Reliability and Timeliness Analysis of Fault-tolerant Distributed Publish / Subscribe Systems

Thadpong Pongthawornkamol*
University of Illinois
Urbana, IL
tpongth2@illinoisalumni.org

Klara Nahrstedt
University of Illinois
Urbana, IL
klara@illinois.edu

Guijun Wang
Boeing Research & Technology
Seattle, WA
guijun.wang@boeing.com

Abstract

Distributed publish / subscribe paradigm is a powerful data dissemination paradigm that offers both scalability and flexibility for time-sensitive applications. However, its nature of high expressiveness makes it difficult to analyze or predict the performance of publish / subscribe systems such as event delivery probability and end-to-end delivery delay, especially when the publish / subscribe systems are deployed over distributed, large-scale networks. While several fault tolerance techniques to increase reliability in distributed publish / subscribe systems have been proposed, event delivery probability and timeliness of publish / subscribe systems with such reliability enhancement techniques have not yet been analyzed. This paper proposes a generic model that abstracts the basic distributed publish / subscribe protocol, along with several commonly used fault-tolerant techniques, on top of distributed, large-scale networks. The overall goal of this model is to predict quality of service (QoS) in terms of event delivery probability and timeliness based on statistical attributes of each component in the distributed publish / subscribe systems. The evaluation results via extensive simulations with parameters computed from real-world traces verifies the correctness of the proposed prediction model. The proposed prediction model can be used as a building block for automatic QoS control in distributed, time-sensitive publish / subscribe systems such as subscriber admission control, broker deployment, and reliability optimization.

1 Introduction

Recently, distributed publish / subscribe systems have emerged as an effective multi-source, multi-sink communication paradigm for large-scale, time-sensitive applications such as stock report, live sportcasting, and social network messaging. The main concept of publish / subscribe paradigm is that senders and receivers of information are connected loosely based on the content of the information. Specifically, in a publish / subscribe system, a *publisher* (i.e., sender) can produce its events (i.e. *messages*) without specifying the set of *subscribers* (i.e., receivers). Instead, each subscriber specifies the content (or topic) of event it is interested to receive. All events produced from publishers containing content (or topic) that match a subscriber's interest are then delivered to that subscriber via a network of intermediary servers called *brokers*. Since the information flows based on the content of the information, publishers and subscribers are decoupled in space, time, and synchronization [16]. Such transparency allows the system to scale and adapt well under dynamic environments, resulting in wide adoption of publish / subscribe paradigm in many contexts such as cloud computing [26], mobile computing [9], and peer-to-peer services [36].

While distributed publish / subscribe systems achieve scalability and fault tolerance, failures at brokers or links between brokers can still cause time-sensitive events to be lost or expired before being delivered to the subscribers [19, 28]. Several reports have shown that while high-end commercial servers with high maintenance generally achieve at least 99.9% availability (i.e., available 99.9% of the time) [4], most standard, off-the-shelf commodity servers with low to moderate maintenance may have less than 90% availability [3]. To cope with such component failures in the context of publish / subscribe systems, several fault-tolerant / fault-recovery techniques have been proposed to increase service availability in distributed publish / subscribe systems [13, 14]. However, to the best of our knowledge, the effect of such commonly-used fault-tolerant techniques to a publish / subscribe system's reliability and timeliness has not been analyzed yet.

In this paper, we propose a quantitative, analytical model to predict the effect of failures and commonly used recovery techniques to the quality of service (QoS) each subscriber receives in distributed, time-sensitive publish / subscribe systems. The primary goal of such analytical model is to estimate each subscriber's *real-time re-*

*The author is currently employed at Google Inc.

liability, which is the percentage of events that are successfully delivered to each subscriber on time (i.e., before the event is expired). The analytical model covers common component failures and recovery mechanisms, resulting in the model's high applicability. The evaluation results via simulations with parameters computed from real-world traces yield correctness of the proposed analytical model. The proposed model can be used as a building block for automatic QoS control in distributed, time-sensitive publish / subscribe systems such as subscriber admission control, broker deployment, and reliability optimization.

The organization of this paper is as follows. In Section 2, we first describe the basic publish / subscribe model, failure model, and commonly used fault tolerance / recovery techniques. In Section 3, we then formulate the subscriber real-time reliability estimation problem and propose a generic, protocol-independent analytical framework to solve such problem. In Section 4, we propose a set of protocol-dependent, publisher-subscriber pairwise reliability estimation models for each fault tolerance / recovery mechanism from Section 2. Section 5 then presents the evaluation results to validate the proposed analytical model via simulations. Related work is summarized in Section 6. Conclusions and future work are then discussed in Section 7.

2 Model and Assumptions

In this section, we describe the details of the distributed publish / subscribe architecture model, quality of service (QoS) model, failure model, and commonly used fault-tolerant techniques used in this paper. In the next section, we will then present the mathematical framework that realizes the model described in this section in order to predict QoS level each subscriber receives.

2.1 Distributed Publish / Subscribe Model

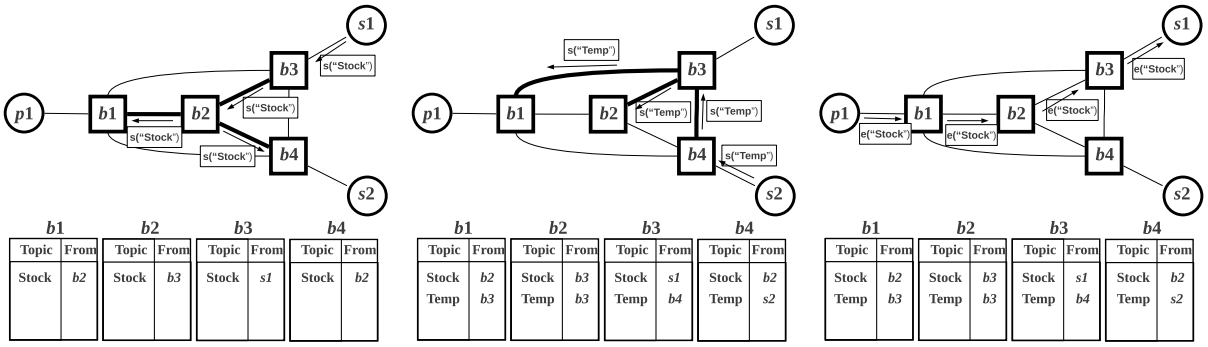
We model the publish / subscribe network as a network of brokers. Brokers can be placed inside the same domain (e.g, brokers within cloud), across different private domains (e.g., federated clouds), or across different public domains (e.g., peer-to-peer systems). Each subscriber / publisher is connected to only one of the brokers. The broker that is connected to a subscriber / publisher is called the *home broker* with respect to that subscriber / publisher. Figure 1 shows an example of a publish / subscribe network with 4 brokers , where the home brokers of subscriber s_1, subscriber s_2, and publisher p_1 are broker b_3, b_4, and b_1 respectively.

