A Survey on Design and Application of Publish-Subscribe System

HAO Shuai
Department of Computer Science
The College of William and Mary

December 9, 2011

Abstract

The Publish-Subscribe (Pub-Sub) System, as a distributed messaging paradigm, is a classic subject on distributed computing system. In recent years, with the rise and mature of Cloud Computing and Wireless Sensor Networks in both academia and industry, the Publish-Subscribe System has been also receiving the increased attention. The dynamic, asynchronous, loose-coupling and multipoint-communications features enable the Pub-Sub system to serve as an effective communication paradigm in large-scale distributed systems and applications. In this survey, we present the basic models, frameworks and components of Publish-Subscribe system, and investigate several existing Pub-Sub systems. Also, we pay attention to the emerging application of Pub-Sub System in large-scale networking and mobile environment, such as Ad-Hoc networks and Wireless Sensor Networks. Finally, we summarize some research problems and challenges, and also discuss some potential research topics in this area.

1 Introduction

The Publish-Subscribe system is a massaging paradigm in distributed computing. The Publishers publish the events, which are defined as the interaction information between sender (*publisher*) and receiver (*subscriber*), and the subscribers, who subscribe to the publishers, will receive the events they are interested in. The subscribers are not directly targeted from the publishers: they express their interest by subscribing to specific information classes (called *notifications*). The publishers independently produce and distribute information. The Pub-Sub middleware is to ensure that the producer's releases are reliably delivered to all of the interested subscribers.

A typical Publish-Subscribe system includes topology structure, event model, subscription model, matching algorithm, routing system and other QoS or security components. The topology structure determines the scalability of system. The Event Model defines the event data structure; the subscribe model defines the subscription conditions that the system can support, which point out the subscribers how to express interest in a subset of the event. The matching algorithm is responsible for efficiently finding and matching the events to subscribers interests; and routing system is responsible for selecting the appropriate path, and sending the events from the publishers to the subscribers.

The rest of this report is organized as follows. In Section 2, we will introduce the Publish-Subscribe system architecture based on the subscription model, including Topic-based Pub-Sub, Content-based Pub-Sub and Type-based Pub-Sub. In section 3, we explore the design of Publish-Subscribe system based on the system components, including notification service, matching algorithms, Routing Algorithms, and QoS and security design. We also investigate some existing Publish-Subscribe implementations in this section. In Section 4, we present the new developments and achievements of Pub-Sub system in recent years over large-scale networking and mobile environment. We conclude this report in Section 5.

2 Overview

The objective of a publish-subscribe system is to let information propagate from publishers to interested subscribers by an anonymous and decoupled model. In this section we first present a general, high-level specification of a pub-sub communication system, then we

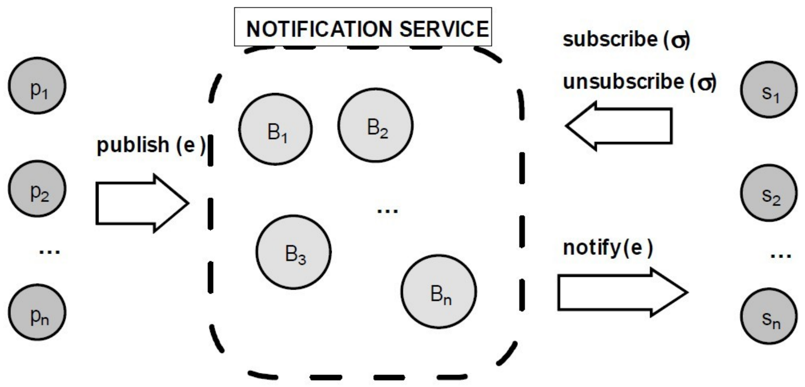


Figure 1: High Level View of Pub-Sub System (from [20])

survey the state of the art in this research field by investigating the different subscription models.

