Reliable Publish/Subscribe Middleware for Time-sensitive Internet-scale Applications

Christian Esposito, Domenico Cotroneo
Dept of Computer and Systems Engineering,
Università di Napoli - "Federico II"
Napoli, 80125 - Italy
{christian.esposito, cotroneo}@unina.it

Aniruddha Gokhale
Institute for Software Integrated Systems
Dept of EECS - Vanderbilt University
Nashville, TN 37235 - USA
a.gokhale@vanderbilt.edu

ABSTRACT

Federating mission critical systems over wide-area networks still represents a challenging issue. For example, it is hard to assure both reliability and timeliness in a hostile environment such as the Internet. The publish/subscribe (pub/sub) interaction model is a promising solution for scalable data dissemination over wide-area networks. Nevertheless, currently available pub/sub systems lack efficient support to achieve both reliability and timeliness in unreliable scenarios. This paper describes an innovative approach to fill this gap making three contributions. First, a cluster-based peer-to-peer organization is introduced to handle a large number of publishers/subscribers. Second, the cluster coordinator is replicated to mask process crashes and to preserve cluster connectivity toward the outside world. Third, multiple-tree redundancy is applied to tolerate link crashes thereby minimizing unpredictability in the delivery time. We present a simulation-based evaluation to assess the effectiveness of the proposed approach in an unreliable setting. This study indicates that our approach enforces the reliability of event delivery without affecting its timeliness.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Reliability, availability, and serviceability

Keywords

Fault-tolerance, Timeliness, Peer-to-Peer Overlay, Replication, Multiple-Tree Redundancy

1. INTRODUCTION

In the last few years, more and more industrial projects aim to develop the so-called *Large scale Complex Critical Infrastructures* (LCCIs) [1], *i.e.*, Internet-scale federation of several autonomous and heterogeneous systems that work collaboratively and synergistically to provide critical facilities. This represents a novel prospective on how mission critical systems are architected: from small, monolithic and vertical architectures, which characterized traditional systems, there has been a shift toward large highly modular and integrated systems. As a practical example, we can consider the road map outlined by EuroControl for the European Air Traffic Control (ATC) evolution, object of the European Research Project called *Single European Sky* (SESAR) [1]. The current European ATC framework is segmented among several systems, namely Area Control Centers (ACCs), each one responsible for a well-defined portion of the air space. In order to handle more efficiently the growing aviation traffic, the solution is a framework where the ATC operations are seamlessly and fully integrated. In fact, the novel ATC framework will be based on a data-centric model in which all Area Control Centers cooperate via a data distribution service.

The effectiveness and performance of LCCIs strongly depend on the quality of the adopted interconnection middleware, which has to deal with some serious challenges. First, critical operations exhibit a time-sensitive behavior: information is useful only if delivered "on time", *i.e.*, respecting certain time boundaries. So, *messages have to be exchanged with a predictable latency* (**Timeliness**). Second, one of the key properties that a critical system has to provide is to be dependable and able to handle properly error conditions imposed by network and process faults. Then, *message dissemination has to be guaranteed despite manifestations of several faults* (**Reliability**). Last, the rising of the activity of a single system and the escalation of connected systems increase the number of data exchanges, and consequently the time of computation. Therefore, the adopted *middleware must be able to scale while maintaining suitable performance* (**Scalability**).

Middleware solutions that adopt a publish/subscribe interaction model, *i.e.*, pub/sub services, are very appealing to efficiently interconnect several systems since the inherent decoupling properties promote scalability [2]. The focus of the publish/subscribe community has never been on reliable event dissemination for two main reasons. On one hand, guaranteeing message delivery despite network failures has been always thought as inherited by the pub/sub system from the protocol used to implement the notification service. On the other hand, there were more challenging issues, such as scalability and expressiveness. However, research interest of the community is recently increasing more and more toward novel approaches to satisfy reliability requirements. So, some techniques have been developed specifically for these middleware to cope with several kind of faults. However, these architectures lack an adequate support to jointly assure reliability e and timeliness. In order to fill this gap, this paper presents a novel approach based on a

[1]This is one of the key objectives of the research project, namely IniziativaSoftware (www.iniziativasoftware.it), supported by the University of Naples and by FINMECCANICA, one of the largest Italian company which develops mission critical and complex system infrastructures in the context of military and civil scenarios.

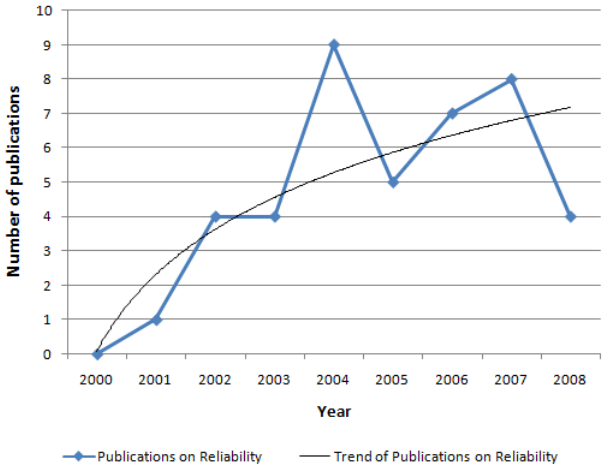

Figure 1: Efforts on reliability aspects in publish/subscribe middleware

peer-to-peer infrastructure. Our driving ideas are the following:

- Clustering publishers and subscribers that reside on the same routing domain, and using a hierarchical peer-to-peer organization so to handle large numbers of participants without affecting the message latency;
- Implementing a replication schema of the coordinator in each cluster so to treat process crashes without leading to disconnections among the clusters or considerable fluctuations in the message delivery time;
- Adopting a multi-tree approach so to cope with link crashes and to preserve connectivity into the coordinators overlay without worsening the timeliness of the data dissemination.

The reminder of the paper is organized as follows. Section 2 provides 1) a definition of the properties that a reliable pub/sub service has to provide in order to be used to federate time-sensitive applications, 2) background on the faults that can occur in pub/sub middleware, and 3) a taxonomy of the available solutions to implement a reliable pub/sub system. In Section 3, we describe in details our approach, while in Section 4 we report experimental results of our initial simulation-based study to assess the quality of our approach. Lastly,, section 5 discusses related work, and we conclude in Section 6 with some remarks on future works.

2. BACKGROUND AND OPEN ISSUES

Since its early years, the publish/subscribe community has been more focused on scalable architectures, efficient delivery, and expressive subscriptions rather than reliable event dissemination. As a proof of this lack of attention on fault-tolerance issues, main standardized, and mature, commercial pub/sub services do not address them at all, such as *Java Message Service* (JMS) [3], or provides very basic mechanisms, such as the recent OMG standard called *Data Distribution Service* (DDS) [4]. However, the status is changing since pub/sub services have started to be also used in application domains that expose stringent reliability requirements. *E.g.*, EuroControl decided that the technology of reference to be used in the project SESAR, which aims to device the novel European ATC framework by interconnecting critical systems, is the DDS specification. We have performed an assessment of the available literature on reliable pub/sub services in order to measure the efforts spent on such topic. We have collected all the papers published at international conferences and journals, counting how many of them are focus on reliable pub/sub services at a each year starting from 2000[2]. As shown by the black line in Figure 1, our study of the literature demonstrates that in the last decade we have witnessed a growing interest of the community in studying novel approaches to guarantee a reliable event dissemination upon networks and nodes that expose a faulty behavior. This section presents a study of reliability aspects in pub/sub service by means of 1) defining the fundamental properties that a reliable pub/sub service has to provide; 2) formalizing which failures are of interest to be handled in order to guarantee a reliable event dissemination; and 3) analyzing the current approaches to devise a reliable pub/sub service in order to find the most suitable one to provide both reliability and timeliness.