When a subscriber joins the system, it chooses a broker in the system as its home broker and sends its *subscription* message to the home broker. The subscription message specifies a *topic*[1] value, which describes the category of event that the subscriber wants to receive. Upon receiving the subscription from the subscriber, the home broker stores the subscription and the subscriber into its routing table before propagating the subscription message to other brokers. Each published event has *topic* and *deadline* associated with it. When a publisher publishes a new event, the publisher sends the published event to its home broker, who then routes the event via the broker network to all subscribers whose subscriptions have the same topic as the published event. If the event arrives at a subscriber before the event's deadline, we say that the event is delivered to that subscriber on time. Otherwise, we consider that event to be *expired* with respect to that subscriber.

For genericity, this paper does not make any assumption about subscription propagation / event routing processes within broker network. The only assumptions are that the broker network must be *stable* (i.e., neighborhood relationships between brokers do not change frequently over time) and the event routing path must be *consistent* (i.e., for each publisher-subscriber pair, brokers will always use the same path to route all events from the publisher to the subscriber). For demonstration, this paper focuses on *tree-based forwarding* (e.g., Figure 1), which is a publish / subscribe routing scheme that satisfies path consistency assumption. In tree-based forwarding scheme, a broker tree overlay is arbitrarily but consistently formed for each topic. When a broker receives a new subscription, the broker stores the subscription and its source to the broker's routing table before forwarding the subscription to its neighbors in the subscription's topic broker tree. Figure 1(a) shows an example when subscriber s_1 subscribes to topic "Stock", which forms the broker tree rooted at broker b_2. Note that subscriptions with different topics can have different corresponding broker trees. For example in Figure 1(b), subscriber s_2 subscribes to topic "Temp", which forms the broker tree rooted at broker b_3. Upon receiving a published event from a publisher, the publisher's home broker checks the event with each subscription stored in its routing table. For each subscription whose topic matches the event, the broker forwards the event to the link which it receives that subscription from. Note that an event is forwarded *once* per link even though there are multiple matching subscriptions from that link. The event forwarding process then continues, and the event is propagated hop-by-hop along the topic tree in the reverse direction of the subscription until it reaches the designated subscribers. Figure 1(c) shows an example of publisher p_1 publishing an event with topic "Stock".

While Figure 1 describes topic-based, tree-based pub-

[1] While this paper focuses on topic-based publish / subscribe, our approach can be extended to support content-based publish / subscribe as well [29].



(a) s_1 subscribes with topic 'Stock' (b) s_2 subscribes with topic 'Temp' (c) p_1 publishes an event with topic 'Stock'

Figure 1: Example of subscription propagation and event routing in a tree-based publish / subscribe network

lish / subscribe model, our analytical model can be applied to content-based publish / subscribe systems with any routing scheme that satisfies the path consistency assumption [29].

2.2 QoS Model

In our previous work, we proposed the subscriber-level QoS metric called *subscriber real-time reliability* [30]. Subscriber real-time reliability can be defined as follows: A subscriber s is said to receive the service with real-time reliability $r_s \in [0,1]$, where r_s is defined as the fraction of all events of s's interest that arrives at s before its deadline (i.e., delivery delay less than the message lifetime). This metric both standard reliability and timeliness properties, making it a suitable indicator of QoS in time-sensitive publish / subscribe applications.

2.3 Failure Model

In this paper, we discuss two types of failures that could affect subscriber real-time reliability : *link failures* and *broker failures*. We assume crash-recovery failure model for both broker failures and link failures, which means each broker / link is assumed to be either *on* or *off* at any point of time. When a broker fails (i.e., is in *off* state), it stops its activity until it recovers (i.e., is repaired back to *on* state). When a failed broker recovers, it loses all of its soft-state information (e.g., subscription routing table and queued events) that it had before the failure. However, we assume that the failed broker does not lose the broker graph information (i.e., the list of all brokers and their neighborhood relationship), which is stored in each broker's persistent, non-volatile storage. When a link fails, any event that is sent to the link will be lost.

In this paper, we assume that link failures and broker failures are *independent* and *exponentially distributed* for analysis feasibility. Previous studies also have shown that the assumption of exponential time between failures is true in many distributed systems [3, 35].

2.4 Fault-Tolerant Mechanisms

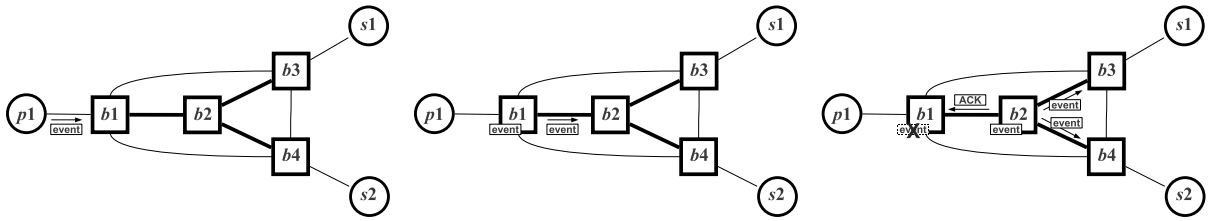
In order to cope with broker and link failures, several fault tolerance / recovery schemes for publish / subscribe systems have been proposed [7, 8, 13, 19, 22]. This section summarizes and discusses such techniques. Note that some of these techniques are generic and not limited to publish / subscribe systems. However, this paper focuses on the analysis of such techniques in the publish / subscribe context.

2.4.1 Periodic Subscription

In periodic subscription scheme [33], each subscriber periodically re-issues its subscription message to its home broker, which then propagates the subscription to other brokers in the network. Each broker also maintains a timestamp for each subscription entry in its routing table. The timestamp is refreshed every time the broker receives the corresponding subscription. The broker discards any subscription from its routing table if the subscription is not refreshed within a period of time (i.e., timeout). The periodic subscription scheme can help prevent subscription loss, but it cannot prevent event loss. More details about periodic subscription can be found by several previous works [19, 20].

2.4.2 Event Buffering / Retransmission

In event buffering scheme [8, 13], each broker ensures event delivery to its next hop neighbor as follows. When a broker receives an event from one of its immediate neighbors, it performs the event matching and calculates the event's forwarding set (i.e., the set of immediate neighbors to forward the event to). The broker then stores the event and its forwarding set into the broker's non-volatile storage and sends the acknowledgment message (ACK) containing the event sequence number back to its upstream neighbor. The broker then forwards the event to the event's forwarding set. The broker then waits



(a) Publisher publishes an event (b) Home broker stores and forwards event (c) Broker receives ACK and discards event

Figure 2: Example of event buffering / retransmission scheme

for the ACK message from each next-hop neighbor in the event's forwarding set. The broker discards the event from its non-volatile storage once it collects all the ACK messages from all brokers in the forward set, as now it is certain that the event has been received by all of the next-hop brokers. If, due to failures, the broker does not receive ACK messages from some next-hop neighbors, it retransmits the event to each of such neighbors until all ACK messages are collected or the buffered event becomes expired. Figure 2 illustrates an example of event buffering / retransmission scheme.

