Evaluating the Robustness of Publish/Subscribe Systems

Tobias R. Mayer, Lionel Brunie
Laboratoire LIRIS
INSA de Lyon
Lyon, France
Email: {tobias-rene.mayer, lionel.brunie}@insa-lyon.fr

David Coquil, Harald Kosch
Department of Distributed Information Systems
University of Passau
Passau, Germany
Email: {david.coquil, harald.kosch}@uni-passau.de

***Abstract*—In recent years, publish/subscribe (pub/sub) systems have evolved as serious candidates for the implementation of multicast communication. However, to date they have not been properly analysed with respect to the critical dimension of robustness. Indeed, the robustness evaluations found in the literature only cover a subset of the types of failure situations and generally lack a comprehensive architectural classification. In this context, this paper introduces the necessary foundations for systematic evaluation of the robustness of pub/sub system: a comprehensive classification scheme and a complete taxonomy of failures. We argue that both robustness evaluation of pub/sub systems and development methods of robust pub/sub systems should take rational behavior into account. We demonstrate this point through an experimental study.**

***Keywords*-publish/subscribe; application-layer multicast; fault-tolerance; rational behaviour; robustness**

I. Introduction

Group communication has been available for several years in IP networks by means of IP multicast. However, until now this opportunity has only been scarcely used. Among other reasons, this is due to the frequent omission of the IP multicast stack in modern routers in favor of speed and to the lack of expressiveness of subscriptions for notifications of interest. The publish/subscribe (pub/sub) paradigm was proposed as a means to overcome these drawbacks by providing application-level group communication services. Over the years, it has evolved into a viable middleware option for event-based multicast communication. It is indeed used in commercial products, such as the Microsoft BizTalk Server 2004 or in the communication middleware of NetAcquire. When implemented in a large scale, robustness concerns become critical in such systems. Indeed, the probability of nodes being affected by Byzantine failures increases strongly [1]. As soon as the system spans over multiple administrative domains, direct administrative control may be lost, and other administrators need to be trusted. Indeed, they may want to change the behaviour of nodes in their domains for their own benefit. Moreover, users may as well modify their nodes to maximize their gain from the system. Such selfish or rational behaviour may be considered as an attack if it breaks the system policy and puts global performance at risk [2]. In a large-scale environment, these threats cannot be completely eliminated. This emphasizes the need for *robust* systems able to tolerate not only Byzantine failures, but also to maintain system goals in the presence of rational nodes. This property is known as BAR tolerance [3].

Hence, from a practical point of view, a person responsible for the communication layer of a P2P system could ask the following questions:

(i) *How robust are current systems? How can their robustness be evaluated?*

To answer it two other questions need to be clarified before:

(ii) *What types of pub/sub systems are available and in which aspects do they differ?*

(iii) *What types of faults can affect them?*

Despite much research effort in pub/sub systems in recent years, these questions remain difficult to answer for several reasons. Available system types are manifold and surveys or other overviewing works have been done focusing on specific architectural or functional aspects. The result is that they consider only a subset of architectural dimensions and may differ in terminology. We argue that they are not adequate for a classification from comprehensive point of view, which is not yet available for pub/sub systems. This makes it hard to perform meaningful architectural comparisons and to draw conclusions about the robustness of the different system types. In a similar way, existing studies of faults in pub/sub systems cannot fully answer the third question, mostly because they do not cover the full spectrum of faults in pub/sub systems.

This paper intends to face these problems by forming a common base for further evaluations. Furthermore we show that these evaluations should also comprise BAR tolerance capabilities. Hence, this paper makes three contributions:

1) A general classification scheme for pub/sub systems, that identifies architectural dimensions.
2) A taxonomy of Byzantine faults, specially tailored to pub/sub systems. The taxonomy classifies atomic faults from the point of view of a peer.
3) A simulative study, that shows exemplary the worst-case impact of rational behaviour for one system and fault type.

The remainder of this paper is organized as follows. In section II, we discuss related work. The general classification

scheme for pub/sub systems is presented in section III. We outline possible faults and failure situations in section IV by providing a taxonomy of Byzantine faults for pub/sub. The following section V discusses rational behaviour and outlines its harmful influence on pub/sub systems an experimental study. Finally, section VI provides a conclusion and discusses future work.