2.1 Reliable Publish/Subscribe

A pub/sub service is made of several processes that exchange messages through a so-called notification service. These processes play the roles of *publishers*, which send messages, and/or *subscribers*, which receive the messages in whom they are interested. In fact, a process p_j receives only those messages that satisfy the k-th, namely $C_{j,k}$, of its active subscription predicates, contained in the set called Σ_j: if the message m_i matches the subscription $C_{j,k}$, then $C_{j,k}(m_i) \equiv \top$ and m_i is delivered to p_j, otherwise $C_{j,k}(m_i) \equiv \bot$ and m_i is not delivered to p_j. Given a message m_i, it is possible to define as its view, namely $viewof(m_i)$, all the processes that have to receive the message m_i:

$$viewof(m_i) = \begin{cases} V & \exists j,k : C_{j,k}(m_i) = \top \\ \bot & otherwise \end{cases}$$

where $V = \{p_j | \exists C_{j,k} \in \Sigma_j : C_{j,k}(m_i) = \top\}$.
So, during its operational phase a process p_j performs a series of the following four operations:

1. *Publish*: At time t, a message m_i is published if p_j sends it to the notification service, namely NS, and the view of the message is not empty:
$$t = pub(m_i, p_j) \Rightarrow t = send(m_i, NS),$$
under the constraint that $viewof(m_i) \neq \bot$;
2. *Notify*: At time t, p_j is notified of a published message when a message is received by the notification service, namely NS, and one of the subscriptions contained in Σ_j is verified:
$$t = not(m_i, p_j) \Rightarrow (t = recv(m_i, NS) \wedge C_{j,k}(m_i) = \top);$$
3. *Subscribe*: A new subscription $C_{j,k}$ is created:
$$sub(C_{j,k}, \Sigma_j) \Rightarrow \Sigma_j = (\Sigma_j \cup C_{j,k});$$
4. *Unsubscribe*: An existent subscription $C_{j,k}$ is erased:
$$unsub(C_{j,k}, \Sigma_j) \Rightarrow \Sigma_j = (\Sigma_j \cap C_{j,k}).$$

A so-defined pub/sub service has to satisfy two main properties:

☞ *Safety*: a process p_j is notified of a message m_i because another process p_i has previously published it:
$\exists p_j : t_n = not(m_i, p_j) \Rightarrow \exists p_i : (t_p = pub(m_i, p_i) \wedge t_p < t_n)$;

☞ *Liveness*: if a process p_j has a subscription $C_{j,k}$, it receives one of the published messages that satisfy it:
$$\exists p_j, \exists C_{j,k} \in \Sigma_j \Rightarrow \exists m_i : not(m_i, p_j).$$

A pub/sub service is defined *reliable* if the message delivery is guaranteed despite that processes may fail and/or the network may be affected by several anomalies. So, considering a view of the message m_i, namely V, the following properties have to be guaranteed by a reliable pub/sub service:

[2]A work can be published in several papers in different years, namely paper distribution. In this case, we have considered it as a single paper in the median year of the paper distribution, *e.g.*, if a work has been published in 3 papers in 2002, 2 papers in 2003 and 2 paper in 2004, we have considered it as one paper in 2003.

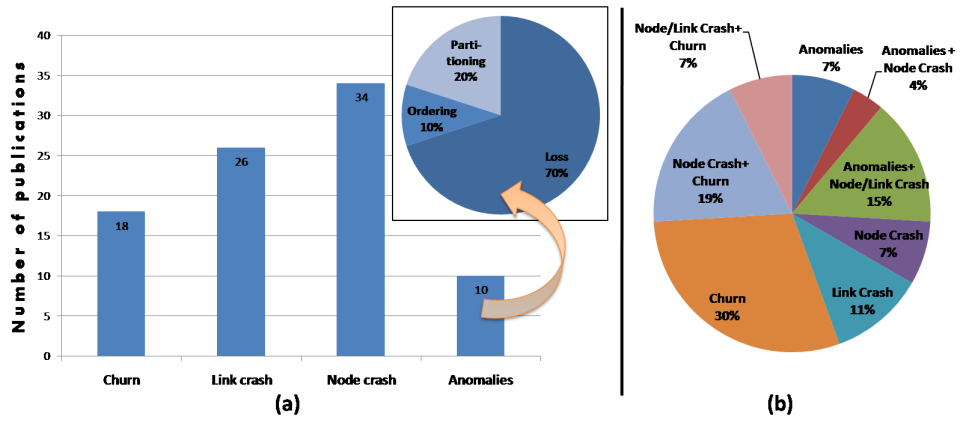

Figure 2: (a) Distribution of current solutions for each fault; (b) Taxonomy of faults addressed by current solutions

- ⋆ *Agreement*: if a non-faulty process p_j is notified, then all the other non-faulty processes of the view are eventually notified:
$$\exists p_j \in V : not(m_i, p_j) \Rightarrow \forall p \in V : not(m_i, p);$$
- ⋆ *Validity*: if a non-faulty process p_j publishes the message m_i, then at least one of all non-faulty processes of the view is notified of the message:
$$\exists p_j \in V : pub(m_i, p_j) \Rightarrow \exists! p_k \in V : not(m_i, p_k);$$
- ⋆ *Integrity*: every non-faulty process p_j performs the notify of the message m_i at most once:
$$\exists p_j \in V : not(m_i, p_j) = t_{n1} \Rightarrow \nexists not(m_i, p_j) = t_{n2} : n1 \neq n2.$$

Moreover, since critical systems, such as LCCIs, exhibit real-time constraints jointly to fault-tolerance, a pub/sub service is suitable in this context only if it verifies an additional property:

- ⋆ *Timeliness*: given a deadline Δ, all non-faulty processes are notified of a published message before Δ is expired:
$$\exists \Delta : pub(m_i, p_j) \Rightarrow \nexists p_k \in V : not(m_i, p_k) > \Delta.$$

2.2 Failure Model

Several kind of faults may affect a pub/sub service, and they need to be carefully treated. Briefly, such faults can be classified as follows:

- **Network anomalies**: temporary misbehaviours of a link:
  - *Loss*: links behave as a fair loss channel, *i.e.*, messages in transit though the link may be lost;
  - *Ordering*: messages are not received in the same order as they have been published;
  - *Corruption*: messages are received corrupted;
  - *Delay*: a message is delivered later than expected;
  - *Congestion*: a link/router is overloaded and several anomalies may suddenly happen;
  - *Partitioning*: network is fragmented into several disconnected parts.
- **Link crash**: links experience loss of connectivity, *i.e.*, packets are always lost for a certain time, which is not necessarily permanent but may dynamically appear and disappear;
- **Node crash**: nodes crash due to hardware/software failures;
- **Churn**: nodes unexpectedly join/leave the system.

We have analyzed which kind of faults, with respect to this taxonomy, are handled by the available reliable pub/sub services. Only 10% of the available systems, as shown in Figure 2(a), handle faults due to network anomalies, and the majority of such systems, 70% of them, focus on recovering from message losses. The reason behind this may be found on the consideration that pub/sub services are designed on top of an *event dissemination protocol*. Studing how to cope with network anomalies may be not felt as a challenging issue by the community since it can be simply addressed using a reliable event dissemination protocol, such as one of the reliable multicast protocols developed during the last twenty years [5, 6, 7].

