The challenges that arise concern the optimization of the execution of such operators over incoming streams of multimedia events. The overlay should (1) ensure fast extraction of semantic concepts **(R1.1.)**, (2) leverage the distributed nature of the system **(R1.2.)**, and (3) ensure the rapid delivery of matched events to interested subscribers **(R1.3.)**. On the other hand, the decentralized and self-organizing nature of peer-to-peer networks forces participating publisher nodes, subscriber nodes, or intermediary nodes, to manage their own operators. The wide semantic spectrum of human recognizable semantic concepts that can be stemmed from multimedia content gives rise to a semantic gap between publishers and subscribers. More precisely, participants may use different terms to describe the same or highly related semantic concepts. This aspect results in a semantic matching and routing challenge that is exacerbated by the dynamic of peer-to-peer networks. In particular, the peer-to-peer network should facilitate the distributed organization of data items based on semantic and syntactic relationships between them **(R2.1.)**. It should also allow for performing approximate semantic lookup operations that allow for matching two semantically related terms via rendezvous routing **(R2.2.)**. Furthermore, participants in the network should have a low barrier to entry which prevents them from establishing strict semantic agreements **(R3.1.)**. With a loose semantic agreement in place, the system should facilitate an efficient and effective mapping **R3.3.** and matching **(R3.2.)** of event and subscription semantics. In this work, we tackle the challenges that arise in both cases and propose solutions that ultimately support multimedia content as native event types in distributed publish/subscribe systems.

7.2 Main Contributions

The section outlines the contributions of this thesis.

7.2.1 Distributed Shared-Filer Ordering and Placement (Q1, R1.1. and R1.2.):

This contribution comprises embedding light-weight binary filters (i.e. image classifiers) into the forwarding function of brokers organized into a federated overlay. The intuition is that the cooperation of multiple binary classifiers provides a simple and broadly applicable knowledge extraction technique that allows for extending the semantic coverage of the system's matching engine by provisioning new classifiers as needed to support a continuously evolving content-space. We further formulate the shared-filer placement and ordering across brokers in a federated broker overlay and the provisioning of event matching and routing algorithms (i.e. MEC and EMEC) for the adaptive ordering and distribution of classifier execution along paths towards interested subscribers (Zaarour et al. [40]). The shared-filter placement and ordering problem consists of a set of subscriptions S, a set of commutative Boolean filters F, and a set of brokers B. Each subscription is represented as a conjunction of filters where the evaluation of each filter on an event results in a Boolean outcome of either *true* or *false*. The objective is to minimize the Processing Delay Cost which may be defined as the time required by filters executed along a path towards a subscriber to process an event. The proposed algorithms leverage a Publication Routing Graph (PRG) maintained at each broker. The PRG consists of two bipartite graphs, one that maps subscriptions to filters, and one the maps subscriptions to the links they came from. The algorithms leverage the cost, selectivity, and the PRG edge-coverage, of individual filters to greedily order and distribute their execution through-out the overlay. Experiments with varying publish/subscribe workloads show clear improvements in the average processing delay cost perceived by individual subscribers, a negligible increase in resource consumption across the overlay, and a low overall messaging overhead.

7.2.2 Topological Reconfiguration of Broker Overlays (Q1, R1.3.):

A skewed distribution of subscribers with shared interests across brokers in an overlay network may result in increasing overall event matching and routing costs. The participation of pure-forwarders in the processing of events increases redundant processing. Pure-forwarders are simply brokers that participate in event processing without having any locally interested subscribers. The existence of such brokers is exacerbated by the early offloading of filters to be executed by downstream brokers. This aspect leads to the execution of the same filters on the same event multiple times at different brokers along separate paths in the overlay. By creating regionalism in the overlay and grouping together brokers with a similar workload, we can enhance the performance by reducing the redundancy overhead. This contribution comprises two methods that utilize vector distance measures to quantify how close two brokers are as per their subscription workload. On the basis of these similarity metrics, the overlay topology is modified by eliminating links between brokers with low similarity and provisioning new links between brokers with high similarity.

7.2.3 Utilizing a semantic vector space as a key-space to build semantically aware DHTs (Q2, R2.1., and Q3, R3.1):

Assuming that semantic coupling can be quantified by the number of mappings between terms, then a semantic model that condenses these mappings can be very useful. Ontological models require granular agreements on term meaning while distributional vector space models leverage the statistics of term co-occurrences in a large corpus to establish semantics. The latter allows event producers and consumers to establish a loose semantic agreement on large corpora of text. This contribution consists of replacing the logical key-space underlying conventional DHTs with a vector space that preserves the semantic properties of the data being mapped (Zaarour et al. [47]). We propose methods based on hierarchical clustering to partition the space and create traversable groupings of word embeddings. We dynamically assign the resulting partitions to nodes in a structured peer-to-peer network where each node maintains a partition of a large semantic vector space comprising tight grouping of vectors representing terms that have semantic relationships.