2.1 Publish-Subscribe Models

A generic Publish-Subscribe communication system can be represented by a triple { Π, Δ, Σ } of sets of processes. The Figure 1 shows a generic model of Publish-Subscribe System [20]. These processes sets in the triple are defined depending on the function of the processes in a Pub-Sub system:

$\Pi = p_1, \ldots , p_n$ is a set of n processes, called the publishers, which are producers of information;

$\Sigma = s_1, \ldots , s_m$ is a set of m processes, called the subscribers, which are consumers of information;

$\Delta = B_1, \ldots ,B_n$ is a set of o processes, called brokers.

The Π and Σ may have a non-zero intersection, which is a same process may act both as a publisher and as a subscriber. A process in Π cannot communicate directly with a process in Σ, and vice versa. A process in Π and Σ can exclusively communicate with any other process in Δ. The set of brokers Δ serves as a logical entity (or middleware) to allow the communication between publishers and subscribers. The set of Δ is referred to as *Notification Service* (or Event Service). Publishers and subscribers act as clients for the Notification Service.

The common data model used in Publish-Subscribe systems defines a notification as a set of *attribute-value* pairs. The subscribers' interest in specific notifications is expressed through the *subscriptions.* A subscription is a pair $\sigma = (f, s)$. The $s \in \Sigma$ is the subscriber and its interest is declared through the filter *f.* The process of verifying whether a notification satisfies a filter f is called *matching.* The Matching algorithms will be presented in Section 3.1.2.

The subscribers are usually interested in particular event classes, not all notifications. The different ways of specifying the events of interest have led to different publish-subscribe paradigms [7]. In this section, we investigate three main subscription-based categories in Publish-Subscribe system: topic-based Pub-Sub, content-based Pub-Sub and type-based Pub-Sub. In this part, we will discuss their basic framework and also the advantages and disadvantages for each model.

2.2 Topic-based Model

The notifications in a topic-based model (also subject-based model) are grouped in topics (or subjects), and the subscribers will receive all notifications related to that specific topics, which can be identified by *keywords.* In other words, the filter of a subscription is simply the specification of a topic.

Each topic class will define a logical *event channel* from each publisher to all interested subscribers. The subscribers subscribe to a topic T means a user is becoming a member of group T, and publishing an event on topic T means a user becoming a publisher for topic

T. The system maintains a static association between a channel and all its subscribers. The topics can be organized by a hierarchical structure to enable the containment relationships of topics. Also, the topic names can contain the *wildcards* to offer the possibility to subscribe and publish to several topics by matching a given set of keywords.

Obviously, the topic-based model provides a simple and neat framework for Pub-Sub system. However, the main drawback of the topic-based model is the very limited expressiveness. The subscribers have to receive all notifications from a topic group even if they are interested in certain specific criteria, which leads to an inefficient use for system resources and bandwidth.

2.3 Content-based Model

In the content-based Publish-Subscribe systems, the subscribers subscribe to the event by specifying their interests on the contents of notification. In this context, the events are not limited to any specific group or class; the decision where an event will point to is expressed by the constraints and attributes of subscribers' queries. That is, a filter in a subscription is a query composed by a conjunction of constraints over the values of attributes of the notification; each subscriber only receives the notifications that match entirely its individual criteria by specifying the properties of the event notifications they are interested in.

The content-based Publish-Subscribe system has the advantage of flexibility. The subscribers do not need to know the definition of the topic names before the subscription. The disadvantage of this model is that the system has to match a large number of events. The number of events is much larger than the topic-based system, so the matching must be highly efficient.

The content-based model is a most common subscription framework for Pub-Sub system. It offers an appropriate trade-off on function and complexity.

2.4 Type-based Model

The type-based model [6] [8] enhances common Pub-Sub with concepts derived from object-oriented programming: the notifications are declared as objects belonging to a specific type (called *obvents*), which can thus encapsulate attributes as well as methods. The events are classified by the type of objects, and the conditions of subscription are encapsulated in subscription objects. The subscriber of a particular type of objects will only receive instances of that type and its subtypes.

