2011 31st International Conference on Distributed Computing Systems

Delay-Cognizant Reliable Delivery for Publish/Subscribe Overlay Networks

Shuo Guo
University of Minnesota
Minneapolis, Minnesota 55455
Email: sguo@umn.edu

Kyriakos Karenos*
Microsoft
London, United Kingdom
Email: kyriak@microsoft.com

Minkyong Kim, Hui Lei, and Johnathan Reason
IBM T. J. Watson Research Center
Hawthorne, New York 10532
Email: {minkyong, hlei, reason}@us.ibm.com

***Abstract*—The number of real-world applications that require QoS guarantees is constantly increasing and they often follow the publish/subscribe (pub/sub) messaging paradigm, which provides loosely coupled many-to-many communication. Many QoS-aware systems use overlay networks as they allow flexible routing. To provide QoS-aware pub/sub messaging in overlay networks, the messaging system should be adaptive to the changes in network conditions (such as delay and failures). However, many pub/sub systems depend on a fixed routing topology and it is costly to rebuild this topology in case of failures. This study seeks to address this challenge with Delay-Cognizant Reliable Delivery (DCRD), a novel and delay-aware dynamic routing algorithm to provide reliable message delivery for pub/sub overlay networks. For reliable message delivery, DCRD no longer uses a fixed routing topology. Instead, it dynamically switches among different links to bypass link failures and increase the chance to meet QoS requirement. Each node tries different neighboring nodes in an order that is mathematically proven to minimize the expected delay of packet delivery. With all possible neighboring nodes sorted this way, DCRD guarantees that packets are delivered as long as there exists a path between the publisher and subscriber and that the expected delay is minimized. DCRD is extensively evaluated in simulation with comparison to existing tree-based routing approaches as well as a multipath approach using different network topologies, delay constraints, and loss probabilities. Simulation results show that DCRD performs better than all the baselines, providing reliable message delivery and satisfying the delay requirement for more than 98% of messages when the link failure probability is 4% or less.**

I. Introduction

As distributed computing applications that involve event-based response to real world sensing become more common, there has been increasing emphasis in managing the end-to-end performance of message delivery, particularly over a wide-area network (WAN). One approach to managing the performance of message delivery in a WAN is to use an overlay network, which provides a messaging substrate for applications, but overlays the physical network. Two common messaging paradigms are message queuing [1] and publish/subscribe (pub/sub) messaging [2]. Message queuing provides buffering between pairs of hosts over point-to-point paths, an approach that is most suitable for applications requiring persistence, but less amenable to applications requiring latency awareness. Pub/sub messaging follows a many-to-many communication pattern, allowing a decoupling between publishers and subscribers. It is this decoupling nature of pub/sub that makes it more suitable for applications requiring latency awareness and adaptation to failures because it allows for rapid route re-configuring when adverse network conditions are detected.

*This work was done while this author was at IBM Research.

By exploiting the decoupled nature of pub/sub messaging and the performance management benefits of overlay networking, a messaging system that is both flexible and able to manage performance can be achieved. However, simultaneously managing latency, per some latency requirement, and providing reliable message delivery is still a challenge. Existing techniques for providing reliable delivery, such as topology recovery and retransmissions, provide very limited control on delay requirement. For example, the Resilient Overlay Networks (RON) [3] approach detects and recovers from path outages within several seconds by periodically monitoring and updating the quality of Internet paths. It decides the route for packets based on the monitoring result. However, packets that are already experiencing a path outage during the interval of two updates have little chance to be delivered on time. Even with the help of retransmissions, it is still hard for packets to get through a failed or highly congested link, leaving the delay management problem unsolved. Similarly, standard approaches to timely delivery of messages, such as priority-based queuing and shortest path tree, do not simultaneously consider reliable delivery of messages. Although some designs, such as the multipath approach [4] and Forward Error Correction (FEC) [5], propose to improve the reliability by adding redundancy into the information transmitted, at the cost of introducing more traffic into the network, they still do not guarantee delivery since they use a fixed set of routing paths; they fail to deliver in case when path outage and congestion affect all the selected paths.

This study seeks to address the challenge of simultaneously managing latency performance and reliable delivery of messages. We propose Delay-Cognizant Reliable Delivery (DCRD), a novel and delay-aware dynamic routing algorithm to provide reliable message delivery for pub/sub overlay networks. DCRD no longer uses a fixed routing topology. Instead, it dynamically switches among different links to bypass link failures online and increase the chance to meet QoS requirement. Each node tries different neighboring nodes in an order that is mathematically proven to minimize the expected delay of packet delivery. With all possible neighboring nodes sorted this way, DCRD guarantees that packets are delivered as long as there exists a path between the publisher and subscriber and that the expected delay is minimized. In summary, we have the following contributions: 1) a novel, delay-aware, dynamic routing algorithm DCRD which considers both delay requirement and reliable delivery when determining the next broker hop in the overlay, and the next-hop decision is based on local information only, providing a means for forwarding brokers to rapidly adapt to changing network conditions; 2) a proof showing DCRD to provide the optimal solution in terms of delay performance under the problem formulation; and 3) extensive simulation results that show DCRD outperforms common tree-based approaches as well as the multipath approach. Compared to common tree-based approaches and the multipath approach, we show that DCRD provides more comprehensive performance management, especially when considering latency requirements.

The rest of the paper is organized as follows. Section II summarizes the related work. The design and analysis of DCRD is described in Section III. Section IV presents the performance evaluation of DCRD under different loss and link failure conditions. Section V concludes the paper.

II. Related Work

Challenges associated with multi-destination reliable delivery have been addressed extensively in the literature. Methodologies are broadly categorized into protocols and techniques implemented at the lower layers of the IP stack (e.g., IGMP) and those implemented at the application layer, where DCRD falls into the latter category. Application layer multicast is significantly more popular due to its flexibility of design, as well as the ability to provide differentiated QoS. Overlay based multicast techniques mostly focus on the problem of constructing a multicast tree to deliver content from one or more source to multiple destination. The authors in [6] provide a survey of techniques on constructing a multicast tree, while [7], [8] focus on the problem of allowing nodes to join and leave the multicast tree efficiently. In [8] the authors use a minimum latency tree to reduce delay. However, unlike DCRD, it does not attempt to meet any delay requirements. Furthermore, static tree-based approaches suffer from the fact that failures across the overlay links would require reconstruction, adding undesirable delay. The authors of [9] also construct a multicast tree, but the overlay links are constructed over TCP rather than UDP, and thus it specifically focuses on the throughput maximization.