On the other hand, many efforts have been spent to address faults that can lead to inconsistent topologies within the broker overlay. The most studied fault is the node crash, 34% of the systems, in particular the case in which the crashed node hosted a process that acted as a broker, which is critical in the system since it glues togethers publishers and subscribers. While, only a smaller number of systems, 18%, consider also churn. This may be due to two reasons: 1) mostly all these systems are built on peer-to-peer infrastructures, which gracefully manage these dynamics[3]; 2) this fault is treated for free, without a particular solution. Looking at Figure 2(b), we can notice that systems dealing with churn also handle node crashes. In fact, churn can be seen as a special case of node crash, so all the techniques adopted to cope with node crashes can be also used in this other case. On the other hand, the second most studied class of faults is the link crash, specifically in 26% of the systems. In conclusion, as shown in Figure 2(b), most of the reliable pub/sub systems deal jointly with node and link crashes, 64% in total. So, putting everything together, a failure model suitable for pub/sub services comprises only node and link crashes.

2.3 Approaches for Reliable pub/sub service

This sections aims to provide a taxonomy of the current approaches adopted in literature to implement reliable pub/seb services. There are two ways to implement the notification service: one uses IP Multicast [4], while the other one adopts an application-level multicast for scalability reasons [8]. The first two approaches in the taxonomy are employed in the first prospective, while the other approaches in the latter one.

Retransmissions (ARQ). A way to achieve reliable event dissemination is to have publishers storing messages, so that subscribers can ask for their retransmission when losses are somehow detected [4]. Despite the ease with which it can be implemented,

[3]In fact, the ones that do not care about churn adopt a static broker overlay rather than a peer-to-peer infrastructure.

this approach is not optimal due to several drawbacks. First, it can deal only with message losses. Second, the time to recover dropped messages is unpredictable. In fact, the number of retransmissions needed to successfully receive a message depends on the network behavior, which is never known a priori. Moreover, there are no guarantees to achieve agreement, due to a link crash that disconnect partially/completely publishers to a part of the interested subscribers or the possibility that a publisher may crash before all the subscribers have recovered the lost messages. Lastly, messages can be received twice due to false-negative detection of dropped messages.

Forward Error Correction (FEC). A different way to recover lost message is to use spatial redundancy rather than temporal redundancy: instead using retransmissions, the sender forwards redundant data, so the receiver can reconstruct the original message even if some packets has been dropped during the delivery [9]. This solution enforces timeliness, since the worst case latency is predictable, however, it presents several drawbacks. The main one is that agreement is reached depending on the redundancy used at the sender side. In fact, a receiver can obtain the message only if the sender has delivered more additional data than the packets lost by the network, otherwise the dropped packets are more than the ones the receiver can reconstruct and the message is considered lost. Choosing the right redundancy is a very hard task in Internet, since the pattern loss of the network is highly variable over time.

Epidemic Algorithm (Ep-ARQ). ARQ suffers from a serious scalability limitation due to the centralization of the recovery duty at the publisher side. To overcome it, a different approach is to use an epidemic approach [10] that distributes the recovery responsibility among the leaf nodes of the forwarding tree. In fact, each process exchanges at a random time its history of the received messages with a randomly-chosen process among the ones constituting the system. Subsequently, inconsistencies, *i.e.*, message drops, are detected by comparison and corrected through retransmissions. It provides more guarantees than using ARQ since it can also recover messages lost due to crashed nodes and/or links. Moreover, since the detection of lost messages is performed by comparison of histories rather than timeout expiration, it avoids to receive the same message twice. However, there is no guarantee that the published message reaches all the subscribers, but agreement is obtained within a certain probability due to the random nature of the algorithm. Moreover, there is no predictable upper bound on the time to reach the agreement since retransmissions are adopted.

Reconfigurations (Reconf). Adopting topological reconfigurations is another possible solution to recover connectivity in the forwarding tree after a node/link has crashed. An example is the self-stabilization publish/subscribe [11], *i.e.*, routing entries into the brokers have a limited validity, and they have to be periodically renewed otherwise they are deleted. This approach aims to guarantee a consistent connectivity into the system, *i.e.*, all the process are connected each other. But, it does not cope with message drops, so not all the subscriber receive the messages of interest if network omissions may happen

Broker Replication (BroRep). Another method to achieve fault-tolerance is applying redundancy into the system. This is realized by replicating the brokers in the forwarding tree [12]. The state of a broker is replicated to its neighbors, so in case of a broker failure it can be easily substituted without losing subscription consistency into the system. This solution is tailored only on node failures, and link crashes may involve that some subscribers are not reachable and so there may be no agreement in the system.

Path Redundancy (PatRed). Systems such as Bayeux [13] and XNET [14] use a different form of redundancy, *i.e.*, establishing redundant paths among the nodes of the system. A message is sent through multiple paths, and only the first-received replica of the message is delivered to the application. This solution appears to be tailored to link failures, but can also cope with node crashes just circumventing the failed node. This allows all the properties of the reliable pub/sub service plus timeliness under one condition: all the paths to a destination exhibit path diversity, *i.e.*, if a path fails, the others are not affected by the same failure.

Table 1: Properties guaranteed by the current approaches

System	Agreement	Validity	Integrity	Timeliness
ARQ [4]		✔		
FEC [9]	✔*	✔	✔	✔
Ep-ARQ [10]	✔**	✔	✔	
Reconf [11]		✔	✔	
BroRep [12]		✔	✔	
PatRed [13, 14]	✔***	✔	✔	✔

* only with the appropriate redundancy
** only within a known probability
*** only if path diversity is guaranteed

As shown in table 1, path redundancy is an appealing solution to architect reliable publish/subscribe middleware with timeliness constraints, however, providing path diversity is still a challenging issue. In fact, different overlay links may be built on top of the same network links, so choosing different overlay links does not guarantee diversity. To mitigate this issue, the adopted broker overlay needs to be topology-aware of the underlay network [15], but current solutions do not provide this feature. This paper aims to define a multi-tree overlay that is aware of the underlay topology. Path redundancy has been extensively used for resilient overlay multicast in the context of multimedia applications [16]. The main difference with such approaches is that the adoption of broker replication allows the proposed approach to fast detect and recover node crashes without leading to a loss of connectivity in the overlay tree for a certain period of time, enforcing the agreement property.

3. THE PROPOSED APPROACH

In this section we describe the design of a novel approach for a topic-based publish/subscribe middleware to federate mission critical systems over wide-area networks, *e.g.*, the so-called *Large scale Complex Critical System* (LCCI) [1]. Due to the requirements that LCCIs impose on the adopted middleware, this novel approach has been designed keeping the following objective in mind: all the subscribers are guaranteed to receive on-time all the messages despite several faults may occur (*reliable and timely event delivery*).