II. Related Work

Publish/subscribe has attracted much research effort over the last decade, leading to a large number of heterogeneous systems. Carzaniga et al. introduced with SIENA [4] one of the first systems. The first systems followed a deterministic structure and targeted mainly core aspects such as subscription type (e.g. topic-, content-based) or overlay organisation. Later on, unstructured approaches introduced systems with non-deterministic dissemination schemes (e.g. [5], [6]). Among general architectural developments, researchers focused also on other aspects such as scalability [7], reliability [8] or quality of service [9]. Worth to mention is also the PSIRP project[1], which tried to redesign the internet architecture from the point of view of publish/subscribe.

Several reviews of research in this domain have been published (among others [10], [11], [12]). However, none of them are able to provide systematic studies of the robustness of pub/sub systems. We argue that this is mainly due to their lacks in two domains: architectural classification schemes, and studies of the types of failures and other issues that may affect the functioning of the systems. Due to the focus on different aspects they are not suitable for a comprehensive architectural classification. Indeed, from this comprehensive point of view, architectural dimensions mentioned in the literature are incomplete and the terminology inconsistent. This is outlined in table I for three reviews and one publish/subscribe systems. The two lines of the table correspond to dimensions studied in the reviews that we have identified as similar. Due to this inconsistency we see the need for a general architectural classification scheme, which is proposed in section III.

Basic byzantine fault-tolerance for pub/sub always uses redundancy in a pro- or reactive manner. Examples are retransmissions of event messages, replicated broker and redundant delivery paths. Esposito et al. [8] provide an overview of redundancy techniques and a taxonomy of Byzantine faults specially tailored to publish/subscribe. The paper reviews thirty approaches starting from the year 2000. It shows that the systems mainly consider failure scenarios of link/node crashes, node churn and packet loss. These are important scenarios but far from being the only situations that can negatively impact pub/sub systems. Modified event message content and injection of arbitrary generated or unauthorized messages are examples of faulty situations that were not yet taken into account. With the taxonomy of faults the paper of Esposito et al. outlines the shortcomings of existing pub/sub systems with respect to types of failures; however, the proposed taxonomy also has its own limitations. Indeed, it does not consider important faults such as message misusage and information leaks. Moreover, it mixes peer faults (e.g. link anomalies, node crash) with failure scenarios such as node churn, which can themselves be caused by peer faults. This results in potentially overlapping categories. Thus, we argue that this taxonomy does not fulfil the requirements for a comprehensive evaluation of the robustness of pub/sub systems. To fill this gap, we propose another taxonomy of faults for pub/sub in section IV.

Rational behaviour for peer to peer systems has been first mentioned in [13]. The BAR model [3] was introduced as abstract development model to consider both Byzantine failures and rational behaviour among altruistic nodes in distributed environments. A few systems have implemented the model or considered selfish behaviour in distributed systems [14], [15], but to our knowledge, there is no pub/sub system doing this. Nielson et al. classified rational behaviour by means of a taxonomy of rational attacks [2]. Rational behaviour, though it can critically influence system functioning, is commonly ignored by reviews of pub/sub systems as well as by many systems.

III. General Classification Scheme

We introduce here a pub/sub classification scheme. Its goal is to provide a comprehensive overview of architectural dimensions for pub/sub, enabling a general comparability. It aggregates and extends previous surveys, other overviewing works around publish/subscribe such as [16], [17], [18] as well as 25 works of the past 10 years. The classification scheme comprises eight architectural dimensions, which are each described in the next subsections. For each dimension, we provide a short description, while the two dimensions that are essential for the overlay structure (dissemination technique and overlay organisation) are explained in more detail. An overview of the classification, showing the architectural dimensions with examples, is outlined in table II. Table III exemplary shows a complete classification of two sample systems using the scheme. To design these architectural dimensions of the classification we have used the following criteria:

1) The dimensions should not overlap.
2) The dimensions should be sufficient to fully classify a set of 25 works of the literature that we have identified as the most important.
3) The dimensions should completely cover all options for the 25 selected works, that are critical for the functioning of a system, namely subscription model, dissemination technique, and overlay organisation.