Due to the increase in the usability of the pub/sub paradigm, the functionality of overlay multicast technique has been extended in several papers [10]–[12]. The goal of these techniques, unlike DCRD, is to construct a topology that is content-aware, meaning that it connects publishers and subscribers based on the content rather than topics. In this respect, these techniques put less weight on satisfying delay requirements. The authors in [13] address failures in a pub/sub network based on a peer-to-peer multi-ring approach similar to [14]. While latency using multiple rings can be reduced, there is still no explicit attempt in these approaches to meet particular delay requirements of applications.

To handle network failures, overlay networks have been used in [3], [4], which propose utilizing indirect overlay paths to avoid a failed underlay link. Furthermore, the authors of [15] address the failures in the context of multicast and propose a tree construction methodology that minimizes the correlation between nodes. Although these techniques improve the resiliency of the delivery overlay, the overlays proposed in these papers are not designed to provide reliable or delay-aware delivery.

III. Design

In this section we present the design of DCRD for an overlay-based pub/sub system. With link failures and congestions unpredictably occurring at overlay links, a fixed delivery structure in traditional solutions [7], [8] that utilize a pre-constructed multicast tree is not efficient. When a link in the multicast tree fails, the tree needs to be rebuilt and this cost is high. Thus, packets are not likely to be delivered within the QoS delay requirements. To solve this problem, DCRD does not use a fixed delivery structure. Instead, a node tries sequentially all its neighboring nodes, who are expected to deliver the packets within the delay requirements. If the node does not get an ACK from a neighboring node, it first retransmits the packet. The loss of packet could be due to the link being temporarily unavailable or the link has failed for a longer time period (called *persistent* failures). By retransmitting the packet, the packet may be delivered successfully in case of former. However, it cannot succeed in case of the latter. If the re-transmission is unsuccessful, the node then tries another neighbor.

If a node fails to deliver the packet after trying all neighboring nodes, it reroutes the packet by sending it to its upstream node and the upstream node tries the next neighboring node to deliver the packet. Conceptually, this recursive approach tries its subtree of paths before sending the packet to the upstream node. In effect, it finds an alternate path close to the source of the problem and also close to the destination. The reason why we need to send the packet to the upstream node is that each node aggressively deletes a copy of packet once it receives an ACK from its downstream neighbor to minimize storage usage. However, this becomes a problem when the packet cannot be delivered through its downstream neighbors. In this situation, this node reroutes the packet to its upstream node, which similarly tries to deliver the packet by sending the packet to the next neighboring node.

In summary, because the route is chosen dynamically based on the current network conditions, our DCRD approach can significantly increase the chance of delivering the packet

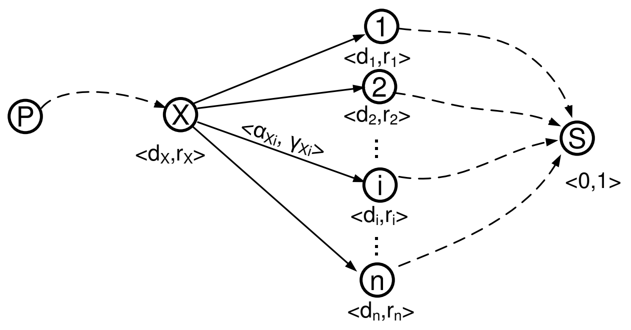


Figure 1. Recursive Computation Process

within the QoS delay requirement. As we consider all possible neighbors, DCRD—using the dynamic rerouting and the ACK mechanism—guarantees that packets are delivered as long as a path (without persistent failures) exists from the publisher and subscriber. To provide the delivery guarantee even in case of persistent failures, we need to persist (by storing in the persistent storage) all packets, and then send them when the failures are recovered. Supporting the persistency mode should be straight forward, but this mode incurs a large overhead. Thus, we do not further discuss the persistency mode in the rest of this paper.

A. Problem Formulation and Design Overview

In an overlay-based pub/sub system with N nodes, the latency, $\alpha_{ij}^{(1)}$, and deliver ratio, $\gamma_{ij}^{(1)}$, for a single transmission through each overlay link $i \to j$ can be collected through either link monitoring or online measurements. Then the expected delay $\alpha_{ij}^{(m)}$ and the expected delivery ratio $\gamma_{ij}^{(m)}$ based on m transmissions can be calculated as

$$\begin{aligned} \alpha_{ij}^{(m)} &= \frac{\sum_{k=1}^{m}(k\alpha_{ij}^{(1)})\gamma_{ij}^{(1)}(1-\gamma_{ij}^{(1)})^{k-1}}{1-(1-\gamma_{ij}^{(1)})^m} \\ \gamma_{ij}^{(m)} &= 1-(1-\gamma_{ij}^{(1)})^m \end{aligned} \quad (1)$$

Noting that the computation of $\alpha_{ij}^{(m)}$ has an implicit condition that the packet is delivered in m transmissions (otherwise the delay is infinite and the expected delay is unavailable). In Eq.1, the numerator of $\alpha_{ij}^{(m)}$ is the sum of the product of the delay that the transmission succeeds at the kth transmission and the corresponding probability, for $1 \leq k \leq m$; the denominator of $\alpha_{ij}^{(m)}$ is $\gamma_{ij}^{(m)}$, which is simply 1 minus the probability that all the m transmissions fail. For short, we replace $\alpha_{ij}^{(m)}$ and $\gamma_{ij}^{(m)}$ with α_{ij} and γ_{ij}. Node i is said to be failed to deliver the packet to j if i does not receive an ACK from j in m transmissions.

Suppose Subscriber S subscribes to a topic published by Publisher P and specifies a delay requirement, D_{PS}. For any node X in the network, the delay requirement from X to S, D_{XS}, is equal to or less than D_{PS} minus the shortest delay from P to X.

Define the *sending list* of node X for subscriber S as a sorted list of neighboring nodes that are expected to deliver the packet to S with the shortest expected delay and the highest probability to meet the delay requirement D_{XS}. Figure 1 shows an illustration of DCRD where node X has n nodes on the sending list labeled from 1 to n, towards subscriber S. We assume that the order of these nodes on the sending list is identical to the node label: node 1 is on the top of the list and node n is on the bottom of the list. Each packet contains the information of not only the destination subscribers but also the broker IDs that have been on the routing path. All the n nodes on the sending list of X towards subscriber S are possible next-hop nodes as long as they have not been on the routing path. When X receives a packet, it tries node 1 first since it is at the top of the list. If X fails to deliver the packet to node 1 (in m transmissions), or X receives the packet back from node 1 (indicating its failure to deliver the packet to S), X sends the packet to node 2. All the brokers that were on the routing path are recorded in the routing path information of the packet. The same process continues until the packet reaches S, or X has tried all n nodes. For the latter case, X reads the routing path information of the packet and sends the packet back to the upstream node from which it received this packet. The upstream node running the same DCRD algorithm tries to reach the destination by sending the packet to the next node on the sending list.