The event buffering / retransmission guarantees that the events will not be lost due to broker / link failures. However, if the routing path is disconnected for too long, the event may be expired before it is delivered to the subscribers.

2.4.3 Redundant Path Bypassing

Redundant path bypassing scheme relies on the fact that even when the routing path between a publisher to a subscriber is disconnected due to broker / link failures, it might be possible to find another publisher-subscriber path that excludes the failed brokers / links [8, 19, 22, 23].

The detail of the redundant path bypassing in the context of tree-based publish / subscribe is as follows[2]. Whenever a broker detects a change of its neighbor's state (e.g., neighbor fails, neighbor recovers), it uses a link state protocol to broadcast the update message to all other reachable brokers. Each broker can update the global view of the entire broker network. With the up-to-date global view of the network, each broker can identify the set of *immediately reachable children* of a failed broker along the tree. The immediately reachable children of a broker b is the set of b's next-hop brokers that are available and reachable. For example in Figure 3, the immediately reachable children set of failed broker $b2$ are $b3$ and $b4$. Hence, each event that is supposed to be sent to the failed broker will be forwarded to the failed broker's immediately reachable children instead (i.e., detour routing). The same approach applies to subscription forwarding as well.

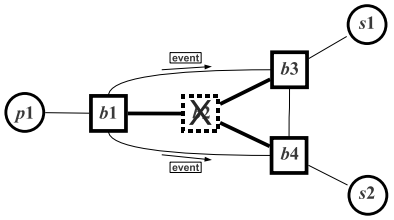


Figure 3: Example of path bypassing scheme where broker $b1$ bypasses failed broker $b2$

The goal of this paper is to estimate each subscriber's real-time reliability when each of such fault-tolerant techniques is employed in the publish / subscribe system. Section 3 will present the generic estimation framework while Section 4 will present the estimation algorithm for each specific fault-tolerant technique.

3 Subscriber Real-time Reliability Estimation Framework

In this section, we propose a mathematical model of the publish / subscribe system and use the proposed model to formulate the subscriber real-time reliability estimation problem. We then present the generic estimation algorithm to solve the problem.

3.1 Problem Formulation

We present the mathematical model of each component in the publish / subscribe system as follows.

3.1.1 Real-time Event Model

Let $\mathbb{E}$ be the set of all events ever published in the system. Each event $e \in \mathbb{E}$ contains topic τ_e and lifetime d_e, which is the duration since the time the event was published until the time the event is expired. Without loss of

[2]The path bypassing technique can be used with other routing schemes as well.

generality, we assume that all events in the system have the same lifetime D (i.e., $\forall e \in \mathbb{E} : d_e = D$), which is a known constant. Our scheme can be modified to work when events have different lifetime values as well.

3.1.2 Publisher / Subscriber Model

Let $\mathbb{P}$ and $\mathbb{S}$ be the set of all publishers / subscribers in the system. Each publisher $p \in \mathbb{P}$ publishes events with topic τ_p with rate λ_p. Each subscriber $s \in \mathbb{S}$ is interested in events with topic τ_s. A subscriber s is said to be a recipient of a publisher p if they share the same topic of interest (i.e., $\tau_s = \tau_p$). We assume that λ_p, τ_p, and τ_s are known for all publishers and subscribers.

3.1.3 Broker / Link Model

Let $\mathbb{B}$ be the set of all brokers in the system. We model each broker $b \in \mathbb{B}$ as a tuple (γ_b, σ_b), where γ_b and σ_b are exponentially distributed failure rate and repair rate, respectively. That is, the broker b has exponentially distributed time between failures and time to repair with mean $\frac{1}{\gamma_b}$ and $\frac{1}{\sigma_b}$, respectively.

Hence, broker b's availability, denoted by a_b, is the fraction of time the broker b is on, which can be computed as $a_b = \frac{\frac{1}{\gamma_b}}{\frac{1}{\gamma_b}+\frac{1}{\sigma_b}}$.

Likewise, let $\mathbb{L}$ be the set of all links in the system. Each link $l \in \mathbb{L}$ has exponentially distributed time between failure and time to repair with rate γ_l and σ_l respectively[3]. Link l's availability value a_l is also calculated as $a_l = \frac{\frac{1}{\gamma_l}}{\frac{1}{\gamma_l}+\frac{1}{\sigma_l}}$

We assume that each broker's failure / repair rates (γ_b, σ_b) and each link's failure / repair rates (γ_l, σ_l) are known via statistical history data collection [1, 17, 21].

With the described mathematical model, we formulate the subscriber real-time reliability estimation as follows.

Subscriber Real-time Reliability Estimation Problem : Given a publish / subscribe overlay network $\mathbb{G} = (\mathbb{N}, \mathbb{L})$ where $\mathbb{N} = \mathbb{B} \cup \mathbb{P} \cup \mathbb{S}$, estimate the value of subscriber real-time reliability r_s for each subscriber $s \in \mathbb{S}$.

We use the term r'_s to denote the estimated value of subscriber real-time reliability r_s. The goal of our analytical model is to calculate r'_s that estimates r_s as accurately as possible (i.e., $\min|r'_s - r_s|$).

[3] Without loss of generality, we assume that the local link connected between publish / subscriber to its home broker does not fail. Our scheme can be simply modified for non-reliable local link scenarios.

3.2 Generic Estimation Framework

This section describes a generic framework to estimate subscriber real-time reliability. The subscriber real-time reliability estimation problem can be generally broken down into two sub-problems, which are estimating publisher-subscriber pairwise flow rate and estimating publisher-subscriber pairwise flow reliability.

3.2.1 Pairwise Flow Rate

The publisher-subscriber pairwise flow rate is the average event traffic flow rate from a publisher to a subscriber when no failure occurs. The publisher-subscriber pairwise flow rate λ_{ps} between a publisher $p \in \mathbb{P}$ and a subscriber $s \in \mathbb{S}$ can be calculated as follows.

$$\lambda_{ps} = \begin{cases} \lambda_p & \text{if } \tau_p = \tau_s \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

That is, the pairwise traffic flow rate from publisher p to subscriber s is equal to publisher p's publishing rate if they share the same topic, and equal to zero otherwise. Since λ_p, τ_p, and τ_s are known for each publisher $p \in \mathbb{P}$ and subscriber $s \in \mathbb{S}$ in the system, we can calculate pairwise traffic flow rate λ_{ps} for each publisher-subscriber pair in the system.

3.2.2 Pairwise Reliability

The publisher-subscriber pairwise reliability is the probability that a publisher's event of a subscriber's interest will be delivered to that subscriber before it is expired. We use the notation $r'_{ps} \in [0,1]$ to denote the pairwise reliability between publisher $p \in \mathbb{P}$ and subscriber $s \in \mathbb{S}$. As mentioned, the pairwise reliability depends on the fault-tolerant technique used in the publish / subscribe system. Section 4 presents the calculation of pairwise reliability for each technique discussed in Section 2.4.