Considering that our application scenario is related to LCCIs, we envision that a pub/sub service is composed by tens or hundreds of processes hosted on nodes scattered over Internet. Moreover, these processes do not expose churn: all the nodes provide a long-running service and suddenly leave the system only due to a failure. Due to the conclusions of subsection 2.2, we also assume that a pub/sub service can be affected by only node and link crashes among the plethora of possible faults. On one hand, processes may fail by crashes, which cause nodes to halt and to lose theirs internal volatile state, and, without lacking generality, we assume that processes do not recover after a crash (*fail-stop failure model*). On the

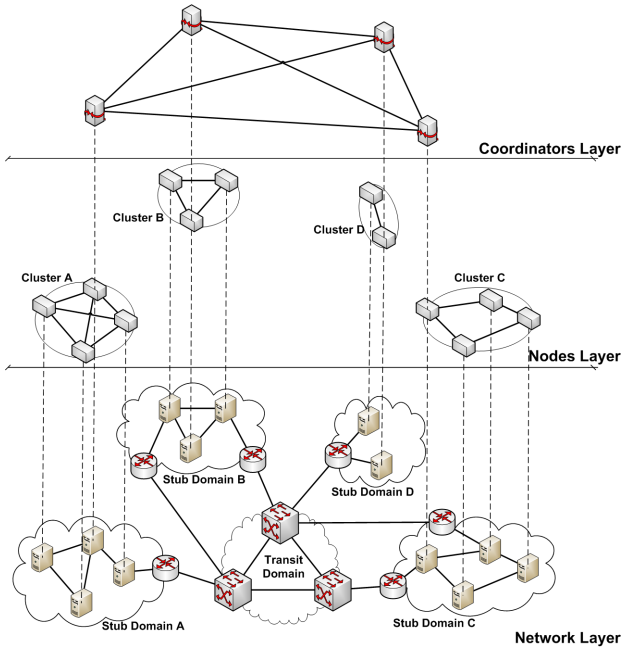

Figure 3: Different layers of abstraction in a pub/sub service

other hand, links can crash, however, they recover after a certain period of time has elapsed since the crash happened (*fault-recover failure model*), so the same link may experience several crashes during the operational phase of the pub/sub service. Moreover, we assume that the network is not partitionable due to link crashes: give two correct processes p and q, there always be a correct path, *i.e.*, constituted by not-crashed links, that allows p to reach q. Even if that situation is likely to happen in real use cases that adopt Internet as interconnection infrastructure, we decided to do not address it in this paper[4] and left it for future work.

3.1 Grouping Nodes in Clusters

The architecture of current Internet is made by a collection of interconnected *Routing Domains*, each one sharing common administration control and routing protocol [17]. Domains exhibit a hierarchical topological organization according to two abstraction levels, as illustrated in the lower part of Figure 3. On one hand, there are the so-called *Stub Domains*, within which the path joining two of its nodes stays. These domains may consist of Local Area Networks (LANs) or Autonomous Systems (AS), and are managed by a central organization, so policies to assure Quality-of-Service (QoS) constraints in the data dissemination may be applied. On the other hand, *Transit Domains* are in charge to efficiently interconnect several stub domains and to form the network backbone. Lacking a central management reference and traffic orchestrator, transit domains are affected by several failures that may compromise the effectiveness and resiliency of the message forwarding. Although important technical progress has been made [18], more work needs to be done before having a data delivery with trustable QoS guarantees in such environment.

A topic-based pub/sub service at the Internet scale exhibits processes 1) scattered across different stub domains and 2) needed to communicate through several transit domains and according to the model introduced in subsection 2.1. As shown in the upper part of Figure 3, we propose a hierarchical approach to organize a pub/sub service, which reflects the previous considerations on the Internet topology: 1) a node runs only a single process of the pub/sub service[5], for simplicity; 2) nodes in the same domain are clustered together; and 3) each cluster holds a *coordinator* that allows interactions with the other clusters. Nodes in the same cluster communicated though an *intra-cluster routing*, and they can send messages outside the cluster only via their coordinator. In fact, a coordinator allows communications to the outside world exchanging messages with others coordinators though an *inter-cluster routing*. Since the Internet comprises multiple domains with dissimilar QoS features, as stated above, intra-cluster and inter-cluster routing can be designed separately.

3.2 Overlay Routing

Since clusters have been built grouping nodes that reside into the same stub domain, it is suitable to use IP Multicast as a mean to implement the intra-cluster routing. In fact, it is feasible that IP Multicast is provided by the network infrastructure in such domains[6]. Using IP Multicast, we can achieve efficiency both in term of wise use of network bandwidth and scalable support to a large number of nodes. Moreover, such domains usually exhibit low, or even negligible, probability that network faults happen, so it is viable to assume that data delivery into clusters is guaranteed.

Each cluster has to be connected to the other ones through its coordinator, and the various coordinators need to cooperate exchanging messages. In literature there are three possible architectural solutions to implement this cooperation:

- *Network-level multicast*, which is based on IP Multicast [6]. It has been demonstrated to be unsuitable for communications over Internet due to two main problems: 1) the lack of IP Multicast facilities widely deployed over Internet [20], and 2) the so-called *Reliability versus Scalability problem* [21]. In particular, network-level Multicast is unable to guarantee highly-reliable delivery to a large number of subscribers due to an ACK implosion and the centralization of the recovery duty in the multicaster, which represents a single-point-of-failure into the system.
- *Proxy-based overlay network*, which shifts multicast support from routers to end systems that are organized into a static overlay structure [22]. The main drawback of this architectural paradigm is the lack of self-* capabilities (i.e., self-organizing, self-configuring and self-healing) [23], *e.g.*, it needs a strong human intervention for its deployment [24].
- *Peer-to-Peer application-level multicast*, which implements multicast services on top of a peer-to-peer communication infrastructure. This solution provides the same scalability properties of a proxy-based overlay network, and in addiction, it enhances the system with self-* capabilities. Due to these properties, peer-to-peer application-level multicast is the most promising architecture to be used for the inter-cluster routing.

There is a considerable literature on per-to-peer application-level multicast [25], and the debate on-going in this field is what is the

[4]In fact none of the approaches introduced in subsection 2.3 deal with partitionable networks.

[5]Therefore, 'process' and 'node' are sometimes used as synonyms.

[6]For simplicity, we have assumed that Stub Domains offer the possibility to use IP Multicast, however, there may be cases where this is not necessarily true. In these cases, one of the several *Group Communication Toolkits* (GCT) available in literature [19] can be used instead of IP Multicast.