Before showing how to construct the sending list and the detailed routing strategy, we first define the two parameters of expected delay, d and the expected delivery ratio, r.

1) d_X: the expected delay from the time X receives a packet until this packet arrives at the subscriber S.
2) r_X: the expected deliver ratio that node X delivers the packet to S with the expected delay d_X.

Again d_X is conditional, and is calculated under the assumption that the packet is finally delivered to S. Otherwise, the delay is infinite and d_X is not available.

In the next subsection, we focus on the recursive process to compute $< d_X, r_X >$ in DCRD. We present how to build the sending list and decide the order of nodes in III-C and the detailed DCRD routing strategy in III-D.

B. Recursive Computation Process

The recursive computation process starts when a subscriber S subscribes to a topic from publisher P. Initially, the parameters of node S regarding S itself is $< 0, 1 >$ since S is already the destination, and has probability 1 to reach the destination with 0 delay. S then shares its parameters $< 0, 1 >$ with its immediate neighbors. Other nodes who have received the parameters regarding subscriber S from its neighbors start the computation of its own $< d, r >$ distributively.

We again use the example in Figure 1 where X has n neighbors on the sending list labeled from 1 to n with node 1 on top of the list and node n on the bottom. Given the parameters $< d_i, r_i >$ of neighboring node i and the parameters α_{Xi}, γ_{Xi} of link $X \to i$, for $1 \leq i \leq n$, we define $< d_X^i, r_X^i >$ as the expected delay and the expected delivery ratio for node X to reach subscriber S via neighbor

i, respectively. For the ith neighbor, $< d_X^i, r_X^i >$ can be calculated as follows:

$$\begin{aligned} d_X^i &= \alpha_{Xi} + d_i \\ r_X^i &= \gamma_{Xi} r_i \end{aligned} \tag{2}$$

where, d_X^i is the sum of the link delay α_{Xi} and the expected delay d_i from node i to S; r_X^i is the product of γ_{Xi} and r_i.

Given the computation result of $< d_X^i, r_X^i >$ for any neighbor i on the sending list, node X finally calculates its own parameters d_X, r_X. Given the sending list of $\{1, 2, \cdots, n\}$, X tries node 1 first and the packet is delivered with expected delay d_X^1 and probability r_X^1; if it fails, X tries node 2 and the packet is delivered with expected delay $d_X^1 + d_X^2$ and probability $(1 - r_X^1) r_X^2$, and so on. The general equation for X to calculate $< d_X, r_X >$ is given below:

$$\begin{aligned} d_X &= \frac{\sum_{i=1}^n [(\sum_{j=1}^i d_X^j)(r_X^i \prod_{j=1}^{i-1} (1 - r_X^j))]}{1 - \prod_{i=1}^n (1 - r_X^i)} \\ r_X &= 1 - \prod_{i=1}^n (1 - r_X^i) \end{aligned} \tag{3}$$

where the numerator of d_X is the sum of the product of delay and the corresponding probability that the packet is delivered via neighbor i; the denominator of d_X is r_X, which is the probability that the packet is delivered successfully via at least one neighbor.

Using Eq.3, each node can calculate its $< d, r >$ parameters as long as it receives the parameters from its neighbors and decides the sending list. It is easy to find that Eq.3 has linear computation complexity $\Theta(n)$ with n being the total number of neighbors on the sending list.

C. Construction of Sending List

Suppose X has a number of neighbors, and the ith neighbor's parameter regarding subscriber S is computed as $< d_i, r_i >$, which is shared with X. For any neighbor i, X selects this neighbor on its sending list towards subscriber S only if $d_i < D_{XS}$, i.e., X only includes those nodes on the sending list towards subscriber S who are expected to deliver the packets to S within the delay requirement. It can be seen from Eq.3 that the ordering of the nodes on the list does not affect the delivery ratio r_X, but it affects the computation of d_X which affects the performance significantly since d_X is the expected time that node X delivers the packet to S. It is thus important to sort the nodes in a way that can minimize d_X so that the expected delay is reduced. Such an optimal node ordering is given by Theorem 1:

Theorem 1 *Suppose node X has n neighbors on the sending list towards subscriber S which are labeled from 1 to n. The expected delay d_X^i and expected delivery ratio r_X^i from X to S via neighbor i can be calculated by Eq.2. Suppose $\frac{d_X^1}{r_X^1} \le \frac{d_X^2}{r_X^2} \le ... \le \frac{d_X^n}{r_X^n}$. Then {1,2,...,n} is the sending list that minimizes the expected delay d_X.*

Proof: Suppose {1,2,...,n} is already the optimal node sequence that minimize the expected delay d_X. We prove that $\frac{d_X^1}{r_X^1} \le \frac{d_X^2}{r_X^2} \le ... \le \frac{d_X^n}{r_X^n}$ is both sufficient and necessary condition for the optimality of {1,2,...,n}.

From Eq.3 we see that the ordering of nodes affects numerator only. We use Y to denote the numerator and expand it as follows:

$$\begin{aligned} Y &= d_X^1 r_X^1 + (d_X^1 + d_X^2)(1 - r_X^1) r_X^2 + \cdots + \\ &\quad (d_X^1 + d_X^2 + ... + d_X^n)(\prod_{i=1}^{n-1} (1 - r_X^i)) r_X^n \\ &= d_X^1 r_X^1 + (d_X^1 + d_X^2)(1 - r_X^1) r_X^2 + \cdots + \\ &\quad (d_X^1 + d_X^2 + ... + d_X^n)(\prod_{i=1}^{n-1} (1 - r_X^i))[1 - (1 - r_X^n)] \\ &= d_X^1 + d_X^2 (1 - r_X^1) + \cdots + d_X^n (\prod_{i=1}^{n-1} 1 - r_X^i) \\ &\quad -(\sum_{i=1}^n d_X^i) \prod_{i=1}^n (1 - r_X^i) \end{aligned} \tag{4}$$

Since {1,2,...,n} is already the sequence that minimizes d_X, and thus minimizes Y, switching the position of node k and $k+1$ in the sequence for any $1 \le k \le n-1$ should yield an equal or larger d'_X and Y'. Comparing the difference on Eq.4 yields:

$$\begin{aligned} & d'_X - d_X \ge 0 \Leftrightarrow Y' - Y \ge 0 \\ \Leftrightarrow \quad & \prod_{i=1}^{k-1} (1 - r_X^i)[d_X^{k+1} + (1 - r_X^{k+1}) d_X^k \\ & -d_X^k - (1 - r_X^k) d_X^{k+1}] \ge 0 \\ \Leftrightarrow \quad & r_X^k d_X^{k+1} - r_X^{k+1} d_X^k \ge 0 \\ \Leftrightarrow \quad & \frac{d_X^{k+1}}{r_X^{k+1}} \ge \frac{d_X^k}{r_X^k} \end{aligned} \tag{5}$$

Since Eq.5 holds for any $1 \le k \le n-1$, the condition $\frac{d_X^1}{r_X^1} \le \frac{d_X^2}{r_X^2} \le ... \le \frac{d_X^n}{r_X^n}$ is both sufficient and necessary for the optimality of sequence {1,2,...,n} for minimizing d_X.