3.2.3 Generic Estimation Algorithm

Subscriber s's real-time reliability r_s is the probability that s will receive an event of its interest successfully before the deadline D. Hence, the estimated value of r_s, denoted by r_s can be calculated as the weighted average of publisher-subscriber pairwise reliability between each publisher to that subscriber, with the weight equal to the pairwise event flow rate from the corresponding publisher to that subscriber. That is,

$$\begin{aligned} r'_s &= \frac{\text{E[rate of events delivered on time to } s\text{]}}{\text{E[total rate of events of } s\text{'s interest]}} \\ &= \frac{\sum_{p \in \mathbb{P}} (r'_{ps} \cdot \lambda_{ps})}{\sum_{p \in \mathbb{P}} \lambda_{ps}} \end{aligned} \quad (2)$$

Equation (2) is a generic equation that can be used with any fault-tolerant mechanism discussed in Section 2.4.

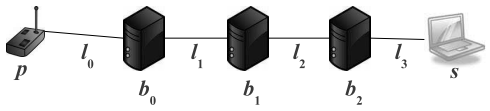


Figure 4: Example of a publisher-subscriber path with length 3

However, different mechanisms require different equations to estimate pairwise reliability r'_{ps}, which will be demonstrated in the next Section.

4 Pairwise Reliability Estimation

This section proposes an analytical model to calculate the estimated publisher-subscriber pairwise reliability r'_{ps} for each different fault tolerance/recovery protocols presented in Section 2.4.

4.1 Static Path

Without any reliable mechanism, the subscription information stored at each broker about the subscriber will be eventually lost when that broker fails. If a subscriber does not have reliable subscription or periodic subscription mechanisms, its subscription along the routing path will be eventually lost, preventing any subsequently published event to be delivered to the subscriber. Hence, the steady-state pairwise reliability each publisher-subscriber pair (p,s) will be zero (i.e., $r'_{ps} = 0$).

4.2 Static Path + Periodic Subscription

With the use of periodic subscription (Section 2.4.1), each failed broker can recover its routing information once it recovers from its failure. However, event loss is still possible as there is no reliable acknowledgement.

We analyze the pairwise reliability of each publisher-subscriber pair with static path as follows. Let δ_{ps} denote the event routing path between a publisher p and a subscriber s. Since we assume our publish / subscribe routing scheme to be consistent, δ_{ps} is static and known for each publisher-subscriber pair (p,s). We define *path length*, denoted by $|\delta_{ps}|$, as the number of brokers in the path. Hence, a path δ_{ps} can be expressed as the sequence $(p, l_0, b_0, l_1, b_1, ..., l_{|\delta_{ps}|-1}, b_{|\delta_{ps}|-1}, l_{|\delta_{ps}|}, s)$. Figure 4 shows an example of a path of length 3.

Since the static path scheme always uses only one path δ_{ps} to forward events between publisher-subscriber pair (p,s), the pairwise real-time reliablity r'_{ps} is equal to the fraction of time that the path δ_{ps} is connected. That is,

$$\begin{aligned} r'_{ps} &= \text{P[path } \delta_{ps} \text{ is connected]} \\ &= a_{b_0} \Pi_{i=1}^{|\delta_{ps}|-1} (a_{l_i} \cdot a_{b_i}) \end{aligned} \quad (3)$$

where a_x is the availability of component (broker or link) x.

4.3 Static Path + Event Buffering

With the reliable acknowledgment protocol (i.e., Section 2.4.2) in static path, an event of a subscriber s's interest that is published by a publisher p will be eventually delivered to s, given that p's home broker is available when p publishes the event (since we assume that p does not have retransmission capability). This is because the event will always be buffered at some broker along the path between p and s, even when the path is disconnected[4]. The event will then be forwarded when the next-hop broker and link are available, and eventually delivered to the subscriber. However, the delay the event spends in the buffer may be longer than its lifetime, which causes the event to be expired.

To analyze the pairwise reliability r'_{ps} between a publisher p and a subscriber s under static path with event buffering scheme, consider the single, unique path δ_{ps} connecting p and s. Assuming the event arrival time to be independent from the path δ_{ps}'s state, we estimate the path real-time reliability as follows.

$$\begin{aligned} r'_{ps} &= \text{P[an event from } p \text{ arrives at } s \text{ on time]} \\ &= \text{P[}p\text{'s home broker is on].} \\ &\quad \text{P[end-to-end delay less than event lifetime]} \\ &= a_{b_0} \cdot \text{P}[d_{ps} < D] \end{aligned} \quad (4)$$

where d_{ps} is the end-to-end delivery delay and D is the event lifetime. Thus, it is necessary to calculate the end-to-end delivery delay distribution d_{ps} first in order to estimate path reliability r'_{ps}.

To calculate delay distribution d_{ps} for path δ_{ps}, we need to calculate per-hop buffering delay at each broker $b_i (0 \le i < |\delta_{ps}|)$ in the path (we assume that link transmission delay and broker processing delay are negligible compared to buffering delay). Consider when the event is received successfully at broker b_i and hence broker b_i will try to forward the event to broker b_{i+1}. If both link l_{i+1} and broker b_{i+1} are up at the moment, the event will be transmitted successfully to broker b_{i+1} immediately, thus incurring zero buffering delay at broker b_i. However, if either link l_{i+1} or broker b_{i+1} is down at the moment, the event will be buffered at the broker b_i, which will keep retransmitting the event until the event gets through to broker b_{i+1}. The broker b_i discards the event if the event expires. Note that the event will get through only when all b_i, l_{i+1}, and b_{i+1} are up at the same time.

Let d_{b_i} be the buffering delay at each broker b_i $(0 \le i < |\delta_{ps}|)$. We first calculate the probability that $d_{b_i} = 0$ (i.e., the probability that the event is successfully delivered to b_{i+1} immediately), which can be calculated as

[4] In the analysis, we assume each broker to have unbounded buffer such that it can always store any incoming event.

$$\begin{aligned} \mathrm{P}[d_{b_i}=0] &= \mathrm{P}[b_i, l_{i+1}, b_{i+1} \text{ are on} | b_i \text{ is on}] \\ &= \mathrm{P}[l_{i+1}, b_{i+1} \text{ are on}] \\ &= a_{l_{i+1}} \cdot a_{b_{i+1}} \end{aligned} \quad (5)$$

Given that delay is always non-negative, we have

$$\begin{aligned} \mathrm{P}[d_{b_i} > 0] &= 1 - \mathrm{P}[d_{b_i}=0] \\ &= 1 - a_{l_{i+1}} \cdot a_{b_{i+1}} \end{aligned} \quad (6)$$