The major advantage for type-based scheme is in the simplicity and flexibility of decentralization implementation to support a large number of client and huge amount of data transfers. Also, this model offers the scalability by use of remote content filtering.

3 Design and Implementation of Publish-Subscribe Subsystem

In this section we will focus on the design and implementation issues by discussing several important subsystems of Publish-Subscribe system, including notification services, matching algorithms, Routing, Quality-of-Service and security design issues. Also, we will briefly introduce several typical existing Pub-Sub system implementations.

3.1 System Components

3.1.1 Notification Service

In this part we present the basic component of pub-sub system, the architecture of distributed Event Notification Service. The interaction between publisher and subscriber

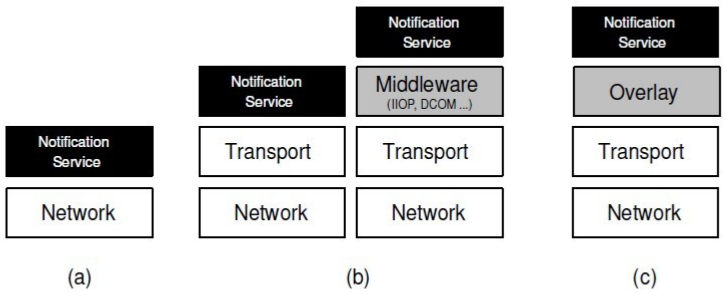


Figure 2: Notification Service Architecture(from [20])

relies on the event notification service providing storage and management for subscriptions and efficient delivery of events. Such an event service represents a neutral mediator between publishers, acting as producers of events, and subscribers, acting as consumers of events.

There are basically three solutions will be discussed in this section: Multicasting (Fig.2(a)), Application-level Network (Fig.2(b)) and P2P overlay network (Fig.2(c)):

Multicasting The Network-layer multicasting is mostly used by topic-based system as each topic corresponds exactly to one multicast group [15]. The main advantage of a network-level approach is that it is the easier way to realize many-to-many diffusion experiencing low latencies and high throughput. However, the main drawback of multicast when applied to wide-area scenarios is in its lack of a widespread deployment. Hence, network-level multicasting cannot in general be considered as a feasible solution for applications deployed over the WAN.

Application-level Network An application-level network of brokers is the most common approach for a distributed Notification Service. Each broker communicates with its neighbor for subscription and publication using TCP/IP or HTTP. This solution is more scalable even if the size grows as the communications in the notifications component is through new neighbor brokers. Messaging involves use of concurrent connections and sets of data structures for matching subscriber interest with information from publishers brokers. The main disadvantage of this method is in a much slower performance as the implementations of notifications systems is through an application layer. The connections among the brokers are not fully automated and may require some active support from user to set up correct connections between any 2 or more brokers.

P2P overlay network An overlay network is a logical application-level network that is built on top of a general network layer. The nodes that are part of the overlay network can route messages between each other through the overlay network.

Structuring a Publish-Subscribe system [3] [4] [23] [14] over an overlay network infrastructure means leveraging the self-organization capabilities of the infrastructure, by building a Pub-Sub interface over it. The Pub-Sub behavior is realized through the communication primitives provided by the underlying overlay. In the peer-to-peer model of Pub/Sub system , all nodes are equal, where each node can act as a publisher, subscriber, root of a multicast tree, or internal node of a multicast tree. That is there are no server nodes or client nodes in this model. Some of the server functionality (i.e. persistence, transactions, security) is embedded as a local part of each node.

3.1.2 Matching Algorithm

The matching algorithm in Publish-Subscribe system is responsible for efficient to find a given event with all the subscriptions that match the conditions of subscription. That is, it is the process that satisfies the filter f defined in section 2.1. The matching algorithm is always dependent on the specific data model.