7.2.4 Mapping semantic data via a novel distributed and iterative vector lookup operation (Q2, R2.2, and Q3, R3.3):

Conventional DHTs leverage Key-based routing primitives to perform ID-centric lookup operations. Such primitives are aided with a closeness metric that allows nodes initiating lookup operations to iteratively find closer nodes to the key being mapped. With the aid of consistent hashing, data items are mapped to locations in the key-space following a uniform random distribution. The uniqueness of each generated key allows for employing exact match search queries. However, key-based routing primitives stand in the way of matching two semantically related terms. The random distribution of data items leads to mapping semantically related terms to arbitrary locations in the network. To mitigate this issue, after replacing logical key-spaces underlying DHTs with a semantic vector space. We propose a semantically decoupled lookup protocol that can be initiated by any node in the overlay (Zaarour et al. [47]). It takes the vector representation v of a data item d, and recursively finds closer nodes until it identifies the target node maintaining the vector space partition where v falls. The receiving node stores data items that share semantic properties with d. This aspect allows us to map data items that share semantic properties to the same nodes via rendezvous routing (Zaarour et al. [172]).

7.2.5 Approximate Semantic Event Matching Rooted in Uncertain Schema Matching (Q3, R3.2):

We propose a matching scheme that decides on the relevance between events and subscriptions based on distributional semantics (Zaarour et al. [48]). Distributional semantics offers a way of quantifying the semantic relatedness between two linguistic items based on their distributional properties in large corpora of text. A distributional semantic model maps terms to high dimensional numerical vectors where we can decide on the relevance between terms based on how close their vector representations are oriented in space (i.e. by measuring the Cosine of the angle between them). Events are comprised of a set tuples, each containing a word:vector pair. Each pair collectively represents a semantic concept that was extracted from a multimedia data item. Subscriptions comprise conjunctions of predicates, each also containing a word:vector pair. We quantify there overall semantic relatedness by obtaining a similarity score for each pairwise mapping between the vectors in predicates and tuples. We then identify the most-likely mapping and compute an aggregated overall similarity score.

7.2.6 Semantic peer clustering and regionalism via gossip-based peer sampling and selection (Q3, R3.3):

In a structured peer-to-peer network offering a publish/subscribe communication service, events and subscriptions may be submitted by any participating node in the network. For the matching process to take place, events and subscriptions should be mapped to the same node in the overlay. A loosely semantically coupled lookup function allows related terms to rendezvous at certain nodes in the overlay. However, due to the high dimensionality of the space and the unsupervised nature of hierarchical clustering, lookup operations can be highly probabilistic. Some events or subscriptions might get erroneously mapped to nodes maintaining distant semantic groupings. To mitigate this issue, we propose a membership management protocol that helps constrain the propagation of events and increase the probability of a match by maintaining sub-overlays of nodes that share semantic properties (Zaarour et al. [48]).

7.3 Limitations and Open Questions

The proposed approach have a number of limitations and shortcoming as is outlined below.

- **Static selectivity of filters.** To tackle the shared-filter placement and ordering problem in federated broker overlays, we propose adaptive greedy algorithms (i.e. MEC and EMEC) that take into account the cost, selectivity, and edge-coverage of individual filters. Based on the *true* or *false* outcome of a filter evaluation against an event, the algorithm proceeds to update the unit costs of the remaining filters and selects the filter with the lowest execution cost and the highest *expected* edge coverage in the local Publication Routing Graph (PRG). However, the expected edge coverage is subject to two metrics, one being the selectivity of the filter, and the other being its popularity among the remaining unresolved subscriptions. The problematic aspect or limitation we identify is that the selectivity measure is assumed to be static and not updated at run-time. Yet, variations in the distributions of semantic concepts that appear in incoming multimedia event streams can have a major impact on the selectivity of filters. Prior works have used approaches based on windowing and confidence interval to update the selectivity of filters at run-time in a centralized setting. However, in a distributed broker overlay setting, the adaptive ordering results in executing the same filter at different locations in the network. Hence a filter's selectivity depends on where the filter is being executed. Hence, devising strategies that take into account the dynamics of subscribers and filter executions across a broker overlay to update the selectivity measure at run-time is a challenging aspect that can improve the system's efficiency and reduce the Processing Delay Cost.

- **Overlay reconfiguration.** The distributed execution of filters across a broker overlay leads to an increase in redundant processing due to the execution of the same filters at various paths towards interested subscribers. To mitigate this issue, we devise two broker similarity metrics and an overlay construction algorithm that calculates the distance between all brokers in the system using either of the similarity measures, then ranks them from the most similar to least, and starts to select edges in a descending order. The limitation with this approach is that the algorithm is assumed to have knowledge about all the brokers in the overlay. In addition, updating the overlay structure needs to be done offline and in a periodic manner. This aspect may not be ideal in practice due to the live and real-time nature of publish/subscribe systems. The challenge at hand demands devising a strategy to eliminate links between brokers with low similarity and provisioning new links between brokers with high similarity in a distributed manner. Each broker in the overlay should be able to leverage its local knowledge about the overlay to identify new potential neighbours and current neighbours that should be eliminated.

- **The curse of high-dimensionality.** Conventional DHTs are built on the basis of virtual key-spaces that take the shape of rings, trees, and d-dimensional coordinate spaces. The key-spaces are partitioned arbitrarily and each node is assigned a unique key as its identifier and maintains a partition of the space with well-defined boundaries. Such networks are ideal of ID-centric exact match queries but fall short when attempting to match two semantically related terms. To mitigate this issue, we replace conventional logic key-space with a high-dimensional vector space that preserves the semantic properties on the data being mapped. However, as the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse. This sparsity is problematic mainly because organizing and searching data often relies on detecting areas where objects form groups with similar properties; in high dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient. We propose unsupervised clustering algorithms to partition the space, however the traversability of the space after partitioning becomes highly probabilistic which we further address by using a constant K to increase the search space of lookup operations. This approach trades off an increase in routing accuracy with a high messaging overhead. The limitation at hand demands provisioning dimensionality reduction techniques and optimised routing strategies that minimize the messaging overhead while increasing the effectiveness of matching two semantically related terms.