Now, in the case that the buffering delay at each broker b_i is not zero (with probability $1 - a_{l_{i+1}}.a_{b_{i+1}}$), we need to find the delay distribution in such case. Let $d^+_{b_i}$ be the conditional buffering delay at broker b_i under the condition that $d_{b_i} > 0$. Assuming the event arrives at arbitrary time at broker b_i, the conditional buffering delay $d^+_{b_i}$ is equal to the time it takes for the next-hop path to be repaired (i.e., time until l_{i+1} and b_{i+1} are both in *on* state). Assuming each component's time between failure and time to repair to be exponentially distributed, we can calculate such delay distribution by using continuous-time Markov process diagram that represents the state of broker b_i, link l_{i+1}, and b_{i+1}. The diagram is shown in Figure 5. Each of 8 states depicts each possible state of sub-path (b_i,l_{i+1},b_{i+1}), with each bit representing each individual component's state ($0 = off$, $1 = on$). The first bit (least significant bit) represents b_i's state. The second bit represents l_{i+1}'s state. The third bit (most significant bit) represent b_{i+1}'s state. For example, state "011" represents the state where broker b_i is on, link l_{i+1} is on, and broker b_{i+1} is off. Note that in the scenario where an event arrives at broker b_i and needs to be buffered at b_i, an event will find the system state in either state "001", "011", or "101" with probability $\frac{(1-a_{l_{i+1}})(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}}$, $\frac{a_{l_{i+1}}(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}}$, and $\frac{(1-a_{l_{i+1}})a_{b_{i+1}}}{1-a_{l_{i+1}}.a_{b_{i+1}}}$ respectively. The event will continue to be buffered at broker b_i (note that b_i can also fail but the event is kept in its non-volatile storage) until the state becomes "111", which the event will be transmitted to broker b_{i+1} successfully. Hence, the diagram depicts the absorbing Markov process with three start states = "001", "011", "101" and one absorbing state "111"with the corresponding transition rate matrix $\dot{Q}$ as

$$\dot{Q} = \begin{pmatrix} -\dot{q}_0 & \sigma_{b_i} & \sigma_{l_{i+1}} & 0 & \sigma_{b_{i+1}} & 0 & 0 & 0 \\ \gamma_{b_i} & -\dot{q}_1 & 0 & \sigma_{l_{i+1}} & 0 & \sigma_{b_{i+1}} & 0 & 0 \\ \gamma_{l_{i+1}} & 0 & -\dot{q}_2 & \sigma_{b_i} & 0 & 0 & \sigma_{b_{i+1}} & 0 \\ 0 & \gamma_{l_{i+1}} & \gamma_{b_i} & -\dot{q}_3 & 0 & 0 & 0 & \sigma_{b_{i+1}} \\ \gamma_{b_{i+1}} & 0 & 0 & 0 & -\dot{q}_4 & \sigma_{b_i} & \sigma_{l_{i+1}} & 0 \\ 0 & \gamma_{b_{i+1}} & 0 & 0 & \gamma_{b_i} & -\dot{q}_5 & 0 & \sigma_{l_{i+1}} \\ 0 & 0 & \gamma_{b_{i+1}} & 0 & \gamma_{l_{i+1}} & 0 & -\dot{q}_6 & \sigma_{b_i} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (7)$$

where γ_x and σ_x are component x's exponential failure rate and exponential repair rate described in Section 3.1, and $\dot{q}_i$ is state i's total outgoing rate. For example, $\dot{q}_0 = (\sigma_{b_i} + \sigma_{l_i} + \sigma_{b_{i+1}})$. Thus, the conditional buffering delay at broker b_i is equal to the time to absorption of the absorbing matrix $\dot{Q}$, which is a phase-type distribution [25] and can be calculated by breaking down the matrix $\dot{Q}$ in to the form of

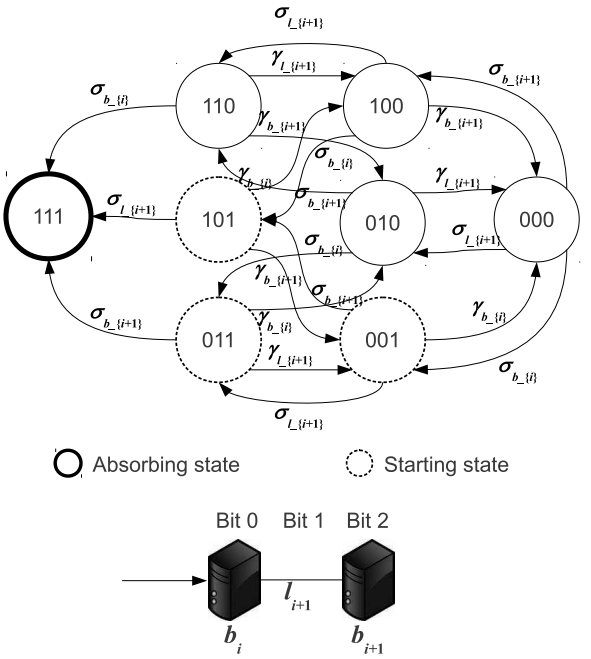


Figure 5: 8-state continuous, absorbing Markov process diagram for per-hop buffering delay analysis

$$\dot{Q} = \left(\begin{array}{c|c} \dot{S} & \dot{\mathbf{S}}^0 \\ \hline \mathbf{0} & 0 \end{array} \right) \quad (8)$$

Where $\dot{S}$ and $\dot{\mathbf{S}}^0$ are the 7x7 top-left sub-matrix and the 7x1 top-right sub-vector of $\dot{Q}$ defined in Equation (7) respectively. Hence, the cumulative distribution of $d^+_{b_i}$ can be calculated as

$$\mathrm{P}[d^+_{b_i} < t] = 1 - \alpha \cdot \exp(\dot{S}t) \cdot \mathbf{1} \quad (9)$$

where $\exp(\dot{S})$ is the matrix exponential [27] of $\dot{S}$, α is the 1x7 starting state vector

$$\alpha = [0, \frac{(1-a_{l_{i+1}})(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}}, 0, \frac{a_{l_{i+1}}(1-a_{b_{i+1}})}{1-a_{l_{i+1}}.a_{b_{i+1}}}, 0, \frac{(1-a_{l_{i+1}})a_{b_{i+1}}}{1-a_{l_{i+1}}.a_{b_{i+1}}}, 0]$$

and $\mathbf{1}$ is an 7x1 vector with every element being 1.

With Equation (9), we can compute the conditional buffering delay distribution $d^+_{b_i}$ at broker b_i. Hence, we can estimate buffering delay d_{b_i} at broker b_i as

$$d_{b_i} = \begin{cases} d^+_{b_i} & \text{with probability } 1 - a_{l_{i+1}} \cdot a_{b_{i+1}} \\ 0 & \text{with probability } a_{l_{i+1}} \cdot a_{b_{i+1}} \end{cases} \quad (10)$$

Once we calculate per-hop buffering delay d_{b_i} with Equation (10), we then can calculate the end-to-end buffering delay d_{ps} for path

$\delta_{ps} = (p, l_0, b_0, l_1, b_1, ..., l_{|\delta_{ps}|-1}, b_{|\delta_{ps}|-1}, l_{|\delta_{ps}|}, s)$ as

$$d_{ps} = \sum_{i=0}^{|\delta_{ps}|-1} d_{b_i} \quad (11)$$

Hence, Equation (11) completes the calculation of pairwise reliability for static path with event buffering scheme in Equation (4).

