Towards Timeliness and Reliability Analysis of Distributed Content-based Publish/Subscribe Systems over Best-effort Networks

Thadpong Pongthawornkamol, Klara Nahrstedt
University of Illinois at Urbana-Champaign
{tpongth2,klara}@cs.uiuc.edu

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

Abstract

Content-based publish/subscribe is a powerful data dissemination paradigm that offers both scalability and flexibility. However, its nature of high expressiveness makes it difficult to analyze or predict the behavior of the system such as event delivery probability and end-to-end delivery delay, especially when deployed over unreliable, best-effort public networks. This paper proposes the analytical model that abstracts both expressiveness nature of content-based publish/subscribe systems, along with uncertainty of underlying networks, in order to predict quality of service in terms of delivery probability and timeliness based on partial, imprecise statistical attributes of each component in the system. The evaluation results yields good prediction accuracy even when each component's information is imprecise.

1 Introduction

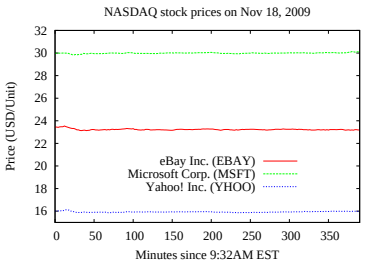
Over the past few years, publish/subscribe systems have recently become an emerging paradigm for large-scale information dissemination. The nature of publish/subscribe where the producers of the information (i.e. publishers) and the consumers of the information (i.e. subscribers) are interacting via intermediaries (i.e. brokers) allows both sides of the communication to be decoupled in space, time, and synchronization [9]. Such flexibility and scalability makes publish/subscribe paradigm one of few viable choices for designing and building large-scale data dissemination systems.

So far, there have been significant efforts from both academia and industry domains to design standards and build implementations of scalable and efficient distributed publish/subscribe systems [5–8, 10, 17, 20, 21, 24, 25, 25]. Based on commonly accepted taxonomy, publish/subscribe systems can be categorized into *topic-based* publish/subscribe systems [6, 20, 25] and *content-based* publish/subscribe systems [7, 8, 10, 17, 21, 24]. In topic-based publish/subscribe systems, the event from publishers are delivered to subscribers that share the same single interest value called *topic*. In content-based publish/subscribe systems, each event can contain multiple attributes. Any subscriber that is interested in a topic can further specify, at the attribute level, which portion of the events in the topic that it wants to receive. Content-based publish/subscribe systems hence give more flexibility to the subscribers at the cost of increasing processing complexity at brokers.

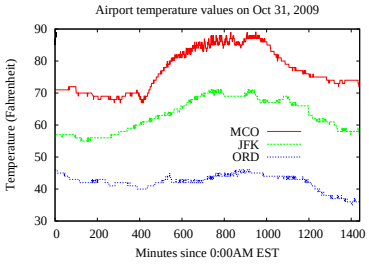
Besides the increasing complexity compared to topic-based publish/subscribe systems, another drawback of content-based publish/subscribe systems is a lack of predictability. Since each subscriber has flexibility in choosing information it wants at fine-grained attribute level, it is less trivial to determine the event flow from each publisher to each subscriber. Hence, it is also less trivial to analyze the quality-of-service performance and correctness of a content-based publish/subscribe system when compared to its topic-based counterpart. For example, it is less trivial to check how much resource is needed to service each subscriber properly, or to verify if the system's current state is stable. Moreover, deploying publish/subscribe systems over unreliable networks further decreases system determinism and predictability. Such uncertainty and complexity becomes a hindrance in applying content-based publish/subscribe systems to delay-sensitive applications that require quality of service and tight resource control such as soft real-time systems or cyber-physical systems. The need to solve such problem calls for a good analytical model that could capture expressiveness and uncertainty nature of content-based publish/subscribe systems yet predict the system behavior accurately.

However, while it is infeasible to calculate *exact* resource consumption and quality of service each subscriber receives in content-based publish/subscribe systems, it is still possible to do so in probabilistic manner when some *partial* information of each component in the system is given to some extent. The term *partial* information refers to trend or pattern of behavior of each component, ranging from underlying networks (i.e. how likely that a message will be transmitted over a link within 5 seconds), hardware capabilities (i.e. the average broker event processing time),

to the information pattern (i.e. how likely a publisher will publish a value or how likely that a publisher will publish the next message within a specific time). Many real-world event publishers exhibit temporal locality such that content pattern prediction can be done based on previously published events (i.e. Figure 1 for examples). Such pattern information can be either explicitly given by or implicitly observed from each component [12], thus making it possible to model and predict behavior of the publish/subscribe system as a whole. The performance prediction can then be used as a building block for quality of service control such as subscriber admission control, broker capacity planning, and resource adaptation.