[1] http://psirp.hiit.fi/

Table I
HETEROGENEITY OF OVERVIEWING WORKS HINDERS COMPARABILITY OF PUBLISH/SUBSCRIBE SYSTEMS

Liu and Plale [10]	**Baldoni et al. [11]**	**Hosseini et al. [12]**	**Carzaniga et al. [4]**
System Architecture - Client/Server Model - P2P Model	Overlay Infrastructure - Broker Overlay - P2P Structured Overlay - P2P Unstructured Overlay - Overlay for Mobile Networks - Overlay for Sensor Networks	Group Management - Mesh First vs. Tree First - Source Specific Tree vs. Shared Tree - Distributed vs. Centralized - IP Multicast Capability - Refinement	Architecture - Hierarchical Client/Server - Acyclic P2P - General P2P - Hybrid
Event Distribution Scheme - Multicast	Event Routing - Flooding - Selective - Gossiping	Routing Mechanism - Shortest Path - Minimum Spanning Tree - Clustering Approach - P2P Structure	Routing Strategies - Subscription Forw. - Advertisement Forw.

Subscription Model

The subscription model enables nodes to specify the events in which they are interested. Subscription models mainly differ in their expressiveness and may allow a more fine grained specification of interest up to single messages (at the price of higher resource usage). More detailed descriptions can be found in [11].

Event Data Model

Publish/subscribe messages may contain complex information and thus, they require a structure to identify attributes and values. The event data model specifies the representation of a pub/sub related message (e.g. subscribe, notify) in the system. In systems with a high rate of published events, the data model can cause significant delays due to messages processing times. A performance comparison of two formats can be found [7].

Matching Algorithm

Matching algorithms are used by pub/sub systems in order to identify subscribers who are interested in a given event. This is done by checking related subscriptions for matching the given event (subscription matching). Matching algorithms gain in importance in large-scale systems as the amount of subscriptions and/or processed events increase. Content-based systems allow complex subscription situations. In this case, efficient matching algorithms are crucial to the scalability and performance of the publish/subscribe service. More information and a formal analysis can for example be found in [19] [20].

Overlay Organisation

This dimension specifies the organisation of the logical overlay infrastructure at the application level. This includes the choice of communication partners and the group management policy. Thus, it has a strong correlation with the dissemination technique. Three common types of organisation are *hierarchical* systems, which follow the client/server principle, *peer-to-peer* (P2P) systems and *clustered* approaches. The hierarchical organisation in form of a tree enables logarithmic hop counts on the overlay. It is suitable for systems with a low density of clients with high rates of subscription and unsubscription [4]. A clustered overlay organisation defines independent domains of administrative control. This allows different overlay organisation types and dissemination techniques, resulting in hybrid systems.
Early P2P systems had a hierarchical structure with supernodes. However, current systems are often characterized by a flat hierarchy and can be divided into *structured*, *unstructured* and *broker overlay* systems [16]. Structured P2P overlays (S-P2P) use a deterministic approach to set up the overlay organisation such as making specific nodes responsible for subscription matching. In contrast, unstructured P2P overlays (U-P2P) are of a probabilistic nature and use non-deterministic dissemination techniques such as gossiping. Broker overlays (BO), e.g. [7], introduce a distinction between nodes that publish or subscribe (clients) and forwarder nodes (servers), also called brokers. This allows systems with low performance clients such as sensors, while brokers must have better capabilities. Client nodes have exactly one broker as access point to the overlay network, which is formed by brokers. This structure can be seen as a hybrid between a one-level-hierarchical and a P2P overlay. Here, we consider broker overlays as a subdivision of P2P. Indeed, the set of nodes formed by a broker and its client can be conceptually seen as single node in a P2P overlay, which aggregates the interests of all its clients.