4.4 Path Bypassing + Periodic Subscription

With the path bypassing scheme discussed in Section 2.4.3, a new routing path between p and s will be used if the old path fails. Hence, an event of a subscriber s's interest that is published by a publisher p will be delivered to s as long as the broker graph $\mathbb{G}$ is not partitioned between p and s. Thus, pairwise reliability r'_{ps} is then equal to the graph $\mathbb{G}$'s connection probability between p and s. However, the calculation of such connection probability for any generic graph is considered to be a #P-complete problem [34], which has higher complexity that a NP-complete problem.

Due to such computational complexity, we propose an algorithm to approximate the lower bound of graph $\mathbb{G}$'s connection probability between any publisher-subscriber pair (p,s) by constructing a subgraph $\mathbb{G}' \subseteq \mathbb{G}$ that consists only parallel, broker-disjoint paths between p's home broker and s's home broker.

That is, the multi-path subgraph $\mathbb{G}'$ contains multiple, broker-disjoint path between p's home broker and s's home broker, assuming there are m of such paths in subgraph $\mathbb{G}'$, namely $\delta_{ps}^{(0)}, \delta_{ps}^{(1)}, ..., \delta_{ps}^{(m-1)}$ where $\delta_{ps}^{(i)} = (p, l_0^{(i)}, b_0^{(i)}, l_1^{(i)}, b_1^{(i)}, ..., l_{|\delta_{ps}^{(i)}|-1}^{(i)}, b_{|\delta_{ps}^{(i)}|-1}^{(i)}, l_{|\delta_{ps}^{(i)}|}^{(i)}, s)$

Note that $b_0^{(i)}$ refers to the same broker for all $0 \leq i < m$, which is publisher p's home broker. Likewise, $b_{|\delta_{ps}^{(i)}|-1}^{(i)}$ refers to the same broker, which is the subscriber s's home broker. Let b_0 and $b_{|\delta_{ps}|-1}$ denote publisher p's home broker and subscriber s's home broker respectively. Hence, the pairwise reliability r'_{ps} between publisher p and subscriber s in dynamic tree scheme is estimated as

$$\begin{aligned} r'_{ps} &= \text{P}[\mathbb{G} \text{ is connected between } p \text{ and } s] \\ &\geq \text{P}[\mathbb{G}' \text{ is connected between } p \text{ and } s] \\ &\geq \text{P}[p\text{'s home broker is on}]\cdot \\ &\quad \text{P}[s\text{'s home broker is on}]\cdot \\ &\quad \text{P[at least one path is connected]} \\ &\geq a_{b_0} \cdot a_{b_{|\delta_{ps}|-1}} \cdot \\ &\quad (1 - \Pi_{i=0}^{m-1}(1 - \frac{r_{ps}^{(i)}}{a_{b_0} \cdot a_{b_{|\delta_{ps}|-1}}})) \end{aligned} \quad (12)$$

where $r_{ps}^{(i)}$ is the pairwise reliability of each path $\delta_{ps}^{(i)}$ in subgraph $\mathbb{G}'$, which can be calculated by Equation (3).

4.5 Path Bypassing + Event Buffering

We can combine the path bypassing scheme with the event buffering scheme in order to exploit path diversity as well as guarantee eventual delivery as follows. Each broker uses the event acknowledgement / buffering scheme as mentioned in Section 4.3. When a broker b_1 detects its neighbor b_2's failure, it uses the bypass routing *without* acknowledgement to forward the event to the failed broker b_2's immediately reachable children. The broker b_1 also keeps the event in its buffer and keeps retransmitting the event to the failed broker b_2 until b_2 recovers, receives the event, and sends the acknowledgement back to b_1. b_1 then discards the event. This scheme combines eventual delivery guarantee of the retransmission scheme with path diversity of the path bypassing scheme. The drawback of this approach is the additional overhead and potential event duplication at the subscribers. Event duplication, however, can be filtered out at the last-hop broker.

We can calculate the publisher-subscriber pairwise reliability for the path bypassing with event buffering scheme as follows. Let r'^{A}_{ps} be the estimated publisher-subscriber pairwise reliability for the path bypassing scheme (i.e., Equation (12) and d_{ps} be the end-to-end buffering delay for the event buffering scheme (i.e., Equation (11)). We can calculate the estimated pairwise reliability for the combined scheme as

$$\begin{aligned} r'_{ps} &= \text{P[event delivered immediately]} + \\ &\quad \text{P[partition]}\cdot\text{P[event delivered on time]} \\ &= r'^{A}_{ps} + (1 - r'^{A}_{ps}) \cdot \frac{\text{P}[d_{ps} \leq D]}{\text{P}[d_{ps} > 0]} \end{aligned} \quad (13)$$

where D is event lifetime. That is the total reliability is the probability that either the event can be delivered immediately via path bypassing scheme, or the event suffers network partition but the delay caused by the buffering scheme is still less than the event lifetime.

Note that estimated pairwise reliability r'_{ps} can be either calculated by Equation (3), Equation (4), Equation (12), or Equation (13), depending on which fault-tolerant technique is used. Once the estimated reliability r'_{ps} values of all publisher-subscriber pairs are calculated, they can be used to calculate the estimated subscriber reliability r'_s using Equation (2).

5 Evaluation

We evaluate the proposed analytical model via simulations with NS-2 network simulator [2]. In the simulator, we implement a topic-based, tree-based publish / subscribe protocol with four different reliability mechanisms described in Section 2.4 and analyzed in Section 4. In the experiment, we assume each publisher publishes its own unique topic. We assign a topic to each subscriber such

that each topic's popularity follows Power law model.

5.1 Parameter Settings

We investigated several host availability traces and reports, ranging from commercial server log to distributed testbed log [3, 4, 18, 35]. In most cases, server's time between failures tends to range from several days to weeks, while time to repair usually range within hours.

Motivated by such finding, we describe a component[5] from availability perspective by two metrics, *period* and *availability*. We define the term *period* of a component as the summation of the component's mean time between failure and mean time to repair (i.e., mean failure-repair cycle length) and the term *availability* as the fraction of time the component is on. Thus, given a component x's period PR_x and availability a_x, we can calculate x's mean time between failures $MTBF_x = a_x.PR_x$ and mean time to repair $MTTR_x = (1-a_x).PR_x$ respectively.

In the simulation, a component x will be *on* for the time period which is drawn from the exponential distribution with mean $MTBF_x$ before going to *off* state. Likewise, the component x will then be *off* for the time period drawn from the exponential distribution with mean $MTTR_x$ before going to *on* state again. Thus, such component x will have exponential failure rate $\gamma_x = \frac{1}{MTBF_x}$ and exponential repair rate $\sigma_x = \frac{1}{MTTR_x}$.