In the topic-based systems, the messages are grouped into logical units in the form of a group, channel, or topic. Thus, the purpose of the matching algorithm is to manage these channels. To manage these topics or channels, a centralized server can be used to lookup the topic ID for the arriving messages to determine the connection information for the subscribers.

In the Content-based systems, there is no notion for groups or topics. Hence these systems require more complex approaches of matching messages to subscribers. Many content-based systems use a *matching tree* algorithm. This algorithm preprocesses a set of subscriptions into a matching tree. Each node serves as a partial condition on the attributes of a predicate. Each lower level of the tree is a refinement of the test performed at the next higher level, and the leaves of tree represent the subscriptions. Upon arrival of an event, the subscriptions matching the event are found by navigating the decision tree starting at the root.

3.1.3 Routing

The routing algorithm in Publish-Subscribe system is to solve how to find an appropriate path in event broker network to deliver the events and notifications to the relevant subscribers. In this part we discuss two main routing schemes that are determined by the *subscription assignment polices.*

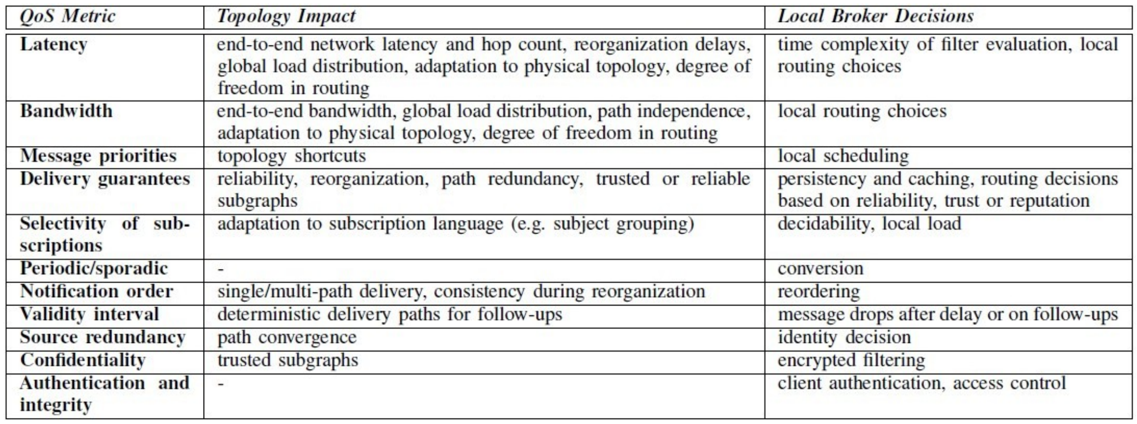
Subscription Assignment The subscription assignment is the identification of a policy used to determine how to distribute the subscriptions among the brokers. The Strategies for subscription assignment can be summarized in the following two categories: 1) *Access-Driven Assignment (ADA)* The subscription is stored in the access points of the subscriber. In this case access points and target brokers coincides for the subscriber. In other words, the assignment depends only on the subscriber, not considering filter; 2) *Filter-Driven Assignment (FDA)* The subscription filters are partitioned into a set of clusters. Each subscription is assigned to a different broker depending on the cluster it is assigned to.

The Clusters are determined according to subscriptions' filters. That is, the assignment considers only filter, independently from subscribe.

Subscription Routing and Event Routing Before discussing the routing strategies, we first distinguish the concepts on *subscription routing* and *event routing*. The subscription routing is the process of dispatching the subscription from the access point to the target broker. The event routing is the process of identifying the target brokers for the notification and forwarding the notification through the broker network in order to reach all possible target brokers.

Routing Strategies for Access-Driven Assignment Under the ADA approach, event routing and notification routing coincide as the target broker for the subscription. The trivial solution is to broadcast each notification to all the brokers (flooding). This method obviously leads to an undesirable waste of network resources because all notifications are always sent to all brokers even if they do not match any of their subscribers.