Dissemination Technique

This dimension specifies the routing technique on the overlay network and is a crucial factor of the scalability of the system. Typical techniques are *flooding*, *selective* and *gossiping* algorithms. Flooding can be divided into subscription and event flooding. Subscription flooding informs all nodes about the subscription, while event flooding distributes a published event to all nodes without respecting subscriptions. Selective algorithms make use of a deterministic routing process based on subscriptions and can be divided into filtering and rendezvous-node based systems. In selective/filtering the subscription matching is done in a decentralized way: a node forwards events only to those

Table II
CLASSIFICATION SCHEME OVERVIEW

Architectural Dimension	Categories
Subscription Model	- topic-based - type-based - content-based - context-based
Event Data Model	- serialized - tagged - untagged binary
Matching Algorithm	- Predicate indexing, e.g. - lookup tables - Hanson et al. algorithm - Testing network, e.g. - matching trees - binary decision diagrams
Overlay Organization	- hierarchical - P2P - structured (S-P2P) - unstructured (U-P2P) - broker overlay (BO) - clustered
Dissemination Technique	- subscription / event flooding - selective - filtering / rendezvous node - gossiping - basic / informed
Adaptation Technique	active / passive
Underlay Awareness	- proximity-awareness - quality-awareness
Quality of Service	- reliability - e.g. guaranteed delivery - delivery semantics - e.g. at most once, exactly once - latency/bandwidth guarantee - message ordering

outgoing links that have (matching) subscribers behind them. Selective/rendezvous based systems are characterized by a specific node that is responsible for subscription matching of a multicast group (rendezvous or root node). This node is usually chosen by applying a mathematical function (e.g. hash function) on parameters of the event (e.g. topic identifier). All events of a group are sent to the rendezvous node, which is aware of subscribers and forwards the event to them. Gossiping is a probabilistic dissemination technique in which events are forwarded to a set of randomly chosen nodes. These are partially defined in a deterministic way for the informed gossiping, e.g. by forming a logical ring in order to guarantee delivery.

Adaptation Technique

Publish/subscribe systems may be able to adapt to system changes. Caused by faults or protocol coherent operations, adaptations are in general related to topological changes such as communication links or leaves/joins of nodes. The adaptation itself depends on the specific system and type of change. Therefore, the adaptation technique is defined here by how the need for adaptation is identified.

Underlay Awareness

A pub/sub system may be underlay-aware. In such a system, knowledge about the underlying physical network is available at the overlay level. This enables an optimisation of the overlay, which can adapt itself to physical network changes. A topic-based pub/sub system that follows a clustered approach could for example leverage the already available IP Multicast as intra-cluster dissemination technique to notify subscribers [8].

Quality of Service (QoS)

Publish/subscribe systems offer a loosely coupled type of communication. Therefore, they can hardly provide guarantees on the quality of service and the result is an operation only in a best-effort mode. However, QoS guarantees are helpful for specific applications, such as video streaming. QoS is often orthogonal to the system architecture and could be interpreted as QoS parameters. Here, for a clear arrangement we define it as own dimension specifying the QoS capabilities without direct relation to the architectural dimensions (this should be done in a closer look to the system). A taxonomy of QoS for event-based middlewares are presented by Mahambra et al. [22].

IV. FAULTS AND THREATS IN PUB/SUB

Here, we propose a comprehensive taxonomy of faults to study the robustness of pub/sub systems. It focuses on elementary peer operations. Thus, the faults are the origin for more complex failure scenarios but affect both the network under- as well as the overlay.
The taxonomy comprises the faults listed below. These faults may occur for a number of reasons including lossy communication links, programming or technical failures. *Information leaks* are not directly a danger for the robustness but important with respect to privacy and security. Indeed, it enables to draw conclusions about other peers or system states. The gained knowledge can be used to design a rational strategy; therefore, they should be prevented in addition to direct failures.

- *Operation Anomalies*
  - *Link/node unavailability*: A communication link or peer is temporarily unavailable and can no longer interact with the system.
  - *Link/node crash*: A communication link or the whole peer crashes (and therefore its links crash as well). It does not interact anymore with the system and needs to be restarted.
- *Message Anomalies*
  - *Delay*: The forwarding of a message is delayed, the message is delivered later than expected.
  - *Information Leak*: Message occurrences and contents (header and body are fully readable) allow conclusions to be drawn.