Based on the previous work [11], we set each overlay link's availability set to 0.99 and period to 60,000 seconds. Each publisher has default publishing interval equal to 1 minute. Each event has default lifetime equal to 3,600 seconds (1 hours). Each simulation is run for 14 days of simulation time. The evaluation result of each simulation parameter set is averaged from 10 runs.

5.2 Evaluation Results

We conduct the experiment with two sets of broker network topology. The first set is tree-based broker topology in order to study the performance of static path schemes (Section 4.2 and 4.3) without the effect of path diversity. The second set is random broker topology in order to study both static path schemes and path bypassing schemes (Section 4.4 and 4.5) with the effect of path diversity.

5.2.1 Tree-based Broker Network Topology

To study the performance of static-path reliability schemes (Section 4.2 and 4.3), we generate a random broker tree consisting of 10 brokers, 10 publishers, and 500 subscribers. We randomly assign a home broker to each publisher and each subscriber. We divide all generated trees into four sets. The first set of trees has each broker availability falling into [0.9, 0.95] range and use periodic subscription scheme. The second set of trees has each broker availability falling into [0.9, 0.95] range and use event buffering scheme. The third set of trees has each broker availability falling into [0.99, 0.999] range and use periodic subscription scheme. The fourth set of trees has each broker availability falling into [0.99, 0.999] range and use event buffering scheme. The [0.9, 0.95] availability range represents standard, off-the-shelf servers with low-to-moderate maintenance [3]. The [0.99, 0.999] availability range represents high-end, commercial servers with high maintenance [4].

[5] A component means a link or a broker

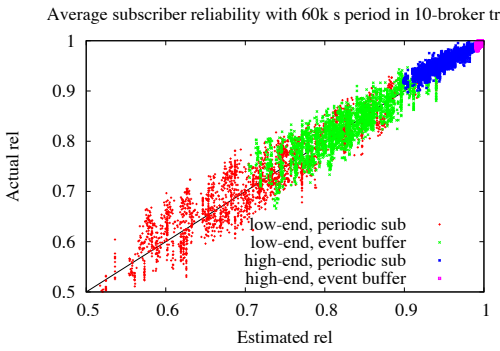
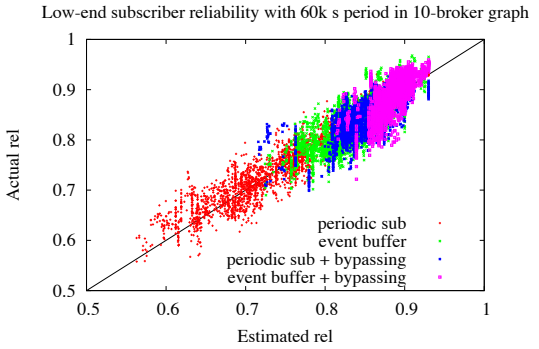


Figure 6: Subscriber reliability in static random tree with 10 brokers and 60K s period

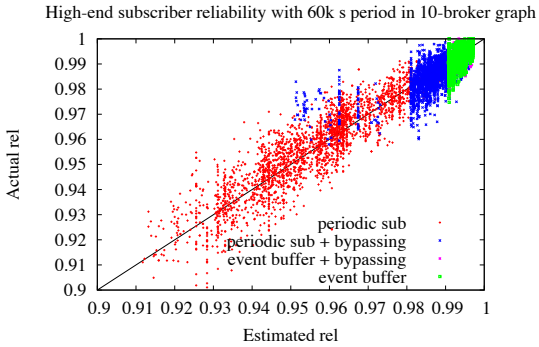
Figure 6 shows the predicted subscriber reliability on x axis and the actual subscriber reliability on y axis. Each point in the graph represents one subscriber. The color of the point represents the broker configuration and fault-tolerant scheme used. As shown in the graph, our proposed analytical model can accurately predict the subscriber reliability for each subscriber (i.e., all the points are clustered around x=y diagonal line). Also, there is a clear distinction of reliability value between different groups of broker configuration. The group with lowest reliability is the low-end servers with periodic subscription scheme, followed by the low-end servers with event buffering scheme. Notice that the event buffering scheme could achieve high reliability than the periodic subscription scheme, although the performance gain effect may be less, compared to the performance gain from the server's quality.

5.2.2 Random Broker Network Topology

We generate a random graph consisting of 10 brokers, 10 publishers, and 500 subscribers. Publishers and subscribers are randomly assigned to each broker. Again, we run the simulations with two broker availability specifications named *low-end* (0.9 - 0.95 availability) and *high-end* (0.99 - 0.999 availability). We compare the perfor-



(a) Low-end brokers



(b) High-end brokers

Figure 7: Subscriber reliability in 10-broker overlay graph with average degree 4 and 60K s period

mance in terms of subscriber reliability among four fault-tolerant techniques (i.e., from Section 4.2 - 4.5).

Figure 7 shows the performance comparison between the four protocols in 10-broker overlay graph. Again, our analytical model accurately predicts the real-time reliability of each subscriber. For the low-end broker configuration, the path bypassing scheme with event buffering has the best performance, followed by the path bypassing scheme with periodic subscription, the static path scheme with event buffering, and the static path scheme with periodic subscription respectively. However, for the high-end broker configuration, the static path scheme with event buffering performs best and as well as the path bypassing scheme with event buffering. This finding suggests that one should prefer to use the static path scheme with event buffering in high-end broker configuration, as it has lower overhead than the path bypassing scheme with event buffering.

6 Related Work

Improving reliability, timeliness, and other QoS metrics in wide-area overlay networks have been a significant topic of researchs for many years [5, 6, 12]. However, the approaches in this category are designed for point-to-point routing and do not specifically address decoupling nature between publishers and subscribers in publish/subscribe systems.

Several fault-tolerant mechanisms have been proposed specially for publish/subscribe systems under failures without considering timeliness property [8, 10, 13, 23]. On the other hand, several works have proposed performance analytical model to predict timeliness in publish/subscribe systems in perfect scenarios (i.e., without broker/link failures) [24, 31, 32]. This paper bridges such two approaches by quantitatively analyzing commonly used fault-tolerant techniques and their effect to *both* reliability and timeliness of publish/subscribe systems. There have been a few works that discuss timeliness of event delivery in publish/subscribe systems under component failures [14, 20]. However, they did not provide analytical model of their proposed systems. Recently, Esposito et al proposed and analyzed the use of network coding and gossiping to provide reliability and timeliness in tree-based publish / subscribe systems [15]. In contrast, our model discusses more commonly used fault-tolerant techniques such as path diversity and buffering. Also, our work does not assume tree-based publish / subscribe, but also works for any generic broker topology.

7 Conclusions

In this paper, we proposed an analytical model to estimate the subscriber real-time reliability for publish / subscribe systems with faulty brokers and links. We first described broker failure and link failure model before discussing several existing fault-tolerant techniques for distributed publish / subscribe systems. We then proposed the generic analytical model to estimate subscriber reliability. The evaluation via simulation has proved the correctness of our predictive model. Our proposed model can then be used as a building block for optimization problems such as subscriber assignment problem or broker network planning problem.