Routing Strategies for Filter-Driven Assignment Under the FDA policy, event routing and notification routing have two distinct phases. First the target broker is reached by the event, and then the notification is delivered to all matching subscribers. The brokers that have to be reached by a notification in both phases can be determined in advance: the target broker with the resolve function and the matching subscribers' access points from the matching operation itself. Then, event and notification routing reduces to routing to some specific brokers. This can be easily implemented either at application-level or overlay network infrastructure. Another possible approach relies on an external centralized notification routing service [22]. The brokers maintain subscriptions and perform matching, and the notification service maintains all the references to subscribers.



QoS Metric	*Topology Impact*	*Local Broker Decisions*
Latency	end-to-end network latency and hop count, reorganization delays, global load distribution, adaptation to physical topology, degree of freedom in routing	time complexity of filter evaluation, local routing choices
Bandwidth	end-to-end bandwidth, global load distribution, path independence, adaptation to physical topology, degree of freedom in routing	local routing choices
Message priorities	topology shortcuts	local scheduling
Delivery guarantees	reliability, reorganization, path redundancy, trusted or reliable subgraphs	persistency and caching, routing decisions based on reliability, trust or reputation
Selectivity of subscriptions	adaptation to subscription language (e.g. subject grouping)	decidability, local load
Periodic/sporadic	-	conversion
Notification order	single/multi-path delivery, consistency during reorganization	reordering
Validity interval	deterministic delivery paths for follow-ups	message drops after delay or on follow-ups
Source redundancy	path convergence	identity decision
Confidentiality	trusted subgraphs	encrypted filtering
Authentication and integrity	-	client authentication, access control

Figure 3: Relation of topologies and local broker decisions to the QoS metrics: Impacts and implementation choices(from [1])

3.1.4 Quality of Service

The publish-subscribe model suggests a number of different quality-of-service metrics [1], briefly summarized in Figure 3. Some are related to subscriptions and single notifications while others describe end-to-end prop-erties of flows of notifications. Each implementation in a distributed publish-subscribe system has mainly two dimensions: The topological organization of the brokers and the local decisions of each broker.

On the other hand, the guarantees provided by the medium for every message vary strongly between the different systems. Among the most common qualities of service considered in Pub-Sub system, we mainly consider the persistence, transactional guarantees, and priorities [7] [12].

Persistence The persistence is generally present in centralized Publish-Subscribe systems where the messages are stored until consumers are able to process them. In that situation, the communicating parties do not control how messages are transmitted and when they are processed. Thus, the messaging system must provide guarantees not only in terms of reliability, but also in terms of durability of the information.

Transactional Guarantees Transaction guarantees are generally used to group multiple operations in atomic blocks, where the transactions are used to group messages and either completely executed or not executed at all.

Priorities The priorities are desirable to sort the messages waiting to be processed by a consumer in order of priority. For example, a real-time event may require immediate reaction and should be processed before other messages.

3.1.5 Security Issues

A publish-subscribe system must handle information dissemination across distinct authoritative domains, heterogeneous platforms and a large, dynamic population of publishers and subscribers. Such environments raise new security concerns [21].

Authentication Authentication establishes the identity of the originator of an action. In Pub-Sub system, we consider two flavors of authentication, end-to-end and point-to-point. The end-to-end authentication verifies the original publisher of the message and the point-to-point verifies the sender of message, where the sender can be publishers, subscribers or network servers. The end-to-end can be implemented outside of the Pub-Sub domain, like PKI.

Confidentiality When information being published contains sensitive content, publishers and subscribers may wish to keep information secret from the pub-sub infrastructure. Also, user subscriptions may reveal sensitive information about the user, but the subscriber may wish to keep the subscriptions private.