Table III
EXEMPLARY CLASSIFICATION OF TWO SYSTEMS

Name	Subscription Model	Event Data Model	Matching Algorithm	Overlay Organisation	Dissemin. Technique	Adaptation Technique	Underlay Awareness	Quality of Service
Scribe [21]	topic	tagged	lookup table	S-P2P	selective (rendezvous)	active	—	reliability
Gryphon [7]	content	untagged binary	predicate matching	BO/S-P2P	selective (filtering)	passive	proximity	reliability

- *Loss*: A message is lost while being delivered to subscribers (while being transmitted through a lossy link or at the peer between transmissions).
- *Generation*: A protocol coherent message of arbitrary type and content is generated and sent to a set of overlay neighbours.
- *Misusage*: Protocol coherent messages are used in an arbitrary way.
- *Tampering*: The message is modified while being delivered. The tampering may affect any field of the message (also header) but the resulting one is protocol coherent.

These Byzantine failures are atomic failures of a peer, which can lead to further system/network failures. The most important kinds of such failures are the following:

- *Node Churn*: Peers leave/join the system at a high rate using protocol coherent messages. The resulting system adaptation consumes system resources and affects the correct system functioning by the extra time needed to disseminate changes in the system.
- *Catastrophic Failures*: A specific failure appears simultaneously at a large number of peers. This may for example arrive for the failures *Link/Node Crash*, *Message Loss* or *Message Tampering*.
- *Network Partitioning*: The overlay network gets split into independent logical networks due to link or node crashes. The events originating in a partition can no longer be transmitted to the others.
- *Message Ordering Violation*: Published messages are not received in the same order as they were sent.
- *Content Interference*: The overall transferred content in the system is tampered with an injection of generated messages with arbitrary content, which can lead to unsuitable actions of the application.

V. RATIONAL BEHAVIOUR IN PUBLISH/SUBSCRIBE

A. Outline of Rational Behavior

The notion of rational nodes in peer-to-peer networks was introduced by Sneidman and Parkes [13]. From a game-theoretic point of view, rational nodes follow a strategy to increase personal benefit (utility), which may lead them to deviate from the protocol. The utility always depends on the circumstances and on the node's goals.

For this paper we follow the economic model of node behaviour of Nielson et al. [2]. The model distinguishes between peers that *do strategize* and those that *do not strategize*. Peers that do not strategize are either *altruistic* (follow the protocol without any other considerations) or *faulty* (involuntarily erroneous, without strategic reasons). Peers that do strategize are further divided into *rational* and *irrational* ones. The action of rational nodes are guided by the goal of maximizing their utility. It is assumed that they have a correct knowledge of the system, which they use to their advantage. They may follow the protocol but may also commit malicious actions if it improves their utility. Irrational peers are strategizing peers, whose actions do not necessarily result in the improvement of their utility, either because they have a wrong or incomplete knowledge of the system or because they have goals that lie outside the economic mechanisms. The selfish behaviour of rational nodes is harmful for the system operation and can be considered as an attack [2], [13]. A rational node can try to hide it behind Byzantine faults but deviating too much or too frequently from the protocol increases the chance to be identified as malicious. This would be the worst case for a rational node, because it may be expelled from the system, and therefore completely lose the utility that it was getting from it. Thus, a good strategy for the node is to behave rationally only for a part of the messages. This makes it difficult for the system to identify it as malicious if no specific functions have been implemented for this purpose.

To the best of our knowledge, rational behavior has not yet been considered for robustness evaluations of pub/sub systems. Furthermore, the definition of rationality used here also covers the possibility of collusion of peers. This is a real threat considering the possibility that the original software logic gets modified at multiple peers (e.g. by insiders or by hacks through exploitation of bugs). Further discussions about rational behavior including mechanisms to prevent it can be found in [2].