There are a few possible directions for future works. The first direction is to use the proposed analytical model to optimize performance of the publish / subscribe systems. Second, the analytical model could be extended to the case where component failure time is not exponentially distributed. Finally, another possible direction is to validate the proposed analytical model using data collected from real systems.

References

[1] Apache logging services. http://logging.apache.org/.

[2] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[3] Planetlab - all pairs pings. http://pdos.csail.mit.edu/ strib/pl_app/.

[4] Site5 uptime reports for all servers. http://www.site5.com/support/uptime/.

[5] AMIR, Y., AND DANILOV, C. Reliable communication in overlay networks. In *DSN* (2003), IEEE Computer Society, pp. 511–520.

[6] ANDERSEN, D. G. *Improving End-to-End Availability Using Overlay Networks*. Ph.D., Massachusetts Institute of Technology, Feb. 2005.

[7] ARIANFAR, S. Optimizing publish/subscribe systems with congestion handling. Master's thesis, Helsinki University of Technology, 2008.

[8] CHAND, R., AND FELBER, P. Xnet: A reliable content-based publish/subscribe system. In *SRDS '04: Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 264–273.

[9] CUGOLA, G., AND JACOBSEN, H.-A. Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev. 6*, 4 (2002), 25–33.

[10] CUGOLA, G., PICCO, G. P., AND MURPHY, A. L. Towards dynamic reconfiguration of distributed publish-subscribe middleware. In *SEM* (2002), A. Coen-Porisini and A. van der Hoek, Eds., vol. 2596 of *Lecture Notes in Computer Science*, Springer, pp. 187–202.

[11] DAHLIN, M., CHANDRA, B. B. V., GAO, L., AND NAYATE, A. End-to-end wan service availability. *IEEE/ACM Trans. Netw. 11*, 2 (2003), 300–313.

[12] DUAN, Z., ZHANG, Z.-L., AND HOU, Y. T. Service overlay networks: Slas, qos, and bandwidth provisioning. *IEEE/ACM Trans. Netw. 11*, 6 (2003), 870–883.

[13] ESPOSITO, C., COTRONEO, D., AND GOKHALE, A. S. Reliable publish/subscribe middleware for time-sensitive internet-scale applications. In *DEBS* (2009), A. S. Gokhale and D. C. Schmidt, Eds., ACM.

[14] ESPOSITO, C., COTRONEO, D., AND RUSSO, S. Reliable event dissemination over wide-area networks without severe performance fluctuations. In *ISORC '10: Proceedings of the 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing* (Washington, DC, USA, 2010), IEEE Computer Society, pp. 97–101.

[15] ESPOSITO, C., RUSSO, S., BERALDI, R., PLATANIA, M., AND BALDONI, R. Achieving reliable and timely event dissemination over wan. In *Proceedings of the 13th international conference on Distributed Computing and Networking* (Berlin, Heidelberg, 2012), ICDCN'12, Springer-Verlag, pp. 265–280.

[16] EUGSTER, P. T., FELBER, P. A., GUERRAOUI, R., AND KERMARREC, A.-M. The many faces of publish/subscribe. *ACM Comput. Surv. 35*, 2 (2003), 114–131.

[17] GERHARDS, R. The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009.

[18] GODFREY, P. B. Repository of availability traces. http://www.cs.uiuc.edu/homes/pbg/availability/.

[19] JAEGER, M. A. *Self-Managing Publish/Subscribe Systems*. PhD thesis, Technische Universität Berlin, 2007.

[20] JERZAK, Z., AND FETZER, C. Soft state in publish/subscribe. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems* (New York, NY, USA, 2009), ACM, pp. 1–12.

[21] KALBFLEISCH, J. D., AND PRENTICE, R. L. *The statistical analysis of failure time data*. Wiley-Interscience, 2011.

[22] KAZEMZADEH, R. S., AND JACOBSEN, H.-A. Reliable and highly available distributed publish/subscribe service. In *SRDS '09: Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 41–50.

[23] KAZEMZADEH, R. S., AND JACOBSEN, H.-A. Opportunistic multipath forwarding in content-based publish/subscribe overlays. In *Middleware* (2012), pp. 249–270.

[24] KOUNEV, S., SACHS, K., BACON, J., AND BUCHMANN, A. P. A methodology for performance modeling of distributed event-based systems. In *ISORC* (2008), pp. 13–22.

[25] LATOUCHE, G., AND RAMASWAMI, V. *Introduction to Matrix Analytic Methods in Stochastic Modelling*, 1 ed. ASA SIAM, 1999, ch. 2: PH Distributions.

[26] LI, M., YE, F., KIM, M., CHEN, H., AND LEI, H. A scalable and elastic publish/subscribe service. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium* (Washington, DC, USA, 2011), IPDPS '11, IEEE Computer Society, pp. 1254–1265.

[27] MOLER, C., AND LOAN, C. V. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review* (1978), 801–836.

[28] MUHL, G. *Large-scale Content-based Publish/Subscribe Systems*. PhD thesis, University of Technology Darmstadt, 2002.

[29] PONGTHAWORNKAMOL, T., AND NAHRSTEDT, K. Towards timeliness and reliability analysis of distributed content-based publish/subscribe systems over best-effort networks. Tech. Rep. http://hdl.handle.net/2142/14415, University of Illinois at Urbana-Champaign, November 2009.

[30] PONGTHAWORNKAMOL, T., NAHRSTEDT, K., AND WANG, G. Probabilistic qos modeling for reliability/timeliness prediction in distributed content-based publish/subscribe systems over best-effort networks. In *ICAC '10: Proceeding of the 7th international conference on Autonomic computing* (New York, NY, USA, 2010), ACM, pp. 185–194.

[31] SACHS, K. *Performance Modeling and Benchmarking of Event-Based Systems*. PhD thesis, TU Darmstadt, 2010. SPEC Distinguished Dissertation Award 2011.

[32] SCHRÖTER, A., MÜHL, G., KOUNEV, S., PARZYJEGLA, H., AND RICHLING, J. Stochastic performance analysis and capacity planning of publish/subscribe systems. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems* (New York, NY, USA, 2010), DEBS '10, ACM, pp. 258–269.

[33] SHANKAR, A. U., AND LAM, S. S. Time-dependent distributed systems: Proving safety, liveness and real-timeproperties. Tech. rep., Austin, TX, USA, 1985.

[34] VALIANT, L. G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing 8*, 3 (1979), 410–421.

[35] YALAGANDULA, P., NATH, S., YU, H., GIBBONS, P. B., AND SESHAN, S. Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems. In *In Proc. of USENIX Workshop on Real, Large Distributed Systems (WORLDS* (2004).

[36] ZHANG, C., KRISHNAMURTHY, A., WANG, R. Y., AND SINGH, J. P. Combining flexibility and scalability in a peer-to-peer publish/subscribe system. In *Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware* (New York, NY, USA, 2005), Middleware '05, Springer-Verlag New York, Inc., pp. 102–123.