- **Lookup Path Convergence.** In conventional DHTs, when two arbitrary nodes issue lookup operations for the same key, the underlying KBR primitive iteratively finds closer nodes until it reaches the target node. As it gets closer to the target, there is high probability that the two paths will meet at the same node before reaching the rendezvous node. The proximity metrics used, e.g., Kademlia's XOR metric, or Chord's clockwise identifier circle, are unidirectional which allows for paths to converge before reaching the rendezvous node. This enables Caching and generating better spanning trees (e.g., Scribe [18]). However, the high-dimensionality of semantic vector spaces and the angular nature of the proximity metric used by the approximate lookup operation (i.e. Cosine distance) prevents the convergence of two paths. This limitation is mainly concerned with the ability to create efficient spanning trees that aid with the dissemination of events to a group of interested subscribers.

- **Load-balance.** In conventional DHTs, the underlying key space is arbitrarily partitioned among nodes in the overlay. Data items are mapped to locations in the key-space via consistent hashing. These two aspects result in the assignment of data items to nodes following a uniform random distribution. This inherent property carries a major benefit in terms of load-balancing across nodes in the overlay. However, in the case of a semantic vector space, the partitioning is not arbitrary. Clustering methods are used to make sure that data items sharing semantic properties are assigned to the same group or cluster. As a consequence, the distribution of data items being mapped to nodes in the overlay network is subject to the distribution of the underlying vector space. Which is in turn also impacted by the popularity of terms being searched for by arbitrary nodes in the overlay. Hence, this limitation demands provisioning approaches that identify and relieve hot spots in the network. That is, nodes that are overloaded with search requests. Such nodes may pose bottlenecks and lead to reliability issues when it comes to storing and retrieving highly popular data items.

7.4 Avenues for Future Work

This section discusses interesting research directions that we identify based on the research problems tackled in this work.

- **Multi-class Classifiers.** To support multimedia content as native event types in federated broker overlays, we propose the integration of binary classifiers in the forwarding function of individual brokers. Such classifiers would serve as a generalizable model for extracting meaningful semantic concepts from incoming data streams. The intuition is that users may continuously provision new classifiers to expand the semantic coverage of the matching engine in response to a continuously evolving content-space. Such filters work together in a commutative manner where each filter is mutually independent which allows for leveraging the ordering of their execution in the face of overlapping subscriptions. The introduction of multi-class classifiers, that is, classifiers that are able to detect a handful of classes (i.e. semantic concepts) would ease with expanding the semantic coverage of the system but would render the proposed shared-filter placement and ordering algorithms insufficient for identifying a near optimal ordering and distribution of their executions across brokers in the overlay. Hence an interesting avenue for future work would be to look at filters that are trained to detect multiple concepts instead of binary filters. This increases the complexity of the ordering problem where each individual filter can hold the impact of executing multiple filters on the current unresolved subscriptions.

- **Self-Organizing Semantic DHTs:** In this work we presented algorithms to construct a semantic DHT on the basis of a semantic vector space. The resulting overlay network organises data in a way that takes into account shared semantic properties. Yet, conventional DHT's are known to be self-organising, coping naturally with node departures and arrivals, and resistant to node failures. Our bare-bones design of a semantic DHT, however, assumes a static overlay. This challenge demands provisioning approaches that deal with nodes leaving the network and node failures. Such approaches need to ensure redundancy, that is, data items being stores at various locations in the network. In addition to making sure that nodes have the ability to reallocate the partition they are responsible for to other nodes before leaving the network.

- **Distributional Models of Word Meaning:** Many distributional models based on neural networks exist today (i.e. BERT, Fasttext, etc.). We acknowledge that the quality of the Lookup Operations in SemanticPeer is also dependent on the quality of the vector representation of terms.thus an interesting avenue for future work is exploring various learning methods and text corpora and assessing their utility for generating semantic vector spaces that can be used to construct semantically aware DHTs.

7.5 Conclusion

The work tackled in this thesis concerns supporting multimedia content as native event types in distributed publish/subscribe systems. The work can be split into two main categories of approaches. In the first category, we focus on semi-centralised broker overlays and propose the integration and optimization of binary filters that extract meaningful semantic concepts from the multimedia data being disseminated. The optimization of their execution is facilitated via an adaptive greedy algorithm that orders and distributes their execution across brokers in an overlay based on the execution cost, selectivity, and popularity of individual filters (Zaarour et al. [40]). We further propose two metrics to assess the similarity in workload between two brokers. The metrics are then used to modify the overlay structure in response to skewed distributions of subscribers with similar interests across brokers in the overlay. In the second category of approaches, we focus on fully decentralized peer-to-peer network. We propose approach to elevate the tight semantic coupling that exists between publishers and subscribers. In particular, we identify that the exact match lookup functionality offered by underlying DHTs would not suffice to support the wide semantic spectrum of human-recognizable concepts that can be stemmed from multimedia data. We present a peer-to-peer content-based publish/subscribe model that we equip with an approximate semantic event matcher leveraging distributional semantics (Zaarour et al. [47, 48]). We propose a hybrid event routing model that combines rendezvous-based routing and gossiping over a structured peer-to-peer network. The network is built on the basis of a high-dimensional semantic vector space as opposed to conventional logical key-spaces. We propose methods to partition the space, construct a semantic DHT via bootstrapping, perform approximate semantic lookup operations, and cluster nodes based on their shared interests.