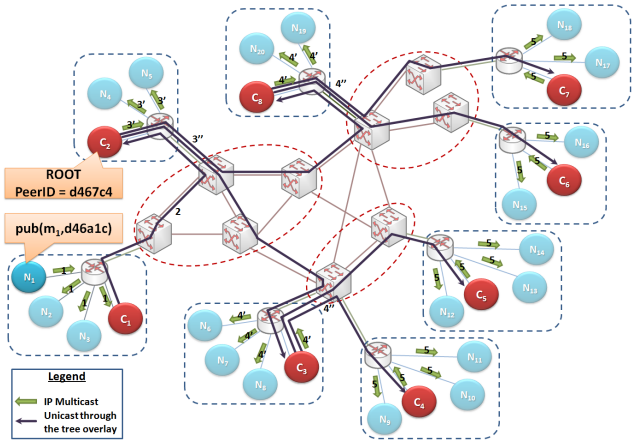

Figure 4: Event Dissemination

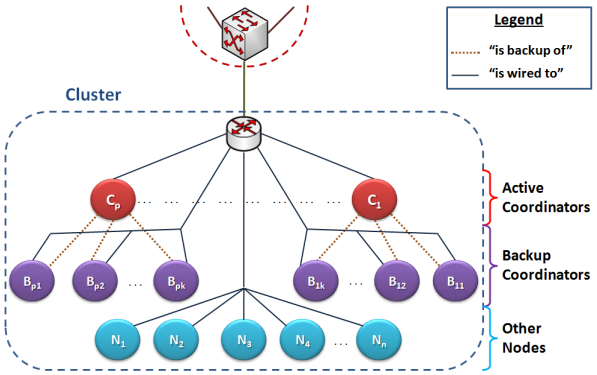

Figure 5: Replication Schema inside a cluster

right way, in term of performance and reliability, to structure the participant to a multicast session [26]. On one hand, there are *tree-based* approaches that organize the nodes into a tree, where each node can implicitly define its parent from which it receives the incoming messages. On the other hand, there are *mesh-based* approaches that expose a less structured organization letting each node to employ a swarming delivery mechanism to a certain subset of nodes. For two reasons our choice has been to adopt a tree-based solution, and to propose mechanisms to overcome its intrinsic weaknesses: tree-based approaches give direct control on the path followed by messages and present lower communication overhead [27]. In the proposed approach, the tree-base application-level multicast is built on top of a structured or Distributed Hash Table (DHT) overlay since it simplifies the tree construction. Among the available DHT solutions, we preferred the ones based on the *Plaxton Mesh* [28] data structure since it enforces a fast search operation (its complexity goes mostly around $O(log(N))$ hops, where N is the number of the peers into the overlay). Therefore, we have chosen to built our system on top of Pastry [29], however, other similar DHT overlays can be used since most of Plaxton-based DHTs do not present strong differences among each other. Moreover, we have drawn on the experience of Scribe [30] for the tree construction. Each peer in the system is univocally identified by a peerID and each multicast communication has a unique groupID. The coordinator whose peerID is numerically close to the groupID is selected as the root of the related multicast tree. A coordinator that wants to join an existent multicast-tree, because one of the peers in its cluster or itself is interested to receive messages from it, sends a join message to the root. Each coordinator along the way checkes if it is already in the tree: in the positive case, it registers the sender as a child and informs it to be its parent, otherwise, it forwards the message to the next coordinator on the way to the root. If a coordinator wants to send a message, because one of the peers in its cluster or itself has published a message, it forwards the message to the root, and the root would deliver the message though the layers of the tree.

Putting everything together, let consider an example of a node publishing a message of a given topic, as shown in Figure 4:

1. Node N_1 publishes the message m_1 with groupID equal to $d46a1c$, so a multicast message is sent to all the interested nodes in the cluster and to the coordinator;
2. The coordinator C_1 of the cluster receives the message m_1 and passes it to the root of the overlay tree;
3. The root C_2 exchanges the received message with all the interested nodes in his cluster, arrows (3') in Figure, and then forwards m_1 to its children, arrows (3") in Figure;
4. All the coordinators reached by the message of the root C_2 perform an IP Multicast in their cluster, arrows (4') in Figure, and then forward m_1 to their children, arrows (4");
5. The message is propagated though the layers of the tree until there are coordinators, such as C_4-C_7, with no children. Then, these coordinators multicast the received message into their cluster, and the delivery is concluded.

3.3 Fault-tolerance at the cluster level

The main weakness of the proposed hierarchical approach is that the entire system is vulnerable when a coordinator fails. In fact, the failure of a coordinator leads to both the isolation of the related cluster from the rest of the system and the disconnection of some coordinators from the rest of the overlay tree. Moreover, since the entire traffic toward the outside world passes through the coordinator, the consequent strong workload exacerbates its time to fail.

The trivial solution to this issue is to replicate the coordinator, however, the choice of which replication flavor to use is critical since it affects the quality of the system perceived by the end-users. On one hand, one option is to use a passive replication schema [31], in which the failed coordinator is replaced by one of its backups. The case that both the coordinator and all of its replicas fail at the same time is extremely rare, so this solution improves the availability. However, timeliness is compromised since the cluster would be isolated for a certain time window, *i.e.*, time to detect the failure of the coordinator and election of a new coordinator among the backups, so the overlay tree would be disconnected in some of its parts. On the other hand, an other option is to use an active schema [32], in which a cluster exposes a virtual coordinator made of a set of partners with equal responsibilities. Since there are several coordinators available at the same time, a failed coordinator is instantaneously replaced without isolating the cluster or affecting the overlay tree. So timeliness is achieved, however, availability may suffer of a possible failure of all the coordinators due to common mode errors. We propose a hybrid schema where the coordinator is actively p-redundant, *i.e.*, there are p coordinators active at the same time, moreover, there are k backups for each active coordinator. The system designer is free to choose the robustness of the cluster varying <p,k>. Such organization is illustrated in Figure 5 and constructed through a distributed bully election algorithm. This solution also has a positive impact on the overlay tree management. In fact, since the participants to the tree would result highly available, there is no

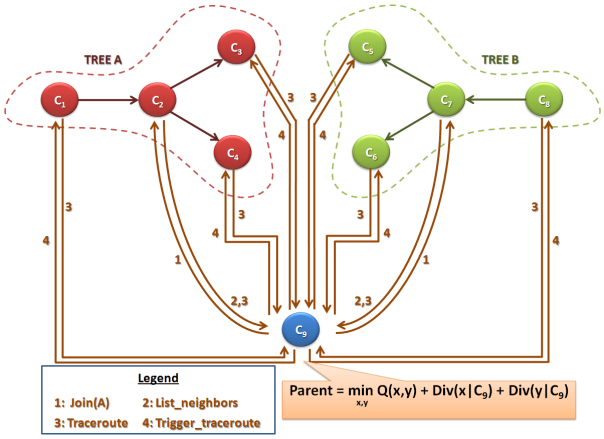

Figure 6: How a new node join two distinct trees

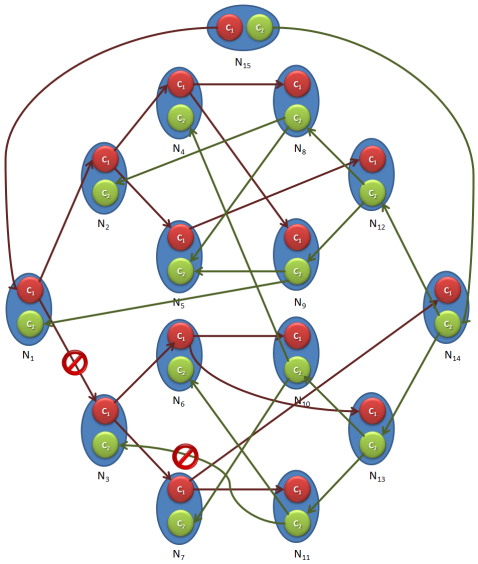

Figure 7: Event delivery among coordinators

need to cope with node crashes, avoiding traditional routing approaches that circumvent the failed element and compromise the timeliness of the message delivery.

3.4 Fault-tolerance at the overlay level

The proposed replication-based approach does not cope with link crashes, in fact, a cluster does not receive a message when the link connecting it to its parent in the multicast tree has crashed. Considering that several prior studies have demonstrated that Internet exhibits redundant connections at AS level [33], link crashes can be handled exploiting path redundancy. There are three alternative prospectives to implement such solution [16]: 1) *cross-link*, *i.e.*, connecting random peers via extra cross-cutting links; 2) *in-tree*, *i.e.*, establishing alternative links among different layers of the tree; and 3) *multiple-tree*, *i.e.*, creating several overlapping trees. While all of them have been demonstrated to improve the resiliency of the multicast service, we have chosen to adopt the latter approach since it is able to reduce delivery ratio and to cope better with stringent real-time deadlines [34].