B. Experimental Study

We now describe a simulation-based experimental study, which (a) verifies the harmful influence of rational behaviour

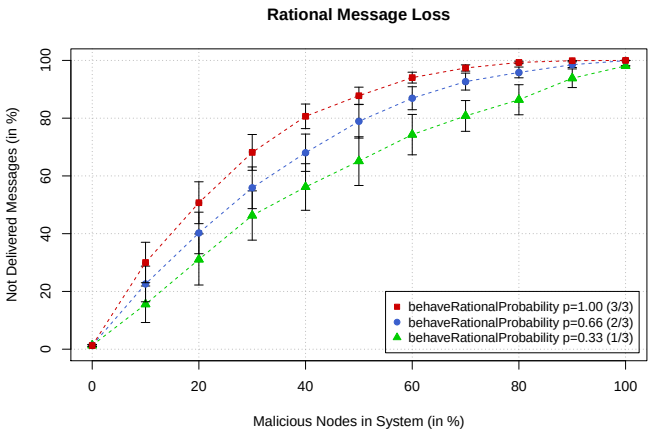

Figure 1. Worst-case scenario of rational behaviour for message loss.

and (b) outlines the impact on system functioning in a worst-case. The experiments are only exemplary for one system and one failure type (message loss). We selected Scribe [21] as our target system for the following reasons. First, Scribe uses a rendezvous node based overlay organisation, which is a clear (no clustered/hybrid approach etc.) and common type in pub/sub systems. Second, being one of the first systems, Scribe has only limited fault-tolerance capabilities. It tolerates only link/node crashes and implements no further redundancy techniques, i.e. each node in the multicast tree is a single-point-of-failure in terms of message loss. Scribe thus directly shows the impact of faults without being moderated by fault-tolerance techniques and can therefore serve as a worst-case system with respect to the consequences of the faults. The experiments consider rational behaviour by implementing the option of malicious nodes only dropping a fraction of messages (a standard strategy to try to hide their malicious behaviour). Note that due to the overlay organisation and dissemination of Scribe (multicast tree without redundancy), the scenario of these tests also covers the case of message tampering. The subscribers being affected are the same, independent of the failure type (lost/dropped or modified message). The rational aspect can also be considered as a group of colluding peers, which are working together to harm the system by dropping or modifying messages.

The simulation was done with peersim[2], an open-source peer-to-peer simulation environment. The simulated overlay dealt with 200 topics and 1000 nodes in the system, each topic having 50 (0.05 % of network size) randomly chosen subscribers. The study comprises a total amount of 8000 published messages (40 messages per topic) per experiment and is performed equally distributed over all nodes. Measured values are then calculated by averaging the results of 50 experiment runs. The probability of behaving rationally is varied by setting it to 1 (100 %), 0.66 (2/3) and 0.33 (1/3). The 100 % value corresponds to the situation in which affected nodes lose all their messages due to Byzantine failures or in which peers can behave rationally without fearing punishment. Lower values represent situations in which a rational peer tries to hide its deviation from the protocol, i.e. acts maliciously only for a fraction of messages.

Figure 1 shows the results of the simulation with standard deviation values. Considering a 100 % probability of rational behavior, one can see that more than the half of all subscribers are not informed with the corresponding event message in the presence of 20 % rational nodes, which increases to about 80 % if a third of the peers are rational. The negative impact is slightly reduced if rational peers try to hide the deviation of the protocol but the amount of not informed subscribers remains significant. For the presence of a third of malicious peers, which deviate from the protocol for only a third of messages ($p = 0.33$), the simulation shows that more than 40 % of all subscribers are not informed.

This simulative study outlines the strong negative influence of rational behaviour on correct system functioning. As a single system was tested, the results can only be considered as samples. However, they can be transferred to similar systems due to the common type of the studied system and the general lack of anti-rational techniques in typical pub/sub systems identified in this paper.

VI. Conclusion and Future Work

This paper introduced a structured approach to robust publish/subscribe systems from a practical point of view. It consists of a general architectural classification scheme, which aggregates a multitude of works, and a taxonomy of faults from the viewpoint of a peer. We argue that rational behaviour should also be considered to enhance the robustness of a system. This need has been exemplary verified in a simulative study, which also outlined the harmful impact in a worst-case scenario.

The proposed approach can be used to improve the robustness of pub/sub systems in two directions. First, it may serve as common base for comprehensive evaluation models of the robustness of pub/sub systems. Second, it can support the development of robust pub/sub systems. In particular, the taxonomy of failures by focusing on elementary faults of a peer can show what techniques are critical to include in order to avoid system failures.