Bibliography

[1] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1, 2000.

[2] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44 (3):1–62, 2012.

[3] Roberto Baldoni and Antonino Virgillito. Distributed event routing in publish/-subscribe communication systems: a survey. *DIS, Universita di Roma La Sapienza, Tech. Rep*, 5, 2005.

[4] Souleiman Hasan. Loose coupling in heterogeneous event-based systems via approximate semantic matching and dynamic enrichment. *National University of Ireland, Galway*, 12, 2016.

[5] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[6] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001.

[7] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[8] Alessandro Lenci. Distributional models of word meaning. *Annual review of Linguistics*, 4:151–171, 2018.

[9] Andreas Tilevik. Evaluation of clustering methods for analyzing drug cytokine profiles, 2017.

[10] Edward Curry and Amit Sheth. Next-generation smart environments: From system of systems to data ecosystems. *IEEE Intelligent Systems*, 33(3):69–76, 2018.

[11] Edward Curry, Wassim Derguech, Souleiman Hasan, Christos Kouroupetroglou, and Umair ul Hassan. A real-time linked dataspace for the internet of things: enabling "pay-as-you-go" data management in smart environments. *Future Generation Computer Systems*, 90:405–422, 2019.

[12] Edward Curry. *Real-time linked dataspaces: Enabling data ecosystems for intelligent systems*. Springer Nature, 2020.

[13] Khin Me Me Thein. Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research*, 3(47): 9478–9483, 2014.

[14] Lovisa Johansson and David Dossot. *RabbitMQ Essentials: Build distributed and scalable applications with message queuing using RabbitMQ*. Packt Publishing Ltd, 2020.

[15] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3):332–383, 2001.

[16] Hans-Arno Jacobsen, Alex Cheung, Guoli Li, Balasubramaneyam Maniymaran, Vinod Muthusamy, and Reza Sherafat Kazemzadeh. The padres publish/subscribe system. In *Principles and Applications of Distributed Event-Based Systems*, pages 164–205. IGI Global, 2010.

[17] Peter R Pietzuch and Jean M Bacon. Hermes: A distributed event-based middleware architecture. In *Proceedings 22nd international conference on distributed computing systems workshops*, pages 611–618. IEEE, 2002.

[18] Miguel Castro, Peter Druschel, A-M Kermarrec, and Antony IT Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications*, 20(8):1489–1499, 2002.

[19] Vinay Setty, Maarten Van Steen, Roman Vitenberg, and Spyros Voulgaris. Poldercast: Fast, robust, and scalable architecture for p2p topic-based pub/sub. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 271–291. Springer, 2012.

[20] Aleksandar Antonić, Martina Marjanović, Krešimir Pripužić, and Ivana Podnar Žarko. A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things. *Future Generation Computer Systems*, 56:607–622, 2016.

[21] Cenk Gündoğan, Peter Kietzmann, Thomas C Schmidt, and Matthias Wählisch. Hopp: Robust and resilient publish-subscribe for an information-centric internet of things. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 331–334. IEEE, 2018.

[22] Lukas Malina, Gautam Srivastava, Petr Dzurenda, Jan Hajny, and Radek Fujdiak. A secure publish/subscribe protocol for internet of things. In *Proceedings of the 14th international conference on availability, reliability and security*, pages 1–10, 2019.

[23] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35 (2):114–131, 2003.

[24] International Data Corporation (IDC). Worldwide global datasphere forecast, 2020–2024: The covid-19 data bump and the future of data growth. April 2020.

[25] Sheeraz A Alvi, Bilal Afzal, Ghalib A Shah, Luigi Atzori, and Waqar Mahmood. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, 33:87–111, 2015.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[27] Dong Yu and Li Deng. *Automatic Speech Recognition.* Springer, 2016.

[28] Julia Hirschberg and Christopher D Manning. Advances in natural language processing. *Science*, 349(6245):261–266, 2015.

[29] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[30] Hsinchun Chen, Roger HL Chiang, and Veda C Storey. Business intelligence and analytics: From big data to big impact. *MIS quarterly*, pages 1165–1188, 2012.

[31] Anne-Marie Kermarrec and Peter Triantafillou. Xl peer-to-peer pub/sub systems. *ACM Computing Surveys (CSUR)*, 46(2):1–45, 2013.

[32] Chen Chen, Yoav Tock, and Hans-Arno Jacobsen. Overlay design for topic-based publish/subscribe under node degree constraints. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 549–559. IEEE, 2016.

[33] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 219–227, 2000.

[34] Shelley Q Zhuang, Ben Y Zhao, Anthony D Joseph, Randy H Katz, and John D Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, 2001.

[35] Abhishek Gupta, Ozgur D Sahin, Divyakant Agrawal, and Amr El Abbadi. Meghdoot: content-based publish/subscribe over p2p networks. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 254–273. Springer, 2004.

[36] Asra Aslam and Edward Curry. Towards a generalized approach for deep neural network based event processing for the internet of multimedia things. *IEEE Access*, 6:25573–25587, 2018.