As discussed in subsection 2.3, multiple-tree approach enforces reliability and timeliness under the condition to guarantee diversity among the paths composing the multiple trees. When a message has to reach a node from an other one, it travel through a path, consisting of a succession of network devices, *e.g.*, routers and/or switches. Given two paths P_1 and P_2, we can define a measure of their reciprocal diversity, namely $Q(P_1, P_2)$, as the number of the overlapping networked devices, and the two paths are diverse if $Q(P_1, P_2)$ is zero. The overlapping nodes can be identified thought measurements at path- and AS- level as described in [35]. Given such measure, it is possible to have two different formulations of path diversity. Given a forest of n trees, namely T_i with i = 1,...,n, such forest verifies the path diversity constraint if and only if two paths, taken from any of the n trees, that exhibit a positive value as measure of their reciprocal diversity do not exist (*Global Diversity*):

$$F = \bigcup_{i=1,\dots,n} (T_i) : F \; is \; diverse \Leftrightarrow$$
$$\nexists P_i, P_j \in F : (Q(P_i, P_j) > 0) \wedge (i \neq j).$$

In large scale networks, it is impossible to verify if trees satisfy this condition since it requires global knowledge on all the connections among the coordinators in the systems and their reciprocal diversity. Then, to use a multi-tree approach in a distibuted manner, we have to provide a different formulation of tree diversity: given a node N_A and n trees, namely T_i with i = 1,...,n, the forest of n trees verifies the path diversity constraint if and only if all the paths from N_A to its parents and children in the i-th tree, namely $P_{i|N_A}$ do not exhibit a positive value as measure of their reciprocal diversity (*Local Diversity*):

$$F = \bigcup_{i=1,\dots,n} (T_i) : F \; is \; diverse \Leftrightarrow$$
$$\forall N_A \; \nexists P_{i|N_A}, P_{j|N_A} : (Q(P_{i|N_k}, P_{j|N_k}) > 0) \wedge (i \neq j).$$

The local diversity does not imply the global diversity, however, it makes possible to implement a distributed algorithms that is able to construct locally-diverse multiple trees. Therefore, we have modified the joining procedure of Scribe so to construct multiple path-disjoint trees that satisfy the local diversity constraint. As illustrated in Figure 6, let consider that a new node C_9 wants to join two different trees, namely A e B. At the beginning, it sends two join messages through Scribe and two nodes of the systems, indicated in Figure as C_2 and C_7 that respectively have joined tree A and B, are contacted. Each one of these nodes replies C_9 with a message containing 1) a list of their neighbors in the tree to whom it belongs (*e.g.*, its parent and children) and details on their path, and 2) content of a traceroute on the path to C_9, *e.g.*, the list of the traversed network devices. Then, C_9 contacts all the nodes in each the received lists, and receives traceroute messages about the path to them, too. After collecting such informations, C_9 can make the decision on which will be its parent in each tree, according these two rules:

1. the paths from C_9 to its parents have to expose the lowest measure of diversity;
2. given a parent of C_9, the paths to its children have to maintain a measure of diversity closer to the value they had before the inclusion of C_9 as a child.

So the node C_9 decides on its parents by performing the following optimization:

$$\widehat{x} = \min_{\bar{x}, y=C_9} \left[\check{Q}(\bar{x}, y) + \sum_{x_i \in \bar{x}} Div(x_i|y) \right],$$

where $\bar{x} = \{x_1, x_2, ..., x_n\}$ is a list of the possible parents for C_9, while $\widehat{x}$ is the list of the chosen parents for C_9. Moreover, the first addend of the sum to be minimized formalizes the first rule,

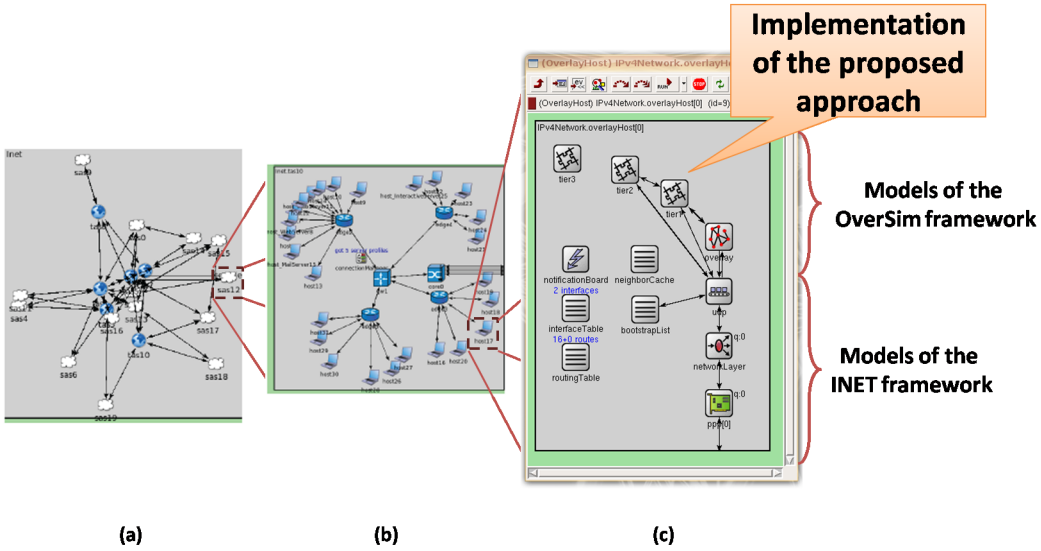

Figure 8: OMNET Model used in our study: a) topology at AS-level, b) topology at the router-level, and b) implementation of a node

where $\check{Q}(\bar{x}, y)$ measures the diversity of the paths from node y to the parents contained in vector $\bar{x}$, whose length is equal to n:

$$\check{Q}(\bar{x}, y) = \sum_{\substack{x_i, x_j \in \bar{x} \\ i \neq j}} Q(P(x_i, y), P(x_i, y)),$$

where the path from a node x to a node y is indicated as $P(x, y)$. While, the second addend formalizes the second rule and evaluates the variation of the diversity of the neighbors of a node x if x promoted as child of x:

$$Div(x|y) = \check{Q}(\{V_x \cup y\}, x) - \check{Q}(V_x, x),$$

where V_x is the list of the neighbors of node x before putting y in the children list. In the optimal case, due to the locality of Pastry, we will always find nodes that exhibit a diversity measure equal to zero, however, in the real case this is not always possible. Since, the achievable diversity is always lower than the intrinsic diversity of the topology at the network level, we acknowledge that there are cases where the minimum of the previous optimization is not zero.

Since clusters host multiple active coordinators, there is the problem to decide how performing the joining procedure. In order to avoid the case of coordinators of a same cluster exhibiting different dependencies in the same overlay tree, only one coordinator at time performs the joining procedure. Recalling the previous example, at the end of the join procedure, C_9 1) sends to its parents the IP addresses of its partners and backups, 2) receive back the IP address of partners and backups of its parents, and 3) updates with the received informations forwards its partners and backups. So, each active coordinator will be in charge of sending messages only through one tree, then the parameter p also defines the number of multiple trees that are established into the system.

Even using a multiple-tree approach, there are cases in which some nodes may experience message losses due to link crashes. As shown in Figure 7, since two links have crashed, node N_3 will never receive messages published by N_{15} even if it is not isolated. This is possible when all the inbound connections to the link have crashed[7]. However, the outbound connections are still correct, and they can be used to recover from this situation. Messages exchanged though different trees do not reach a node at the same time. Let consider node N_7 in Figure 7, it would receive a message before from N_3 and then from N_{10}. So, when N_7 receives a message from N_{10}, but nothing from N_3 before a timeout is expired, it can assume that the message has not reached N_3 and notifies it that he has received a message. So, N_3 knows that he has missed a message and asks to N_7 for its transmission. Since it has lost a message, it assumes that all his inbound connections are incorrect and executes the joining procedure to restore its connections to the trees.

4. SIMULATION STUDY

In this section we present the results of a simulation study to assess the quality of our approach. Instead using a real wide-area network, such as PlanetLab[8], that exhibits uncontrollable loss patterns and makes tests unrepeatable, we have conducted the study of our approach in a simulation environment called OMNET++[9]. Our choice is motivated by its ease of use, modular architecture, parametric approach and open-source code base. Moreover, OMNET++ is rapidly becoming a popular simulation platform in the scientific community as well as in industrial settings. In fact, it has an advantage over other existing simulators since, due to its generic and flexible architecture, it easily allows for the simulation of virtually any modular, event-driven system, and not just communication-network oriented systems.