■

D. Dynamic Routing

Based on the distributed computation process in III-B and the sending list construction in III-C, each broker follows Algorithm 1 to set up the sending list for subscriber S. Initially, the sending list of X for S is empty (Line 1). After receiving $< d, r >$ parameters from neighbors (Line 2), the ones whose expected delay can meet the delay requirement are added into the sending list and the corresponding expected delay and delivery ratio to S via these nodes $< d_X^i, r_X^i >$ are calculated based on Eq.2 (Line 3 to Line 8). Then the nodes on the sending list are sorted in increasing order of $\frac{d_X^i}{r_X^i}$ (Line 9). Based on the sending list, X finally computes its $< d_X, r_X >$ and shares them with its neighbors (Line 10 and 11).

Algorithm 1 Dynamic Routing Setup at Node X

```
1: List ← $\emptyset$
2: Receiving $< d, r >$ parameters from $l$ neighbors for $S$
3: for $i \leftarrow 1$ to $l$ do
4:   if $d_i < D_{XS}$ then
5:     Add $i$ into List
6:     Calculate $d^i_X, r^i_X$ based on Eq.2
7:   end if
8: end for
9: Sort the nodes in List in increasing order of $\frac{d^i_X}{r^i_X}$
10: Calculate $d_X, r_X$ based on Eq.3
11: Share $< d_X, r_X >$ with all neighbors
```

Each node uses the sending list for routing the packets toward each subscriber. To avoid possible forwarding loops, (e.g., two brokers are on the sending list of each other), each packet contains not only the information of multiple subscribers but also the brokers that has been on the routing path. When node X receives a packet from a neighbor, it finds the next-hop receiver from the sending list of each subscriber. A node is selected to be the next-hop receiver from the sending list if and only if it is the first node in the node sequence that has not been on the routing path for this packet. If a node is the next-hop receiver on the sending lists of multiple subscribers, only one copy of the packet is sent.

The hop-by-hop acknowledgement (ACK) is used in DCRD. With hop-by-hop ACKs, senders immediately know the reception status of its neighboring nodes and can switch to other neighbors on the sending list quickly if the packet is delayed or lost, increasing the probability of delivering the packet in time. The expected link delay α_{ij} can be used for a sender i to wait for the ACK before sending the packet to the next node on the sending list. If a node has tried all the nodes on the sending list but the packet is still not delivered, it adds itself into the routing path information of the packet and reroutes the packet to the upstream node from which it receives the packet. The upstream node can be easily found by checking the routing path information recorded in the packet, and thus there is no need to store the state information for every packet at each node. The upstream node then finds the new next-hop node on its sending list (the first node that has not been on the routing path), trying to deliver the packet via the new next-hop node.

Algorithm 2 shows the forwarding scheme implemented at node X. Upon receiving a packet from the immediate neighbor Y (Line 1), X first sends an ACK back to Y (Line 2), and gets the l destinations $S_1...S_l$ (Line 3). Initially the processing flags for all subscribers are set to 0 (Line 4 to 6). X finishes processing S_i ($flag[i] = 1$) if and only if S_i receives an ACK from the downstream next-hop receiver of S_i, or X has tried all nodes on the sending list but fails to deliver the packet. For the latter case X reroutes the packet to its upstream node obtained from the routing path information of the packet.

Algorithm 2 Dynamic Routing Scheme at Node X

```
1: Receive a packet from $Y$
2: Send ACK to $Y$
3: Get the destinations $S_1...S_l$
4: for $i \leftarrow 1$ to $l$ do
5:   $flag[i] \leftarrow 0$ //$flag$ is 1 if $X$ finishes processing $S_i$
6: end for
7: while ($\exists S_i$ s.t. $flag[i] = 0$) do
8:   for (any $S_i$ s.t. $flag[i] = 0$ and no ACK awaiting ) do
9:     $k \leftarrow$ the first qualified node on the sending list of $S_i$
10:    if ($k$ not found) then
11:      $k \leftarrow$ upstream node of $X$ from the routing path
12:    end if
13:    $dest \leftarrow \{S_i\}$ //add $S_i$ into the destination
14:    for $j \leftarrow 1$ to $l$ do
15:      if (($flag[j] = 0$) and (next-hop node is also $k$)) then
16:        //find other subscribers who has the same next-hop node as $k$
17:        $dest \leftarrow \{dest, S_j\}$ //add $S_j$ into destination
18:      end if
19:    end for
20:    Add $X$ into the routing path
21:    Send packet to $k$ with destinations $dest$
22:    Set a timer and cache the packet for $\alpha_{Xk}$ of time
23:    if (ACK received from $k$ before timer expires) then
24:      for (any subscriber $S_j$ in $dest$) do
25:        $flag[j] \leftarrow 1$
26:      end for
27:    end if
28:  end for
29: end while
```

After initializing the flags, X starts to find the next-hop downstream receivers from the sending lists of the subscribers that still needs to be processed (Line 7 to 8). We use k to denote the downstream next-hop receiver found for S_i (Line 9). A qualified k should be the first node on the sending list of S_i that has not been on the routing path. If there is no qualified next-hop node k found for S_i, meaning S_i cannot be reached through X, X has to reroute the packet to its upstream node, which is also denoted as k (Line 10 to Line 12). It is worth noting that Y can be either the upstream node of X such that X has not been on the routing path before receiving this packet from Y, or the downstream node of X that Y fails to deliver the packet to S and reroutes the packet back to X. As a result, X needs to read the routing path information to find what its upstream node is. After k is found as either the downstream node from the sending list or the upstream node from the routing path, S_i is added to the destination information of the new packet to be sent to k (Line 13). X further checks if other destinations use k as the next-hop receiver and adds them into the destination information if they do (Line 14 to 19). Then X adds itself into the routing path (Line 20) before sending a packet to k (Line 21) with the destination information including all nodes for which k is their next-hop receiver. X caches the packet and waits for an ACK from k acknowledging the reception of the packet at k. If there is an ACK received in α_{Xk} of time, X sets the processing flag of all destinations covered by this packet to 1 (Line 23 to 26). Otherwise the "while" loop continues until all the l subscribers are processed with flag set to 1.