[37] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 57–70, 2016.

[38] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1493–1508, 2018.

[39] Dmitry Korzun, Ekaterina Balandina, Alexey Kashevnik, Sergey Balandin, and Fabio Viola. *Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities: Emerging Research and Opportunities.* IGI Global, 2019.

[40] Tarek Zaarour and Edward Curry. Adaptive filtering of visual content in distributed publish/subscribe systems. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–5. IEEE, 2019.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going

deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[45] Zan Gao, Hai-Zhen Xuan, Hua Zhang, Shaohua Wan, and Kim-Kwang Raymond Choo. Adaptive fusion and category-level dictionary learning model for multiview human action recognition. *IEEE Internet of Things Journal*, 6(6):9280–9293, 2019.

[46] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*, 2017.

[47] Tarek Zaarour and Edward Curry. Semanticpeer: A distributional semantic peer-to-peer lookup protocol for large content spaces at internet-scale. *Future Generation Computer Systems*, 2022.

[48] Tarek Zaarour, Anuurag Bhattacharya, and Edward Curry. Openpubsub: Supporting large semantic content spaces in peer-to-peer publish/subscribe systems for the internet of multimedia things. *IEEE Internet of Things Journal*, 2022.

[49] Stamatia Rizou, Frank Dürr, and Kurt Rothermel. Solving the multi-operator placement problem in large-scale operator networks. In *2010 Proceedings of 19th International Conference on Computer Communications and Networks*, pages 1–6. IEEE, 2010.

[50] Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik, and Robert Grimm. A catalog of stream processing optimizations. *ACM Computing Surveys (CSUR)*, 46(4):1–34, 2014.

[51] Wolfgang John, Konstantinos Pentikousis, George Agapiou, Eduardo Jacob, Mario Kind, Antonio Manzalini, Fulvio Risso, Dimitri Staessens, Rebecca Steinert, and Catalin Meirosu. Research directions in network service chaining. In *2013 IEEE SDN for future networks and services (SDN4FNS)*, pages 1–7. IEEE, 2013.

[52] Anastasios Kementsietsidis, Marcelo Arenas, and Renée J Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 325–336, 2003.

[53] Jianfeng Qian, Jianwei Yin, Jinxiang Dong, and Dongcai Shi. Jtangcsps: A composite and semantic publish/subscribe system over structured p2p networks. *Engineering Applications of Artificial Intelligence*, 24(8):1487–1498, 2011.

[54] Spyros Voulgaris et al. Epidemic-based self-organization in peer-to-peer systems. *These de doctorat, VU University, Amsterdam, Netherlands*, 2006.

[55] Souleiman Hasan and Edward Curry. Approximate semantic matching of events for the internet of things. *ACM Transactions on Internet Technology (TOIT)*, 14 (1):1–23, 2014.

[56] Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. S-topss: Semantic toronto publish/subscribe system. In *Proceedings 2003 VLDB Conference*, pages 1101–1104. Elsevier, 2003.

[57] Souleiman Hasan and Edward Curry. Thematic event processing. In *Proceedings of the 15th International Middleware Conference*, pages 109–120, 2014.

[58] Michael R Anderson, Michael Cafarella, German Ros, and Thomas F Wenisch. Physical representation-based predicate optimization for a visual analytics database. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1466–1477. IEEE, 2019.

[59] Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, and Hao Yang. Near-optimal algorithms for shared filter evaluation in data stream systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 133–146, 2008.

[60] Kamesh Munagala, Utkarsh Srivastava, and Jennifer Widom. Optimization of continuous queries with shared expensive filters. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 215–224, 2007.

[61] Antonio Carzaniga and Alexander L Wolf. Forwarding in a content-based network. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 163–174. ACM, 2003.

[62] Roberto Baldoni, Roberto Beraldi, Leonardo Querzoni, and Antonino Virgillito. Efficient publish/subscribe through a self-organizing broker overlay and its application to siena. *The Computer Journal*, 50(4):444–459, 2007.

[63] Michael A Jaeger, Helge Parzyjegla, Gero Mühl, and Klaus Herrmann. Self-organizing broker topologies for publish/subscribe systems. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 543–550, 2007.

[64] Chen Chen, Hans-Arno Jacobsen, and Roman Vitenberg. Divide and conquer algorithms for publish/subscribe overlay design. In *2010 IEEE 30th international conference on distributed computing systems*, pages 622–633. IEEE, 2010.

[65] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 109–118, 2007.

[66] Spyros Voulgaris, Etienne Riviere, Anne-Marie Kermarrec, Maarten Van Steen, et al. Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. In *IPTPS*, volume 141. Citeseer, 2006.

[67] Raphaël Chand and Pascal Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *European Conference on Parallel Processing*, pages 1194–1204. Springer, 2005.

[68] Fatemeh Rahimian, Sarunas Girdzijauskas, Amir H Payberah, and Seif Haridi. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 746–757. IEEE, 2011.

[69] Eli Fidler, Hans-Arno Jacobsen, Guoli Li, and Serge Mankovski. The padres distributed publish/subscribe system. In *FIW*, pages 12–30. Citeseer, 2005.

[70] Gianpaolo Cugola, Elisabetta Di Nitto, and Alfonso Fuggetta. The jedi event-based infrastructure and its application to the development of the opss wfms. *IEEE transactions on Software Engineering*, 27(9):827–850, 2001.