Integrity The information integrity can be implemented by the digital signature. When signed on the message digest with the senders private key, it means that the message content has not been changed since it is signed and the message indeed originated from the sender.

Accountability In large-scale system, there may be no direct relationship between a publisher and subscriber. Further, a publisher has no way of knowing which subscribers receive particular datagrams. Therefore, a solution to accountability is necessary for Pub-Sub system. The publishers may bill subscribers by selling keys that decrypted data. Also, the Pub-Sub infrastructure can bill subscribers according to the amount of information they receive and pay publishers according to the information they provide without there being any direct relationship between publishers and subscribers.

The concept of scopes [9] helps to localize and implement security policies as an aspect of structured Publish-Subscribe systems. The scopes in Pub-Sub systems [11] delimit the groups of producers and consumers on the application level and control the dissemination of notifications within the infrastructure. That is, the fundamental idea of the scoping concept is to control the visibility of notifications outside of application components and orthogonal to their subscriptions.

4 Applications

The Internet has considerably changed the scale of distributed systems. Distributed systems now involve thousands of entities and their location and behaviors may vary throughout the lifetime of the system. These constraints motivate the demand for more flexible communication models and systems that reflects the dynamic and decoupled nature of the applications. Also, the rise of mobile Internet, Ad-Hoc Networks and Wireless Sensor Networks has been inspired more requirements for an asynchronous, loose-coupling and multipoint-communications messaging paradigm. Therefore, in this section, we survey some emerging Publish-Subscribe systems for large-scale networking and mobile environment.

4.1 Publish-Subscribe-Based Large-Scale Networking

4.1.1 SCRIBE

The *SCRIBE* [2] is a scalable, decentralized subject-based system built on the top of Pastry [17], which is a generic P2P object location and routing substrate overlay on the Internet. Each broker in Pastry is assigned a unique identifier in the network and messages can be routed to a specific broker by simply specifying its identifier. The SCRIBE implements actually an Pub-Sub interface using Pastry.

In SCRIBE, a multicast tree is built for each topic, rooted at the corresponding target broker. When a new node subscribes for a topic, its subscription is routed by Pastry to the corresponding target broker. When an event is published for a subject, event routing is performed through Pastry by directly routing the event to the target broker. In this process, the target broker is simply addressed by the topic ID.

4.1.2 RTFM

The *RTFM* [19] is a packet-switch internetworking framework based on the Publish-Subscribe paradigm. The key idea is to distribute control over data reception from network endpoints to the network itself. Different from the previous Pub-Sub proposals which are constructed on top of IP protocol or application-level multicast, the RTFM Pub-Sub network gives each host only two primitives: publish and subscribe. When a node subscribes to a named piece of data, the network will create a delivery path from the publisher to subscriber.

In RTFM architecture, two identifiers are created by publications: a *private identifier* and a *public identifier*. The private identifier is known only by publisher, and the public identifier is used to receive and verify the publication by subscriber. The forwarding of RTFM is implemented by the label switching. A label is a bit string in the packet that is used by the routers to make forwarding decisions which can be the same as publication identifier.

4.1.3 Corona

The *Corona* [16] is a topic-based Publish-Subscribe system for the World Wide Web. The Corona provides high-performance and scalability through optimal resource allocation.

In Corona system, the URLs serve as topics or channels in Pub-Sub system and the users register their interest in Web content and receive updates asynchronously by the URLs of Web pages. The Corona monitors the subscribed Web pages, detects updates and disseminates it to users by cooperatively polling the content servers from geographically distributed nodes. Allocation of resources is driven by a distributed optimization engine that achieves the best update performance without exceeding load limits on content servers. The key feature that enables Corona to achieve fast update detection is *cooperative polling*. Corona assigns multiple nodes to periodically poll the same channel and shares updates detected by any polling node.