IV. Evaluation

This section shows the performance evaluation of DCRD in simulations with various settings.

A. Simulation Setup

We evaluate the performance of DCRD in various topologies. Our overlay network consists of 20 broker nodes. For a given link degree, we randomly choose the neighboring nodes. For each link, we randomly choose the delays ranging from 10 ms to 50 ms; this range is chosen based on the AT&T measurements [16]. To simulate dynamic network environment, we change the network condition once every second, i.e., we inject link failures into randomly chosen links that will cause one second of packet loss. Note that each node monitors network conditions only every 5 minutes, while the network conditions change more frequently. We repeat the experiments with different failure probability, P_f, ranging from 0 to 0.1, incremented by 0.02.

Since DCRD is designed with the intention to deal with link failures, we set the packet loss rate, P_l, of each link in most simulations to 0.0001, which is relatively small compared with P_f. This setting helps evaluate how DCRD performs when it reroutes the packet and bypasses failures. The value of m, which is the number of transmissions a node tries before switching to another node, is set to 1 since with a much higher P_f, a missing ACK has higher probability to be caused by a failure than a packet loss. We provide evaluation for P_l and m in Section IV-D7.

Our publishers and subscribers are deployed on 20 broker nodes. We use 10 topics and deploy 10 publishers on 10 randomly chosen broker nodes. (In a real deployment, it is likely that publishers and subscribers are on the nodes other than the broker nodes and join the network by connecting through a nearby broker node. In our simulation, we place publishers and subscribers on the broker nodes for simplicity.) Each publisher sends packets at the rate of 1 packet/s. This is a typical rate in air surveillance networks, where pub/sub systems deliver messages. For example, in ADS-B, an aircraft (publisher) broadcasts its location once per second on average [17]. To place subscribers, we first randomly choose a probability, P_s, in the range of 0.2 and 0.6 for each topic. Then, a subscriber for this particular topic is placed at a node with the probability of P_s. In QoS-aware pub/sub systems, a subscriber specifies the delay requirement. As a hint, the system may provide the typical delay from the publisher to subscriber, and the subscriber then may choose the requirement based on the typical delay. For our simulation, we set the delay requirement to be the three times the shortest-path delay between the publisher and the subscriber.

We repeat our experiments in 10 different topologies. Each experiment has a simulation time of two hours.

B. Comparison

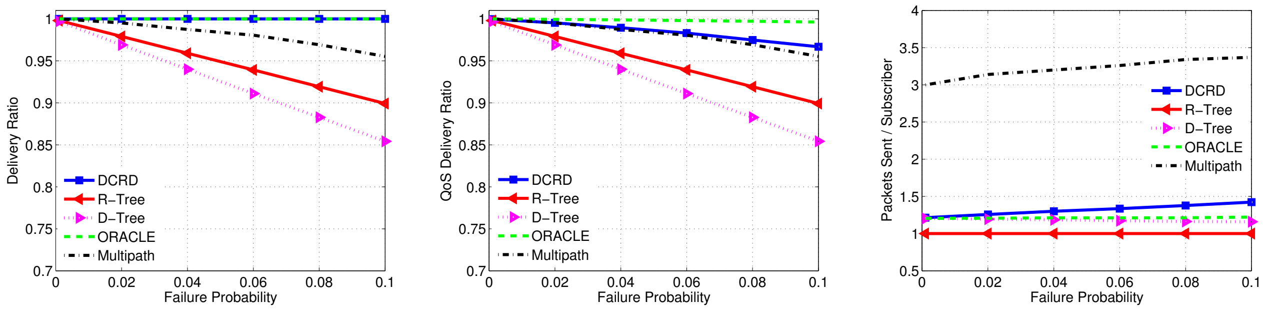
There are many pub/sub systems using tree-based topologies as routing solutions [7], [8], [18], [19]. We compare the performance of DCRD with two typical trees, a multipath approach as well as an oracle performance tree that provides the performance upper bound, as described as follows:

1) Most Reliable Tree (R-Tree): routing tree with the shortest-hop-count path between each publisher and subscriber
2) Shortest-Delay-Path Tree (D-Tree): routing tree with the shortest-delay path between each publisher and subscriber
3) Oracle Performance Tree (ORACLE): routing tree with the shortest-delay path avoiding any failures since the condition of entire network is known. This oracle (or optimal) solution provides the performance upper bound.
4) Multipath Solution (Multipath): the design in which publishers send duplicate packets for every subscriber to increase the chance of sucessful delivery. In our simulation, a single packet to a single subscriber is sent through two paths: one shortest delay path and another path that selected from the top 5 shortest delay paths that has the fewest overlapping links with the shortest delay path.

C. Performance Metrics

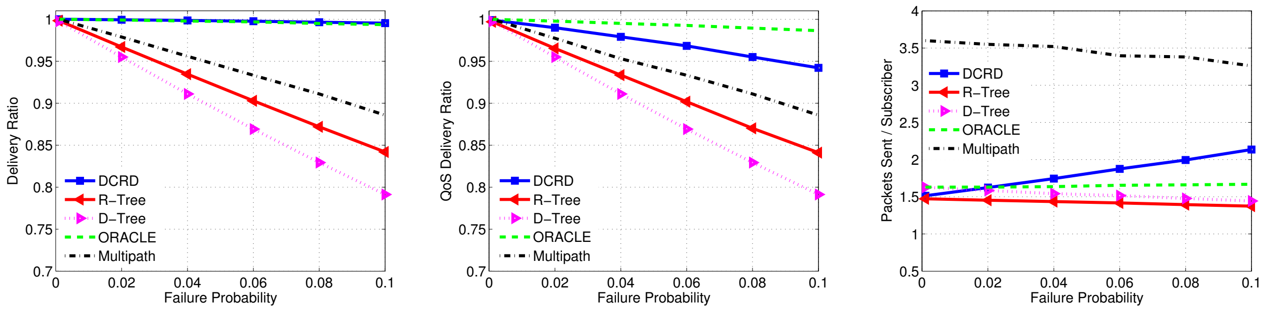
The following performance metrics are used to evaluate our DCRD design:

1) Delivery Ratio: the percentage of packets that are delivered to subscribers. If a single packet has multiple subscribers, 100% delivery ratio means all subscribers received the packet successfully. Note that this ratio includes both the packets that are delivered within and after the delay requirement.
2) QoS Delivery Ratio: the percentage of packets that are delivered to subscribers within the delay requirement. Similar to the delivery ratio, if a single packet has multiple subscribers, 100% QoS delivery ratio means all subscribers received the packet within the delay requirement.
3) Packets Sent/Subscribers: the total number of packets sent by any node divided by the total number of subscribers. For example, when a packet traverses a path with two hops (links), the count for this packet is two. This serves as an indicator of the traffic generated by different approaches.