OMNET++ follows a hierarchical architecture. At the lowest level of the hierarchy there are simple modules which encapsulate behavior of a given protocol or application. These simple modules are represented by C++ classes. A compound module may be composed of simple as well as other compound modules. Modules communicate with each other via message-passing. An event is said to have occurred whenever a module sends/receives a message. OMNET++ enforces code reuse since its community provides several third-parties models that modelers can easily include

[7]We do not treat the case in which all the connections to a node, inbound and outbound, are crashed, because in this case the node would be completely isolated. Since we have assumed that the network is not partitionable, this case can never happen.

[8]www.planet-lab.org

[9]www.omnetpp.org.

in their own model. An example is the INET framework, which models mostly all the protocols of wired, wireless and ad-hoc networks and several networking components, such as router, switches and access points. These models have parameters whose values are specified externally in an initialization file, and can be varied in different simulation runs. In the context of data dissemination protocols, these parameters can be used to simulate and analyze the effect of different network conditions on the performance of event delivery. Additional information about OMNET++ can be found in its User Manual[10].

4.1 Simulation Setup

In our tests we have used a two-level power-law topology with 300 nodes, 8 AS and 500 edges generated by a topology generator for OMNET++ called Rease[11] with default end-to-end delays [36]. Specifically, the AS-level topology, *i.e.*, how different ASs are interconnected each other, is based on the *Positive Feedback Preference* (PFP) model [37], shown in Figure 8(a), while the router-level topology, *i.e.*, how different routers and nodes in a given AS are interconnected each other, is based on the *Heuristic Optimal Topology* (HOT) model [38], illustrated in Figure 8(b). At each test, we have randomly selected a fixed number of nodes from the 300 IP-layer nodes as overlay nodes, among which only one at time is the publisher and all the others are subscribers. Moreover, we have 1) used the Scribe implementation provided by an overlay network simulation framework for the OMNeT++ called OverSim[12], modifying it according to our approach, as shown in Figure 8(c), 2) implemented an OMNET module to perform the path measurements as described in [35], and 3) made a patch to the INET framework in order to reproduce link crashes that do not partition the network. Node and link failures are uniformly distributed over the simulation time, and parameters have been derived from [39, 40]: while the duration of a link crash has a random distribution with a minimum duration of 10 seconds and a maximum duration of 1200 seconds, lastly, the time between link failures is reduced to a distribution with a mean equal to 5000 seconds. Lastly, the publishing rate is one message per second and the simulated duration of a single test is two hours.

4.2 Evaluation Results

The scope of the first test, shown in Figure 9, is to assess the scalability of the proposed approach, indicated in figure as Mod. Scribe, compared with the unmodified version of Scribe. It is possible to notice that clustering AS-related nodes improves the scalability of the approach, and the latency is affected only by the number of groups, while if we vary the number of nodes and leave the same number of groups, the trend of the latency is almost constant. We have studied the timeliness of the proposed approach in terms of the standard deviation of the delivery time, illustrated in Figure 9(b). With small nodes, Scribe shows better values of timeliness, but increasing the number of nodes involves a raise of the standard deviation, while the proposed approach exhibits a trend with lower increasing rate. Lastly, we have evaluated the improvement in terms of reliability, measured as the mean success rate, *i.e.*, the ratio of the received messages and the total published messages, shown in Figure 9(c). Scribe provides no means to guarantee a reliable event dissemination, so its success rate is low and related to the number of nodes in the tree, increasing the number of nodes causes a drop in the success rate. While, the proposed approach exhibits better reliability degree, which is not dependent on the total number of nodes.

[10] http://www.omnetpp.org/doc/manual/usman.html
[11] projekte.tm.uka.de/trac/ReaSE
[12] www.oversim.org/

5. RELATED WORK

The proposed approach is very close only to the one used in [41] since it also adopts broker replication and path redundancy. However, there are some main differences: 1) since the connection among the brokers is ring-based, it seams to be less scalable than the proposed approach that is tree-based, 2) even if paths are replicated in the overlay, the topology-awareness is not addressed at all, last 3) system initialization needs global knowledge into the system, which is impossible to have in Internet-scale pub/sub service, while we do not need such information.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have analyzed the available approaches to implement reliable pub/sub services and their effectiveness to the application scenario of time-sensitive application over Internet. Path redundancy has been shown to be an appealing approach, however, current solutions that apply this approach do not provide path diversity, which has to be satisfied to guarantee that all subscribers would receive all the messages. We have proposed a novel hybrid peer-to-peer approach that combines the replication of the coordinator in a cluster with multiple trees. We have performed a simulation study to assess the quality of the proposed solution in terms of scalability, timeliness and reliability.

We plant to make a more comprehensive simulation study to analyze in details the properties of the proposed solution under different network conditions. Moreover, multiple trees expose the drawback to generate considerable additional traffic, so we aim to apply Network Coding [42] in order to optimize the generated traffic and to achieve a wiser use of the network resources.

7. REFERENCES

[1] S. Bologna, C. Balducelli, G. Dipoppa, and G. Vicoli. Dependability and Survivability of Large Complex Critical Infrastructures. *Computer Safety, Reliability, and Security, Lecture Notes in Computer Science*, 2788:342–353, September 2003.

[2] P.Th. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, January 2003.

[3] Sun Microsystems. Java Message Service, v1.1. *SUN Specification*, 2002.

[4] Object Management Group. Data Distribution Service (DDS) for Real-Time Systems, v1.2. *OMG Document*, 2007.

[5] B. N. Levine and J. J. Garcia-Luna-Aceves. A Comparison of Known Classes of Reliable Multicast Protocols. *Proceedings of the 1996 International Conference on Network Protocols*, pages 112–121, 1996.

[6] Katia Obraczka. Multicast Transport Mechanisms: A Survey and Taxonomy. *IEEE Communications Magazine*, January 1998.

[7] C.L. Abad, W. Yurcik, and R.H. Campbell. A Survey and Comparison of End-System Overlay Multicast Solutions Suitable for Network-Centric Warfare. *Proceedings of SPIE conferences*, 5441:215–226, 2004.