4.2 Mobility Support for Publish-Subscribe System

4.2.1 Virtual Counterpart

The *virtual counterpart* [10] presents an approach to support mobility in existing publish/subscribe middleware. In this work, it provides support for two different and orthogonal types of mobility: 1) *Physical Mobility* In the model, the clients may temporarily disconnect from the Pub-Sub system due to power-saving requirements or the network characteristics. This means that applications are not necessarily aware of the fact that the client is moving. 2) *Logical Mobility* The client remains attached to the their broker.

The basic idea in this solution is to maintain a virtual counterpart of a roaming client at the last known location until some broker at a new location is claiming responsibility and then merge actual and virtual client in such a way that no notification is lost or delivered twice.

4.2.2 Mires

The *Mires* [18] incorporates characteristics of message-oriented middleware by allowing applications communicate in a Publish-Subscribe way for Sensor networks. The Mires implements a three-phases communication between nodes. Initially, the nodes in the network advertise their available topics collected from local sensors. Next, the advertised messages are routed to the sink node using multi-hop routing algorithm. A user application connected to the sink node is able to select the desired advertised topics to be monitored. Finally, subscribe messages are broadcasted down to the network nodes. After receiving the subscribed topics, nodes are able to publish their collected data to the network.

4.2.3 ShopParent

The *ShopParent* [13] propose a distributed greedy algorithm for Publish-subscribe tree (PST) in ad-hoc network by a fully distributed fashion. This solution precisely defines the optimality of a Publish/Subscribe tree by developing a metric to evaluate its efficiency. The optimality metric takes into account both the goal of a Publish-Subscribe system and the characteristics of ad-hoc network.

One node in the system is designated as root of the Publish-Subscribe mechanism. Only the root node can publish new events. The new event has to be generated at root, and then forwarded to other interested nodes. Assume that every node in the system participates in the Publish-Subscribe system. In particular, nodes that are not interested in any event will simply have an empty subscription. When a new event is published at the root node, the wireless nodes run the distributed Pub-Sub protocol to forward this event along the branches of the PST until it reaches all nodes that have subscribed to it. Every node listens for new events broadcast by its parent node in the tree. Then, depending on the event's content, some or all of a set of actions can be performed. Finally, forwarding an event involves calculating the set of children nodes that will be interested in the event.

4.2.4 SocialCast

The *SocialCast* [5] proposes a interest-based routing protocol framework for Publish-Subscribe system to support delay tolerant communication that exploits the predictions based on metrics of social interaction. This approach assumes that socially bound hosts are likely to be co-located regularly: these colocation patterns are then used to efficiently route the messages from publishers to interested subscribers.

The social ties selection is made by taking into account predictions about contextual. The main feature of SocialCast relies on the notion of utility for the selection of message carriers to enable store-and-forward communication. The utility of a node with respect to interest represents how good of a carrier is for messages matching. The utility values are linked to movement patterns and colocation with other hosts. As the basic assumption is that hosts which have same interest spend time co-located, the SocialCast routing aims at exploiting as carrier for messages hosts which have been co-located often with the interested subscribers.

5 Conclusion

The publish-subscribe paradigm is largely recognized as one of the most effective way to realize flexible and scalable large-scale distributed applications. In this survey paper, we investigate the models and mechanisms of Publish-Subscribe system, and the design and implementation issues for Pub/Sub. Also, we explore the emerging applications and systems for large-scale networking and mobile environment in recent years.

References

[1] Stefan Behnel, Ludger Fiege, and Gero Mühl. On quality-of-service and publish-subscribe. In *ICDCS Workshops*, page 20, 2006.

[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC*, 20:2002, 2002.

[3] Chen Chen, Hans-Arno Jacobsen, and Roman Vitenberg. Divide and conquer algorithms for publish/subscribe overlay design. In *ICDCS*, pages 622–633. IEEE Computer Society, 2010.