Our plans for future work in this domain include the definition an analytical model based on this work to evaluate the robustness of publish/subscribe systems, taking rational behaviour into account. Furthermore, we will design a benchmarking suite following the basis for robustness evaluation defined in this paper. This would dramatically simplify

[2]http://peersim.sourceforge.net/

simulative evaluations and at the same time guarantee the comparability of the results.

ACKNOWLEDGMENT

This work was conducted in the framework of the Multimedia, Distributed and Pervasive Systems (MDPS) network. The MDPS is a French-German doctoral college supported by the Université Franco-Allemande (CDFA-05-08) and the Deutscher Akademischer Austausch Dienst (D/08/10806). Further support has been received by the Région Rôhne Alpes.

REFERENCES

[1] D. Oppenheimer, A. Ganapathi, and D. Patterson, "Why do internet services fail, and what can be done about it?" in *Proc. of the 4th conference on USENIX Symp. on Internet Technologies and Systems - Volume 4*, ser. USITS'03, 2003.

[2] S. Nielson, S. Crosby, and D. Wallach, "A taxonomy of rational attacks," in *Proc. of the 4th Int. Workshop on Peer-to-Peer Systems (IPTPS'05)*, 2005, pp. 36–46.

[3] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J. Martin, and C. Porth, "Bar fault tolerance for cooperative services," in *Proc. of SOSP'05*, 2005, pp. 45–58.

[4] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems (TOCS'01)*, vol. 19, no. 3, pp. 332–383, 2001.

[5] S. Voulgaris, E. Rivière, A. Kermarrec, and M. Van Steen, "Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks," in *Fifth Int. Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.

[6] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication," in *Proc. of the 2007 Int. Conf. on Distributed Event-based Systems*, 2007, pp. 14–25.

[7] The Gryphon Team, "Achieving scalability and throughput in a publish/subscribe system," IBM Research, Tech. Rep. Report RC23103, 2004.

[8] C. Esposito, D. Cotroneo, and A. Gokhale, "Reliable publish/subscribe middleware for time-sensitive internet-scale applications," in *Proc. of the Third ACM Int. Conf. on Distributed Event-Based Systems*, 2009, pp. 1–12.

[9] N. Carvalho, F. Araujo, and L. Rodrigues, "Scalable qos-based event routing in publish-subscribe systems," in *Proc. of the Fourth IEEE Int. Symp. on Network Computing and Applications (NCA'05)*, 2005, pp. 101–108.

[10] Y. Liu and B. Plale, "Survey of publish subscribe event systems," Indiana University, Tech. Rep., 2003.

[11] R. Baldoni, L. Querzoni, and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," University of Rome, Tech. Rep., 2005.

[12] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. Georganas, "A survey of application-layer multicast protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[13] J. Shneidman and D. Parkes, "Rationality and self-interest in peer to peer networks," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2735, pp. 139–148.

[14] H. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robinson, L. Alvisi, and M. Dahlin, "Flightpath: Obedience vs choice in cooperative services," in *Proc. of the 8th USENIX Symp. on Operating Systems Design and Implementation*, 2008.

[15] Q. Li, S. Zhu, and G. Cao, "Routing in socially selfish delay tolerant networks," in *Proc. of IEEE INFOCOM 2010*, 2010, pp. 1–9.

[16] X. Shen, H. Yu, and J. Buford, Eds., *Handbook of Peer-to-Peer Networking*. Springer, 2010.

[17] S. Tambe, "State-of-the-art in publish/subscribe middleware for supporting mobility," Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, USA, 2007.

[18] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems*. Springer, 2006.

[19] W. Rjaibi, K. Dittrich, and D. Jaepel, "Event matching in symmetric subscription systems," in *Proc. of the 2002 conference of the Centre for Advanced Studies on Collaborative research (CASCON'02)*, 2002.

[20] S. Kale, E. Hazan, F. Cao, and J. Singh, "Analysis and algorithms for content-based event matching," in *Proc. of the Fourth Int. Workshop on Distributed Event-Based Systems*, 2005, pp. 363–369.

[21] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489 – 1499, October 2002.

[22] S. Mahambre, M. Kumar S. D., and U. Bellur, "A taxonomy of qos-aware, adaptive event-dissemination middleware," *IEEE Internet Computing*, vol. 11, 2007.