[71] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[72] Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen. G-topss: Fast filtering of graph-based metadata. In *Proceedings of the 14th international conference on World Wide Web*, pages 539–547, 2005.

[73] Opher Etzion and Peter Niblett. *Event processing in action.* Manning Publications Co., 2010.

[74] Ted Faison. *Event-Based Programming.* Springer, 2006.

[75] Brenda M Michelson. Event-driven architecture overview. *Patricia Seybold Group*, 2(12):10–1571, 2006.

[76] Hugh Taylor, Angela Yochem, Les Phillips, and Frank Martinez. *Event-driven architecture: how SOA enables the real-time enterprise*. Pearson Education, 2009.

[77] David Sprott and Lawrence Wilkes. Understanding service-oriented architecture. *The Architecture Journal*, 1(1):10–17, 2004.

[78] Jean-Louis Maréchaux. Combining service-oriented architecture and event-driven architecture using an enterprise service bus. *IBM developer works*, 12691275, 2006.

[79] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, 2002.

[80] D Luckham. Rapide: A language and toolset for simulation of distributed systems by partial ordering of events, 1996.

[81] Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, et al. The design of the borealis stream processing engine. In *Cidr*, volume 5, pages 277–289, 2005.

[82] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, et al. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 147–156, 2014.

[83] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[84] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[85] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. *ACM Sigmod Record*, 34(4):42–47, 2005.

[86] David C Luckham and Brian Frasca. Complex event processing in distributed systems. *Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford*, 28:16, 1998.

[87] Eugene Wu, Yanlei Diao, and Shariq Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418, 2006.

[88] Gero Muhl, Andreas Ulbrich, and K Herrman. Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*, 8(3):46–53, 2004.

[89] Kenneth P Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Transactions on Computer Systems (TOCS)*, 17(2):41–88, 1999.

[90] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE, 2008.

[91] Alan Sill. The design and architecture of microservices. *IEEE Cloud Computing*, 3(5):76–80, 2016.

[92] Joao Leitao, Jose Pereira, and Luis Rodrigues. Epidemic broadcast trees. In *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, pages 301–310. IEEE, 2007.

[93] Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure. In *International workshop on networked group communication*, pages 30–43. Springer, 2001.

[94] Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. Tera: topic-based event routing for peer-to-peer architectures. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 2–13, 2007.

[95] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. Lipsin: Line speed publish/subscribe inter-networking. *ACM SIGCOMM Computer Communication Review*, 39(4):195–206, 2009.

[96] Sébastien Baehni, Patrick Th Eugster, and Rachid Guerraoui. Data-aware multicast. In *International Conference on Dependable Systems and Networks, 2004*, pages 233–242. IEEE, 2004.

[97] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. The information bus: an architecture for extensible distributed systems. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68, 1993.

[98] Marcos K Aguilera, Robert E Strom, Daniel C Sturman, Mark Astley, and Tushar D Chandra. Matching events in a content-based subscription system. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 53–61, 1999.

[99] Satyen Kale, Elad Hazan, Fengyun Cao, and Jaswinder Pal Singh. Analysis and algorithms for content-based event matching. In *25th IEEE International Conference on Distributed Computing Systems Workshops*, pages 363–369. IEEE, 2005.

[100] Liangzhao Zeng and Hui Lei. A semantic publish/subscribe system. In *IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, pages 32–39. IEEE, 2004.

[101] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-wide web: the information universe. *Internet Research*, 1992.

[102] Dave Clark, Bill Lehr, Steve Bauer, Peyman Faratin, Rahul Sami, and John Wroclawski. Overlay networks and the future of the internet. *Communications and Strategies*, 63:109, 2006.

[103] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5): 50–58, 2002.

[104] Stefan Saroiu, Krishna P Gummadi, Richard J Dunn, Steven D Gribble, and Henry M Levy. An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36(SI):315–327, 2002.

[105] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 131–145, 2001.

[106] Gianpaolo Cugola and Gian Pietro Picco. Reds: a reconfigurable dispatching system. In *Proceedings of the 6th international workshop on Software engineering and middleware*, pages 9–16, 2006.

[107] Mark Astley, Joshua Auerbach, Sumeer Bhola, Gerard Buttner, Marc Kaplan, Kevan Miller, Robert Saccone Jr, Robert Strom, Daniel C Sturman, Michael J Ward, et al. Achieving scalability and throughput in a publish/subscribe system. Technical report, IBM Research Report RC23103 (W0402-026), 2004.

[108] Gianpaolo Cugola, Elisabetta Di Nitto, and Alfonso Fuggetta. Exploiting an event-based infrastructure to develop complex distributed systems. In *Proceedings of the 20th international conference on Software engineering*, pages 261–270. IEEE, 1998.

[109] Fengyun Cao and Jaswinder Pal Singh. Medym: Match-early with dynamic multicast for content-based publish-subscribe networks. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 292–313. Springer, 2005.

[110] Bill Segall and David Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. *Proceedings of the I997 Australian UNLX Users Group (A UUG'I997)*, pages 243–255, 1997.

[111] Emiliano Casalicchio and Federico Morabito. Distributed subscriptions clustering with limited knowledge sharing for content-based publish/subscribe systems. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pages 105–112. IEEE, 2007.