(a) NASDAQ stock prices (Source: Google Finance [1])



(b) Airport temperature report (Source: NCDC [2])

Figure 1. Example of real-world event streams and their temporal locality

In this paper, we explore the possibility to use such imperfect information to predict event delivery delay and reliability in a distributed content-based publish/subscribe system by applying the techniques from probability theory and queuing theory. Specifically, our work has the following contributions. First, we propose a novel analytical model for generic, *existing* distributed content-based publish/subscribe systems for the purpose of quality of service and performance assessment. Second, we present an algorithm to estimate subscriber's quality of service in terms of reliability and timeliness based on the proposed analytical model and the assumption of imperfect statistics information of each component. Third and finally, we present the simulation results of the proposed analytical model under realistic parameters. The evaluation results yield good prediction accuracy even when the statistics information of each component is inaccurate.

This paper is organized as follows. Section 2 discusses the model and assumption of distributed, content-based publish/subscribe system used in this work. Section 3 proposes the mathematical model of content-based publish/subscribe system along with the subscriber reliability prediction problem formulation. Section 4 presents the algorithm to predict subscriber real-time reliability based on the proposed model. Section 5 presents the evaluation results in terms of prediction accuracy of the analytical model. Section 6 discusses related works in quality of service and modeling of publish/subscribe systems. Section 7 concludes the paper.

2 System Model

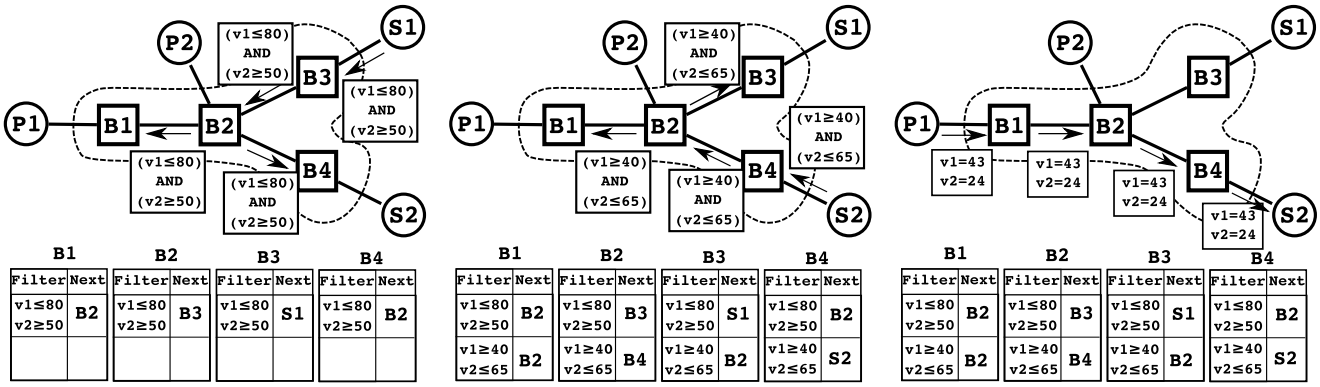
In this section, we first describe the model of soft real-time distributed content-based publish/subscribe model used in our work. We then formulate the problem of subscriber reliability based on the described model.

2.1 Distributed Real-time Publish/Subscribe Model

In this work, we assume acyclic publish/subscribe broker tree networks commonly adopted by existing publish/subscribe system works [7, 8, 17, 24]. A publish/subscribe system consists of a group of *subscribers* (information consumers) and *publishers* (information providers) connected via a network of *brokers* (information intermediaries). Each broker is connected to at least one neighbors to form a tree network (i.e. there is only one path between each pair of broker). Each subscriber/publisher is connected to only one of the brokers in the system. The broker connected to a subscriber/publisher is called the *home broker* with respect to that subscriber/publisher. Each publisher publishes *events* or *messages* to its home broker. Each published event has one or more *attributes* with the associated *value*. Each event also has its *lifetime* value, which is the duration between the time the event was published and the time the event is expired. An event is said to be delivered to a subscriber *on time* if the end-to-end delivery delay is *less* than its lifetime.