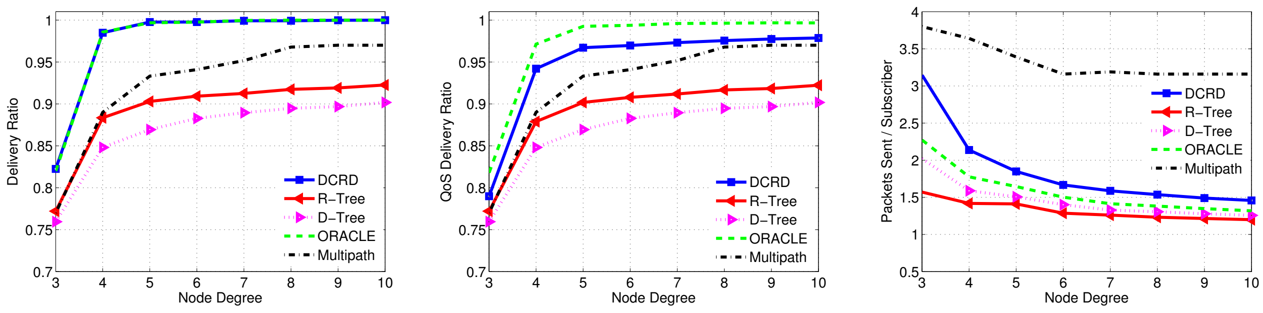
(a) Delivery Ratio (b) QoS Delivery Ratio (c) Packets per Subscriber

Figure 2. Performance Comparison in Fully-Meshed Networks



(a) Delivery Ratio (b) QoS Delivery Ratio (c) Packets per Subscriber

Figure 3. Performance Comparison in Overlay Networks with Degree 5

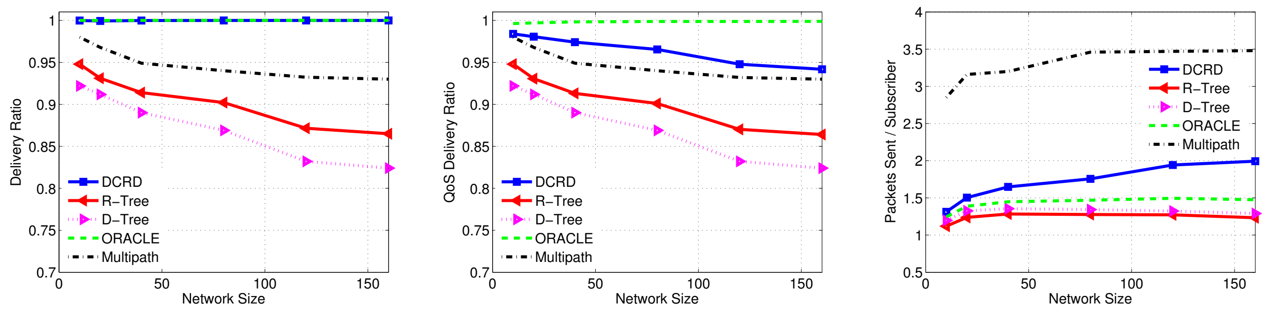


(a) Delivery Ratio (b) QoS Delivery Ratio (c) Packets per Subscriber

Figure 4. Performance Comparison in Overlay Networks with Different Connectivities ($P_f = 0.06$)

D. Performance Evaluation

1) Fully-Meshed Topology: We first consider a full-mesh topology, where every pair of nodes is directly connected by an overlay link. Figure 2(a) shows the delivery ratio of the four designs. For both DCRD and ORACLE, the delivery ratio is always 100% across all failure probabilities. The delivery ratios of D-Tree and R-Tree drop as the failure probability increases. This is because both tree-based approaches do not reroute the packets when a failure occurs. R-Tree has higher delivery ratio than D-Tree because R-Tree is built such that there is minimum number of hops between each publisher-subscriber pair, and thus is more robust to link failures than D-Tree. Multipath delivers more packets than the two trees, at the cost of sending duplicate packets through different paths. However, since it does not reroute packets as well, the delivery ratio drops to 95% as the failure probability increases to 0.1. Figure 2(b) shows the QoS delivery ratio. DCRD delivers about 96.7% of packets within the delay requirement when $P_f = 0.1$ and the percentage is higher for smaller probabilities. This is close to ORACLE, whose percentage is 99.6% for $P_f = 0.1$. The other three approaches suffer from link failure and their QoS ratios are almost the



(a) Delivery Ratio (b) QoS Delivery Ratio (c) Packets per Subscriber

Figure 5. Effect of different network sizes. The node degree is 8 and $P_f = 0.06$. The x-axis shows the number of nodes.

same as the delivery ratios. This is mainly because they do not dynamically explore alternate paths in case of link failures. Figure 2(c) shows the total number of packets sent per subscriber. R-Tree sends only one packet per subscriber since it can always use the direct link in the fully-meshed network. The other four approaches introduce more traffic than R-Tree and Multipath sends the most packets. When $P_f = 0.1$, DCRD introduces about 20% more traffic than D-Tree and ORACLE, but that is still less than 50% of the traffic introduced by Multipath. In this full-mesh topology, DCRD delivers 15% more packets than D-Tree and 11% more packets than R-Tree, when $P_f = 0.1$, while generating 20% extra traffic. It also delivers 5% more packets than Multipath without having to send duplicate packets, a saving of more than 50% of the traffic.

2) Mesh with Reduced Connectivity: We compare the performance of DCRD with other approaches when the overlay network has reduced connectivity. We decreased the link degree of each node to five. Figure 3(a) shows that the delivery ratio of DCRD is similar to the fully-meshed network case, while that of D-Tree, R-Tree and Multipath all dropped by 5%. With less connectivity, the number of hops between the publisher and subscribers becomes larger, and thus there is a higher chance that the path between the publisher and the subscriber gets disconnected due to a link failure.

Figure 3(b) shows the QoS delivery ratio of DCRD dropped by 2% compared with the fully-meshed network due to the reduced degree of connectivity. However, DCRD still delivers within the delay requirements 10% more packets than R-Tree, 15% more than D-Tree and 5% more than Multipath.