[4] Chen Chen, Roman Vitenberg, and Hans-Arno Jacobsen. Scaling construction of low fan-out overlays for topic-based publish/subscribe systems. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 225–236, Washington, DC, USA, 2011. IEEE Computer Society.

[5] Paolo Costa, Cecilia Mascolo, Mirco Musolesi, and Gian Pietro Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, 2008.

[6] Patrick Eugster. Type-based publish/subscribe: Concepts and experiences. *ACM Trans. Program. Lang. Syst.*, 29, January 2007.

[7] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35:114–131, June 2003.

[8] Patrick Th. Eugster, Rachid Guerraoui, and Christian Heide Damm. On objects and events. In *Proceedings of the 16th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '01, pages 254–269, New York, NY, USA, 2001. ACM.

[9] L. Fiege, A. Zeidler, A. Buchmann, R. Kilian-Kehr, G. Mhl, and Tu Darmstadt. Security aspects in publish/subscribe systems. In *In Third Intl. Workshop on Distributed Event-based Systems (DEBS04*. IEEE, 2004.

[10] Ludger Fiege, Felix C. Gärtner, Oliver Kasten, and Andreas Zeidler. Supporting mobility in content-based publish/subscribe middleware. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, pages 103–122, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[11] Ludger Fiege, Gero Mhl, and Felix C. Grtner. Modular event-based systems. *THE KNOWLEDGE ENGINEERING REVIEW*, 17:359–388, 2006.

[12] Shuo Guo, Kyriakos Karenos, Minkyong Kim, Hui Lei, and Johnathan Reason. Delay-cognizant reliable delivery for publish/subscribe overlay networks. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 403–412, Washington, DC, USA, 2011. IEEE Computer Society.

[13] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 122–140, London, UK, 2003. Springer-Verlag.

[14] Melih Onus and Andra W. Richa. Brief announcement: parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design. In Friedhelm Meyer auf der Heide and Michael A. Bender, editors, *SPAA*, pages 39–40. ACM, 2009.

[15] Lukasz Opyrchal, Mark Astley, Joshua Auerbach, Guruduth Banavar, Robert Strom, and Daniel Sturman. Exploiting ip multicast in content-based publish-subscribe systems. In *IFIP/ACM International Conference on Distributed systems platforms*, Middleware '00, pages 185–207, Secaucus, NJ, USA, 2000. Springer-Verlag New York, Inc.

[16] Venugopalan Ramasubramanian, Ryan Peterson, and Emin Gün Sirer. Corona: a high performance publish-subscribe system for the world wide web. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 2–2, Berkeley, CA, USA, 2006. USENIX Association.

[17] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IN: MIDDLEWARE*, pages 329–350, 2001.

[18] Eduardo Souto, Germano Guimar&#x00e3;es, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz, and Judith Kelner. Mires: a publish/subscribe middleware for sensor networks. *Personal Ubiquitous Comput.*, 10:37–44, December 2005.

[19] Mikko Srel, Teemu Rinta-aho, and Sasu Tarkoma. Rtfm: Publish/subscribe internetworking architecture. *ICT Mobile Summit*, pages 1–8, 2008.

[20] Antonino Virgillito. *Publish/Subscribe Communication Systems: From Models to Applications.* PhD thesis, Universita La Sapienza, 2003.

[21] C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9*, HICSS '02, pages 303–, Washington, DC, USA, 2002. IEEE Computer Society.

[22] Yi-Min Wang, Lili Qiu, Dimitris Achlioptas, Gautam Das, Paul Larson, and Helen J. Wang. Subscription Partitioning and Routing in Content-based Publish/Subscribe Networks. In D.Malkhi, editor, *Distributed algorithms*, volume 2508/2002 of *Lecture Notes in Computer Science*, October 2002.

[23] Young Yoon, Vinod Muthusamy, and Hans-Arno Jacobsen. Foundations for highly available content-based publish/subscribe overlays. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 800–811, Washington, DC, USA, 2011. IEEE Computer Society.