The subscriber/publisher joining process and event/subscription matching process in the publish/subscribe systems are shown in Figure 2 as follows. When a new subscriber joins the system, it sends its subscription to one of the brokers (Figure 2(a)). A subscription contains *predicate filter*[1] specifying event attribute content

[1]In Figure 2(a), each predicate filter is in conjunctive form consisting of per-attribute min-max clauses. However, our analytical model supports



(a) Subscriber S_1 subscribes (b) Subscriber S_2 subscribes (c) Publisher P_1 publishes an event

Figure 2. Example of subscription propagation and event routing in a publish/subscribe system

that the subscriber wants to receive. Upon receiving the subscription from the subscriber, the broker stores the subscription into its routing table and propagates the new subscription to its adjacent brokers, which in turn repeat the process until all brokers receive the subscription (Figure 2(a) and 2(b)). When storing a new subscription into its routing table, each broker also stores the link information to the broker which it receives the subscription from. When a broker receives a newly-published event (Figure 2(c)), it checks the event with each subscription stored in its routing table. For each matching subscription, the broker forwards the event to the link which it receives that subscription from. Note that an event is forwarded *once* per link even there are multiple matching subscriptions from that link. The process then continues, and the event is propagated hop-by-hop in the reverse direction of the subscription until it reaches the designated subscribers. The mentioned publish/subscribe model is simple yet generic enough to represent a variety of existing publish/subscribe system works [7, 8, 17, 24]. Further optimization techniques on top of this basic model such as subscription covering or subscription merging [14] are beyond the scope of this paper and considered as future directions.

Another assumption made in this paper is that the probabilistic information of each publish/subscribe component such as the publisher content distribution (i.e. what content a publisher is more likely to publish), inter-broker link delay and bandwidth distribution, and broker event processing time are known a priori either via explicit advertisements from publishers or implicit prediction based on statistical history [12].

all possible forms of filter as long as the filter can be expressed a a subset of the attribute content space.

2.2 Real-time Subscriber Reliability Problem

With the presented content-based publish/subscribe model, one question that may arise is that, given a publish/subscribe system setting along with all subscribers and their subscriptions, how much quality of service each subscriber can have? Specifically, *what fraction of events that match a given subscriber's interests will be delivered to that subscriber on time?* To quantify such quality of service, we define a subscriber-level metric called *subscriber real-time reliability* as follows.

Subscriber Real-time Reliability : A subscriber s is said to receive the service with real-time reliability R_s, 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).

Since the proposed real-time subscriber reliability combines the concept of standard reliability with the concept of timeliness property , it can be used as a good indicator how much quality of service each subscriber receives.

2.3 Network Model

Each broker is linked via asynchronous, non real-time, wired communication link. Inter-broker links can fail with some probability. The broker/publisher and broker/subscriber links can be either wired or wireless links.

As mentioned earlier, we assume tree, acyclic topology of broker networks, which means there is only one communication path between each pair of broker. More complex topologies such as cyclic networks are considered as future direction.

3 Mathematical Formulation

In order to analytically estimate subscribers' real-time reliability, we present a novel mathematical model of the proposed content-based publish/subscribe system as follows. All notations can also be found in Table 1.

Event Model : Let $\mathbb{E}$ be the set of all events published in the system. An event $e \in \mathbb{E}$ is defined as (id_e, a_e, d_e) , which represent event's identifier, content attributes, and lifetime duration respectively. The content of an event e, denoted by a_e, is defined as a tuple

$$a_e = (v_{1e}, v_{2e}, .., v_{ke})$$

, where v_{ie} is the value of the i^{th} attribute of event e. For simplicity of the analysis, we assume that the event topic (τ_e) is always the first attribute (v_{1e}) and the rest $k-1$ attributes are *the union of all per-topic attributes* in the system in an arbitrary but globally consistent order. Hence, an event of any topic in the system can be expressed with such k attributes by setting irrelevant attributes from other topics to null value.