Figure 3(c) shows the number of packets sent per subscriber and as expected, all the values are higher than fully-meshed networks. R-Tree and D-Tree send fewer packets than DCRD because they do not explore alternate paths and just stop sending when a packet is lost in the middle of a path due to a link failure, resulting in a low delivery ratio and low QoS delivery ratio. This also explains why Multipath sends fewer packets as the failure probability becomes larger. In the worst case with $P_f = 0.1$, DCRD sends around 40% more packets than D-Tree and R-Tree, while delivering 15% more packets than R-Tree, 20% more packets than D-Tree and 10% more packets than Multipath. It still saves 40% of the traffic compared with Multipath. Compared to ORACLE, DCRD sends 25% more packets in the worst case because it first needs to try a path in order to discover a failure, while ORACLE knows which paths are (and will be) experiencing failures and thus chooses the path that would deliver the packet successfully to the subscriber.

3) Different Connectivity: We change the node degree from 3 to 10 in simulation with the same failure probability 0.06. Figure 4(a) and Figure 4(b) show that as long as the node degree is 5 or bigger, DCRD delivers more than 96% of the packets while meeting the delay requirement. The QoS delivery ratio of DCRD is only 3% lower than ORACLE. The other three solutions suffer from link failures, and the delivery ratios are about 10% less than DCRD for the two tree solutions, and 5% less for Multipath. For the degree of 4, DCRD still delivers about 98% of packets, but QoS delivery ratio is reduced to 94%. For the degree of 3, both the delivery ratio and QoS delivery ratio of all the five approaches become lower than 85% because when the network is sparsely connected, there may not be a path that is not experiencing a failure and is shorter than QoS delay requirement. Figure 4(c) shows the number of packets sent per subscriber. DCRD sent 25% more packets when the node degree is 4 compared to ORACLE.

4) Different Network Sizes: We change the network size to {10,20,40,80,120,160} nodes to see the scalability of DCRD. In the simulation, the failure probability is set to 0.06 and the node degree is 8. Figure 5 shows that the performance degrades for all the five designs as the network becomes larger. This is because with a fixed node degree, the diameter of the network increases, the number of hops between each publisher and subscriber pair also increases. As a result, it is more likely for R-Tree and D-Tree to fail since they have more hops on the routing paths. The QoS delivery ratio of DCRD, however, is still around 5% lower than ORACLE when P_f is 0.1, with 33% more packets sent. As the network size increases, from 10 to 160, DCRD introduces 60% more traffic than the two tree-based solutions, meaning that with a larger network size and

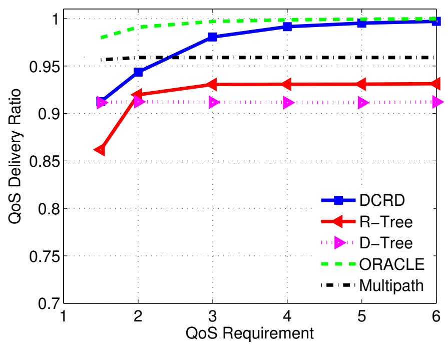


Figure 6. Effect of different level of QoS delay requirements. The node degree is 8 and $P_f = 0.06$.

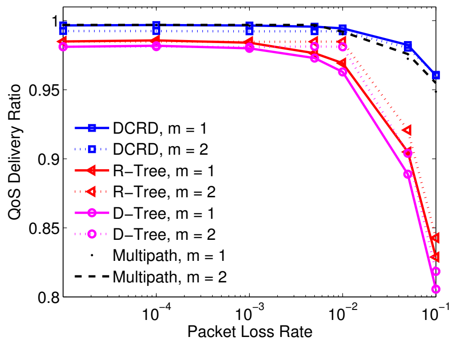


Figure 7. CDF of Packets that have missed deadline. $P_f = 0.06$.

relatively less connectivity, DCRD reroutes packets through longer paths to bypass the failure links. However, the cost is still less than that of Multipath, which sends duplicate packets at the beginning.

5) Different QoS Delay Requirements: Figure 6 shows the QoS delivery ratio in the 20-node networks with different level of QoS delay requirement. The failure probability is set to 0.06 and the node degree of the networks is 8. The delay requirement for each publisher-subscriber pair is specified as the multiple of the shortest path delay. Prior to this experiment, we used the factor of three, and we vary this factor in this experiment. The x-axis shows this multiplication factor.

When the multiplication factor increases from 1.5 to 2, the QoS delivery ratio of DCRD increases by 4%; when the multiplication factor increases from 2 to 3, the QoS delivery ratio of DCRD increases by another 4%. Almost 100% of the packets can be delivered with a delay requirement shorter than 4 times of the shortest path delay. Even for a multiplication factor of 1.5, more than 90% of the packets are delivered by DCRD, meeting the delay requirement. For R-Tree and D-Tree, since they suffer from link failure, the QoS delivery ratio almost remains the same as the multiplication factor increases. Only the QoS delivery ratio of R-Tree increases by 5% when the multiplication factor increases from 1.5 to 2. This is because with a looser delay requirement, there is higher chance that the most reliable path can meet the deadline. For Multipath, however, we see that it delivers within the QoS requirement 5% more packets than DCRD when the requirement is 1.5 times of the shortest path delay. As the requirement becomes looser, DCRD delivers more packets within requirement than Multipath, and the QoS delivery ratio of Multipath almost remains the same due to link failures. We conclude from the simulation that when there are very urgent tasks, Multipath delivers more packets that can meet the deadline at the cost of doubling the internet traffic. If there is relatively loose requirement (e.g., 2∼3 of the shortest delay path), DCRD performs better and introduces much less internet traffic.

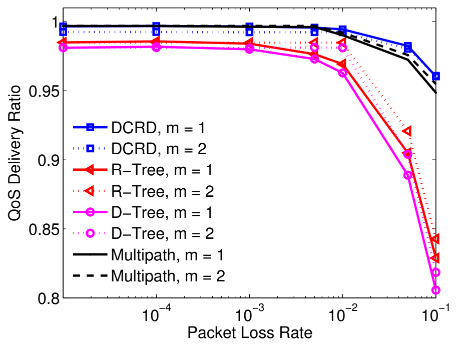


Figure 8. Effect of different packet loss rate P_l and different number of transmissions through each link m. The node degree is 8 and $P_f = 0.01$.

6) Delay Statistics: We show in Figure 7 the cumulative distribution of those packets in DCRD who miss the delay requirement in 20-node networks with fully-meshed topology and with degree of 8. The x-axis is the actual delay divided by the delay requirement. (This value starts from 1 because only those messages that have violated the delay requirement are included in the statistics.) It can be seen from the figure that for both topologies, around 50% of the packets that missed the deadline arrived within 25% of delay requirement from the deadline. Also, around 78% of the packets that missed the deadline took less than 50% longer than the delay requirement in fully-meshed networks, and this number drops to 70% in networks with degree 8. This is because with reduced connectivity, it is less likely to find an alternate path with short delay. However, we still find that 80% of the packets are less than 75% longer than the delay requirement. These results indicate that with DCRD, even those packets who missed the deadline still arrived within a short delay.