[112] Anton Riabov, Zhen Liu, Joel L Wolf, Philip S Yu, and Li Zhang. Clustering algorithms for content-based publication-subscription systems. In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 133–142. IEEE, 2002.

[113] Micah Adler, Zihui Ge, James F Kurose, Don Towsley, and Steve Zabele. Channelization problem in large scale data dissemination. In *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*, pages 100–109. IEEE, 2001.

[114] Ian Clarke, Scott G Miller, Theodore W Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[115] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings first international conference on peer-to-peer computing*, pages 99–100. IEEE, 2001.

[116] Bram Cohen. Bittorrent-a new p2p app. *Yahoo eGroups*, 2001.

[117] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[118] Ioannis Aekaterinidis and Peter Triantafillou. Pastrystrings: A comprehensive content-based publish/subscribe dht network. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 23–23. IEEE, 2006.

[119] Sriram Ramabhadran, Sylvia Ratnasamy, Joseph M Hellerstein, and Scott Shenker. Prefix hash tree: An indexing data structure over distributed hash tables.

In *Proceedings of the 23rd ACM symposium on principles of distributed computing*, volume 37. Citeseer, 2004.

[120] Joao Leitao. Topology management for unstructured overlay networks. *Technical University of Lisbon*, 2012.

[121] Ayalvadi J Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *International Workshop on Networked Group Communication*, pages 44–55. Springer, 2001.

[122] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-man: Gossip-based fast overlay topology construction. *Computer networks*, 53(13):2321–2339, 2009.

[123] Spyros Voulgaris, Daniela Gavidia, and Maarten Van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and systems Management*, 13(2):197–217, 2005.

[124] Joao Leitao, José Pereira, and Luis Rodrigues. Hyparview: A membership protocol for reliable gossip-based broadcast. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 419–429. IEEE, 2007.

[125] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 14–25, 2007.

[126] Miguel Matos, Ana Nunes, Rui Oliveira, and José Pereira. Stan: exploiting shared interests without disclosing them in gossip-based publish/subscribe. In *IPTPS*, page 9, 2010.

[127] Jay A Patel, Étienne Rivière, Indranil Gupta, and Anne-Marie Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 53(13):2304–2320, 2009.

[128] Felix Wortmann and Kristina Flüchter. Internet of things. *Business & Information Systems Engineering*, 57(3):221–224, 2015.

[129] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[130] Sheeraz A Alvi, Bilal Afzal, Ghalib A Shah, Luigi Atzori, and Waqar Mahmood. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, 33:87–111, 2015.

[131] Piyush Yadav, Dhaval Salwala, and Edward Curry. Vid-win: Fast video event matching with query-aware windowing at the edge for the internet of multimedia things. *IEEE Internet of Things Journal*, 2021.

[132] Edward Curry, Dhaval Salwala, Praneet Dhingra, Felipe Arruda Pontes, and Piyush Yadav. Multimodal event processing: A neural-symbolic paradigm for the internet of multimedia things. *IEEE Internet of Things Journal*, 2022.

[133] Hyung-Jun Yim, Dongmin Seo, Hanmin Jung, Moon-Ki Back, InA Kim, and Kyu-Chul Lee. Description and classification for facilitating interoperability of heterogeneous data/events/services in the internet of things. *Neurocomputing*, 256:13–22, 2017.

[134] Katia P Sycara. Multiagent systems. *AI magazine*, 19(2):79–79, 1998.

[135] Asra Aslam and Edward Curry. A survey on object detection for the internet of multimedia things (iomt) using deep learning and event-based middleware: approaches, challenges, and future directions. *Image and Vision Computing*, 106: 104095, 2021.

[136] Souleiman Hasan and Edward Curry. Thingsonomy: Tackling variety in internet of things events. *IEEE Internet Computing*, 19(2):10–18, 2015.

[137] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[138] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[139] Dmitry Korzun and Sergey Balandin. A peer-to-peer model for virtualization and knowledge sharing in smart spaces. In *Proceedings of the 8th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2014)*, pages 87–92, 2014.

[140] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[141] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1): 30–39, 2017.

[142] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[143] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*, pages 169–186. Springer, 2014.

[144] Edward Curry, Fredrik Heintz, Morten Irgens, Arnold WM Smeulders, and Stefano Stramigioli. Partnership on ai, data, and robotics. *Communications of the ACM*, 65(4):54–55, 2022.

[145] Liang-Chi Chiu, Tian-Sheuan Chang, Jiun-Yen Chen, and Nelson Yen-Chung Chang. Fast sift design for real-time visual feature extraction. *IEEE Transactions on Image Processing*, 22(8):3158–3167, 2013.

[146] Olga Russakovsky, Jia Deng, Jonathan Krause, Alex Berg, and Li Fei-Fei. The imagenet large scale visual recognition challenge 2012 (ilsvrc2012), 2012.

[147] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[148] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.

[149] Yannis Kalantidis, Lyndon Kennedy, and Li-Jia Li. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 105–112, 2013.

[150] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[151] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[152] Sriram Ramabhadran, Sylvia Ratnasamy, Joseph M Hellerstein, and Scott Shenker. Brief announcement: prefix hash tree. In *Proceedings of the twenty-third*

annual ACM symposium on Principles of distributed computing, pages 368–368, 2004.

[153] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[154] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[155] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[156] Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, 8(3): 382–439, 1976.

[157] George A Miller. *WordNet: An electronic lexical database.* MIT press, 1998.