[8] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

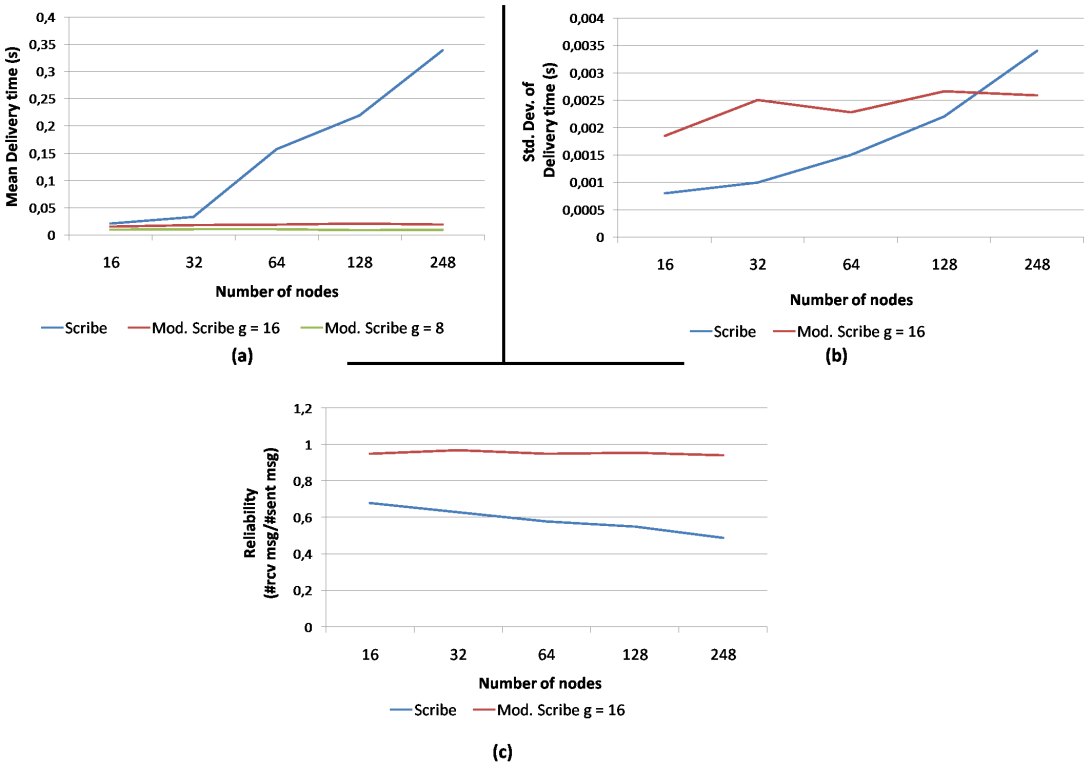

Figure 9: Results of the simulation study: a) Scalability, b) Timeliness, and c) Reliability

[9] Joe Hoffert, Douglas Schmidt, Mahesh Balakrishnan, and Ken Birman. Supporting Large-scale Continuous Stream Datacenters via Pub/Sub. *Proceedings of the Large-Scale Distributed Systems and Middleware Workshop (LADIS 08)*, 2008.

[10] Paolo Costa, Matteo Migliavacca, Gian Pietro Picco, and Gianpaolo Cugola. Introducing Reliability in Content-Based Publish-Subscribe through Epidemic Algorithms. *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS 03)*, pages 1–8, 2003.

[11] G. Muhl, M.A. Jaeger, K. Herrmann, T. weis, A. Ulbrich, and L. Fiege. Self-stabilizing Publish/Subscribe Systems: Algorithms and Evaluation. *Proceeding of the 11th International Euro-Par Conference, Lecture Notes in Computer Science*, 3648:664–674, 2005.

[12] Mohammad Reza Selim, Yuichi Goto, and Jingde Cheng. A Replication Oriented Approach to Event Based Middleware over Structured Peer to Peer Networks. *Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-hoc Computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference*, pages 61–66, 2007.

[13] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 11–20, 2001.

[14] Raphaël Chand. XNet: A Reliable Content-Based Publish/Subscribe System. *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS 03)*, pages 264– 273, 2003.

[15] J. Han and F. Jahanian. Impact of Path Diversity on Multi-homed and Overlay Networks. *Proceedings of the 34th International Conference on Dependable Systems and Networks (DSN 04)*, pages 29–39, 2004.

[16] S. Birrer and F.E. Bustamante. A Comparison of Resilient Overlay Multicast Approaches. *IEEE Journal on Selected Areas in Communications*, 25(9):1695–1705, December 2007.

[17] E.W. Zegura, K.L. Calvert, and S.Bhattacharjee. How to Model an Internetwork. *Proceedings of the 15th Annual Joint Conference of the IEEE Computer Societies on Networking the Next Generation (INFOCOM '96)*, 2:594–602, 1996.

[18] W. Zhao, D. Olshefski, and Henning Schulzrinne. Internet Quality of Service: An Overview. *Columbia University Research Report CUCS-003-00*, 2000.

[19] G.V. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: a Comprehensive Study. *ACM Computing Surveys (CSUR)*, 33(4):427–469, December 2001.

[20] C. Diot, B.N. Levine, B. Lyles, H. Kassan, and D. Balendiefen. Deployment issues for the IP Multicast services and architecture. *IEEE Network - Special Issue Multicasting*, 14:78–88, 2000.

[21] J.F. Rezende and S. Fdida. Scalability Issues on Reliable Multicast Protocol. *Proceedings of COST 237 Workshop*, 1999.

[22] L. Lao, J. H. Cui, and M. Gerla. TOMA: A Viable Solution for Large-scale Multicast Service Support. *Proceedings of the International IFIP-TC6 Networking Conference No4*, pages 906–917, 2005.

[23] E. Anceaume, A. K. Datta, M. Gradinariu, G. Simon, and A. Virgillito. A Semantic Overlay for Self-* Peer-to-Peer Publish/Subscribe. *Proceedings of the 26th IEEE International Conference on Distributed Computing Syste*, pages 22–31, 2006.

[24] P.R. Pietzuch and J. Bacon. Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware. *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS 03)*, pages 1–8, 2003.

[25] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, December 2004.

[26] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective. *Proceedings of the the 33rd IEEE Conference on Local Computer Networks*, pages 20–27, 2008.

[27] M. Hosseini, D.T. Ahmed, S. Shirmohammadi, and N.D. Georganas. A Survey of Application-Layer Multicast Protocols. *IEEE Communications Surveys & Tutorials - 3rd Quarter*, 9(3):58–74, 2007.

[28] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Proceeding of ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[29] A. Rowstrom and P. Drushel. Pastry: Scalable, Decentralized Object Localization and Routing for Large-scale Peer-to-Peer Systems. *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Lecture Notes in Computer Science*, 2218:329–351, 2001.

[30] M. Castro, P. Drushel, A.M. Kermarec, and A. Rowstrom. Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, January 2004.

[31] J. Li and S. Vuong. An Efficient Clustered Architecture for P2P Networks. *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA 04)*, pages 278–284, 2004.

[32] B. Beverly Yang and H-Garcia-Molina. Designing a Super-Peer Network. *Proceedings of the 19th International Conference on Data Engineering*, pages 49–60, 2003.

[33] R. Teixeira, K. Marzullo, S. Savage, and G.M. Voelker. In Search for Path Diversity in ISP Networks. *Proceedings of the 3rd ACM SIGCOMM Internet Measurement Conference (IMC 03)*, pages 313–318, 2003.

[34] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective. *Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN 2008)*, pages 20–27, 2008.

[35] Z.M. Mao, J. Rexford, J. Wang, and R.H. Katz. Toward an accurate AS-level Traceroute Tool. *Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 365–378, 2003.

[36] T. Gamer and M. Scharf. Realistic simulation environments for ip-based networks. *Proceedings of 1st International Workshop on OMNeT++*, 2008.

[37] H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, A. Jamakovic, and M. Rio. Tuning topology generators using spectral distributions. *Proceedings of the International Performance Evaluation Workshop (SPEC 08), In Lecture Notes in Computer Science*, 5119, 2008.

[38] L. Li, D. Alderson, W. Willinger, and J. Doyle. A First-principles Approach to Understanding the Internet's Router-level Topology. *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 3–14, 2004.

[39] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.N. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 04)*, 99(7):7–11, January 2004.

[40] B. Chandra, M. Dahlin, L. Gao, and Amol Nayate. End-to-end Wan Service Availability. *IEEE/ACM Transactions on Networking*, 11(2):300–313, 2003.

[41] H. Jafarpour, S. Mehrotra, and N. Venkatasubramanian. A Fast and Robust Content-based Publish/Subscribe Architecture. *Proceedings of the 7th IEEE International Symposium on Network Computing and Applications (NCA 08)*, pages 52–59, 2008.

[42] M. Ghaderi, D. Towsley, and J. Kurose. Network Coding Performance for Reliable Multicast. *Proceedings of IEEE Military Communications Conference*, pages 1–7, October 2007.