7) Different Packet Loss Rate and m: We vary the packet loss rate P_l from 10^{-1} to 10^{-5} in 20-node networks with node degree of 8 and $P_f = 0.1$. The QoS delivery ratio of DCRD, R-Tree, D-Tree and Multipath for different m values are shown in Figure 8. When $P_l \leq 10^{-3}$, DCRD has about 0.5% more packets delivered within the delay requirement when $m = 1$ compared with $m = 2$, i.e., switching to the next node is more delay-efficient compared with retransmission on the same link. This is because P_l is much smaller than P_f, and a missing ACK is more likely caused by link failure other than packet loss. Thus, it is faster to try the next path immediately rather than first try retransmissions (which is futile in this case) and then look for an alternate path. For R-Tree, D-Tree and Multipath, the QoS delivery ratio for different m values when $P_l \leq 10^{-3}$ does not change since P_l is too small to make the retransmissions affect the result. When P_l becomes equal or larger than P_f, the QoS delivery ratio for different m values of DCRD become close, indicating that the packet loss can be recovered by both rerouting and retransmission. For R-Tree, D-Tree and Multipath, a 1%$\sim$2% increasing on QoS delivery ratio can be easily seen from the figure when m increases from 1 to 2. This is because as P_l increases, there is a higher chance that a packet is lost in a good link, and such a loss can be recovered from retransmissions.

V. Conclusion

This paper proposes DCRD, a dynamic routing scheme for overlay pub/sub systems that provides delay-aware, reliable message delivery. Different from traditional approaches, which utilizes a fixed routing topology to forward packets, DCRD dynamically switches between different next-hop downstream nodes to quickly bypass link failures and to increase the chance of delivering the packets within the delay requirement. Each node distributively calculates the expected delay and delivery ratio regarding each subscriber and then builds a sending list consisting of all possible routes, sorted in a way that minimizes the expected delay. Routing follows this sending list and delivery is guaranteed as long as there exists at least one path from publisher to subscriber. DCRD is evaluated in simulations with various network settings and is shown to outperform the existing tree-based approaches and multipath approach; DCRD provides reliably delivery and delivers 10%, 15%, and 5% more messages on-time than R-Tree, D-Tree and Multipath approach, respectively. DCRD is also shown to closely approach the oracle-based tree performance by providing more than 98% QoS delivery ratio for link failure probabilities below 4% and more than 95% QoS delivery ratio for almost all other settings.

This study focused on the evaluation of the dynamic routing algorithm under different loss and link failure conditions. Work is also underway to evaluate DCRD performance in the presence of node failures. With node failures there is the potential for simultaneous link failures and long outages that render one or more destinations unreachable for a given topology. Thus, DCRD might need to incorporate other strategies, such as dynamic topology construction, to maintain good latency performance, while still providing reliable delivery. The simulations presented in this study provide good insight into how a topology might be constructed for a real deployment. In particular, the results for an overlay with node degree of 5 or greater are not appreciably different from the full mesh results, suggesting that sparsely connected topologies are viable for real deployments. Parallel efforts to implement DCRD in a candidate messaging middleware system [20], [21] and evaluate DCRD's performance in a real deployment are under development as well.

References

[1] B. Blakeley, H. Harris, and R. Lewis, *Messaging and queueing using the MQI*. New York, NY: McGraw-Hill, Inc., 1995. [Online]. Available: http://portal.acm.org/citation.cfm?id=SERIES9694.206786

[2] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, June 2003.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. 18th ACM SOSP*, 2001, pp. 131–145.

[4] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-Path vs. Multi-Path Overlay Routing," in *Proc. of ACM IMC*, 2003, pp. 91–100.

[5] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (pdf) system for packet switched networks," in *IN PROCEEDINGS OF IEEE INFOCOM*, 2003, pp. 663–672.

[6] L. Lao, J.-H. Cui, M. Gerla, and D. Maggiorini, "A Comparative Study of Multicast Protocols: Top, Bottom, or In the Middle?" in *INFOCOM'05*, 2005.

[7] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O. Jr., "Overcast: Reliable Multicasting with an Overlay Network," in *OSDI'00*, 2000.

[8] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," in *INFOCOM'03*, 2003.

[9] G. in Kwon and J. W. Byers, "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections," in *INFOCOM'04*, 2004.

[10] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg, "Content-Based Publish-Subscribe over Structured Overlay Networks," in *ICDCS'05*, 2005.

[11] R. E. Strom, G. Banavar, T. D. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. C. Sturman, and M. Ward, "Gryphon: An Information Flow Based Approach to Message Brokering," 1998.

[12] M. Onus and A. W. Richa, "Minimum Maximum Degree Publish-Subscribe Overlay Network Design," in *INFOCOM'09*, 2009.

[13] M. Junginger and Y. Lee, "A Self-Organizing Publish/Subscribe Middleware for Dynamic Peer-to-Peer Networks," *IEEE Network*, vol. 18, no. 1, pp. 38–43, 2004.

[14] J. Wang, W. Yurcik, Y. Yang, and J. Hester, "Multi-Ring Techniques for Scalable Battlespace Group Communications," *IEEE Communications Magazine*, vol. 43, no. 11, 2005.

[15] G. Tan, S. Jarvis, and D. Spooner, "Improving Fault Resilience of Overlay Multicast for Media Streaming," in *DSN'06*, 2006.

[16] "AT&T Network," http://ipnetwork.bgtmo.ip.att.net.

[17] A. Cruise, "ADS-B for next gen ATC," *Airside International*, September 2008.

[18] Z. Li and P. Mohapatra, "QRON: QoS-aware Routing in Overlay Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 29–40, 2004.

[19] A. Riabov, Z. Liu, and L. Zhang, "Overlay Multicast Trees of Minimal Delay," in *ICDCS'04*, 2004.

[20] M. Kim, K. Karenos, F. Ye, J. Reason, H. Lei, and K. Shagin, "Efficacy of techniques for responsiveness in a wide-area publish/subscribe system," in *ACM/IFIP/USENIX 11th International Middleware Conference (Middleware)*, Bangalore, India, Nov. 2010.

[21] H. Yang, M. Kim, K. Karenos, F. Ye, and H. Lei, "Message-Oriented Middleware with QoS Awareness," in *ICSOC-ServiceWave'09*, 2009.