[158] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[159] Alena Lukasová. Hierarchical agglomerative clustering procedure. *Pattern Recognition*, 11(5-6):365–381, 1979.

[160] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[161] Raymond Austin Jarvis and Edward A Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on computers*, 100 (11):1025–1034, 1973.

[162] Françoise Fabret, H Arno Jacobsen, François Llirbat, João Pereira, Kenneth A Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *ACM Sigmod Record*, 30(2):115–126, 2001.

[163] Shiyou Qian, Jian Cao, Yanmin Zhu, and Minglu Li. Rein: A fast event matching approach for content-based publish/subscribe systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2058–2066. IEEE, 2014.

[164] Dongxiang Zhang, Chee-Yong Chan, and Kian-Lee Tan. An efficient publish/subscribe index for e-commerce databases. *Proceedings of the VLDB Endowment*, 7 (8):613–624, 2014.

[165] Shiyou Qian, Jian Cao, Yanmin Zhu, Minglu Li, and Jie Wang. H-tree: An efficient index structurefor event matching in content-basedpublish/subscribe systems. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1622–1632, 2014.

[166] Qing Xie, Xiangliang Zhang, Zhixu Li, and Xiaofang Zhou. Optimizing cost of continuous overlapping queries over data streams by filter adaption. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1258–1271, 2016.

[167] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35: 99–109, 1943.

[168] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[169] Rui Zhu, Bang Liu, Di Niu, Zongpeng Li, and Hong Vicky Zhao. Network latency estimation for personal devices: A matrix completion approach. *IEEE/ACM Transactions on Networking*, 25(2):724–737, 2017.

[170] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[171] Navneet Kumar Pandey, Kaiwen Zhang, Stéphane Weiss, Hans-Arno Jacobsen, and Roman Vitenberg. Distributed event aggregation for content-based publish/-subscribe systems. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 95–106. ACM, 2014.

[172] Tarek Zaarour and Edward Curry. Semanticpeer: A distributional semantic peer-to-peer lookup protocol for large content spaces at internet-scale. *Future Generation Computer Systems*, submitted July 2021 (under review).

[173] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[174] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[175] Ioannis Aekaterinidis and Peter Triantafillou. Internet scale string attribute publish/subscribe data networks. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 44–51, 2005.

[176] Muhammad Adnan Tariq, Gerald G Koch, Boris Koldehofe, Imran Khan, and Kurt Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *European Conference on Parallel Processing*, pages 458–470. Springer, 2010.

[177] Yutaka Ohsawa and Masao Sakauchi. A new tree type data structure with homogeneous nodes suitable for a very large spatial database. In *Proceedings. Sixth International Conference on Data Engineering*, pages 296–297. IEEE Computer Society, 1990.

[178] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2):170–231, 1998.

[179] Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 175–186, 2003.

[180] Mei Li, Wang-Chien Lee, and Anand Sivasubramaniam. Semantic small world: An overlay network for peer-to-peer search. In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, pages 228–238. IEEE, 2004.

[181] Wilma Penzo, Stefano Lodi, Federica Mandreoli, Riccardo Martoglia, and Simona Sassatelli. Semantic peer, here are the neighbors you want! In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 26–37, 2008.

[182] Muhammad Adnan Tariq, Boris Koldehofe, Gerald G Koch, and Kurt Rothermel. Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 213–224, 2012.

[183] Federica Mandreoli, Riccardo Martoglia, Simona Sassatelli, and Wilma Penzo. Sri: exploiting semantic information for effective query routing in a pdms. In *Proceedings of the 8th annual ACM international workshop on Web information and data management*, pages 19–26, 2006.

[184] Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. S-topss: Semantic toronto publish/subscribe system. In *Proceedings 2003 VLDB Conference*, pages 1101–1104. Elsevier, 2003.

[185] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[186] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[187] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[188] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[189] heikoheiko. Simulation of a peer-2-peer network using simpy. https://github.com/heikoheiko/p2p-network-simulator.

[190] Angelos Stavrou, Dan Rubenstein, and Sambit Sahu. A lightweight, robust p2p system to handle flash crowds. *IEEE Journal on Selected Areas in Communications*, 22(1):6–17, 2004.

[191] Emmanuelle Anceaume, Maria Gradinariu, Ajoy Kumar Datta, Gwendal Simon, and Antonino Virgillito. A semantic overlay for self-peer-to-peer publish/subscribe. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 22–22. IEEE, 2006.

[192] Dmitry Korzun and Andrei Gurtov. *Structured peer-to-peer systems: fundamentals of hierarchical organization, routing, scaling, and security*. Springer Science & Business Media, 2012.

[193] Michael E Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *International conference on scientific and statistical database management*, pages 482–500. Springer, 2010.

[194] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer, 2004.

[195] Sinan Saraçli, Nurhan Doğan, and İsmet Doğan. Comparison of hierarchical cluster analysis methods by cophenetic correlation. *Journal of inequalities and Applications*, 2013(1):1–8, 2013.

[196] commonsense. ConceptNet Numberbatch pre-computed word embeddings. https://github.com/commonsense/conceptnet-numberbatch.

[197] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[198] Ming-Hung Hsu, Ming-Feng Tsai, and Hsin-Hsi Chen. Query expansion with conceptnet and wordnet: An intrinsic comparison. In *Asia Information Retrieval Symposium*, pages 1–13. Springer, 2006.