Let V_i be the value space of the i^{th} attribute of any event ($\forall e \in \mathbb{E} : v_{ie} \in V_i$). Let T be the set of all topics in the system (i.e. $\mathrm{T} = V_1$). Let D be the set of all possible lifetime duration values of events in the system. Note that V_i and D can be either discrete or continuous. Without loss of generality in the analysis, we assume V_i to be discrete in this work. However, the proof also applies to the continuous case.

We define

$$V = \mathrm{T} \times V_2 \times .. \times V_k$$

as the content space of the events in the system.

Subscriber Model : A subscriber s is defined as a tuple

$$s = (id_s, f_s)$$

where id_s is the subscriber's identifier, $f_s \subseteq V$ is the predicate filter defining the content of interest for s. We define a filter set $F_s(E)$ of an event set E with respect to subscriber s as

$$F_s(E) = \{e \in E : a_e \in f_s\}$$

Publisher Model : A publisher p is defined by a tuple

$$p = (id_p, C_p(a, d), I_p(t))$$

where $C_p : V \times D \rightarrow [0, 1]$ is the content-lifetime joint distribution function of events that p publishes (i.e. $C_p(a, d)$ is the probability that p will publish an event with content a and lifetime d), $I_p : \mathbb{R} \rightarrow [0, 1]$ is the inter-event publishing time distribution (i.e. $I_p(t)$ is the probability that the time between p's successive event publications is t), and id_p is the publisher's identifier. Thus

$$\sum_{(a,d) \in V \times D} C_p(a, d) = 1$$

and

$$\sum_{t>0}^{\infty} I_p(t) = 1$$

Both Content-lifetime distribution function $C_p(a, d)$ and inter-event publishing time distribution function $I_p(t)$ can be obtained via statistical estimation from publisher p's publishing history [12].

Broker Model : Each broker in the system has a single event queue that is used to store and match events on a first-come-first-serve basis. A broker b in the system is defined as a tuple

$$b = (id_b, M_b(t))$$

where id_b is the broker's identifier, and $M_b : \mathbb{R} \rightarrow [0, 1]$ is the distribution of broker's event processing (matching and routing) time. For example, $M_b(100\text{ms}) = 0.2$ means with 20% probability, the delay the broker b will take to retrieve an event from its queue and route the event to the appropriate links is 100 milliseconds. Note that the event processing time distribution $M_b(t)$ can be a function that depends on the number of subscriptions stored in broker b's routing table. $M_b(t)$ can be obtained from performance profiling of the broker node.

Publish/Subscribe Network Model : We model the publish/subscribe network as a directed acyclic graph $G(V = (B \cup P \cup S), L)$, where $V = B \cup P \cup S$ is the set of brokers, publishers, and subscribers in the system, and $L \subseteq (P \cup B) \times (B \cup S)$ is the set of directed communication links. Each communication link $l \in L$ is a directed edge that dictates the direction of event flows either from a publisher to a broker (i.e. *publisher-broker* link), a broker to another broker (i.e. *broker-broker* link), or a broker to a subscriber (i.e. *broker-subscriber* link). Each link l has reliability r_l and link delay distribution $D_l(t)$, which can be collected statistically via active or passive probing. We define $out(l)$ and $in(l)$ as the source and the sink of link l respectively. Note that a bi-directional link between two brokers will be modeled as two directional links. For example, Figure 3 shows the analytical view of the publish/subscribe system from Figure 2.

Subscriber Real-time Reliability : Let $\mathbb{E}_s$ be the set of all events that are published during a subscriber s's lifetime in the system. Hence, $F_s(\mathbb{E}_s)$ is the set of all events that match s's interest during its lifetime in the system. For each event $e \in F_s(\mathbb{E}_s)$, let d_e^s be the *delivery delay* of event e to subscriber s (the time period between e's publishing time and

Symbol	Definition
$e \in \mathbb{E}$	an event in the set of all system events
d_e	event e's lifetime
D	set of all events' lifetime values
a_e	event e's attributes
k	number of all attribute types in the system
τ_e	event e's topic (v_{1e})
V_i	value space of i^{th} attribute
V	content space of all events
$s \in S$	a subscriber in the set all subscribers
$f_s \in V$	subscriber s's content of interest
$F_s(E)$	a set of events in E that matches s's interest
d_e^s	end-to-end delivery delay of event e to subscriber s
R_s	subscriber s's real-time reliability
R'_s	subscriber s's estimated real-time reliability
$p \in P$	a publisher in the set of all publishers
$C_p(a,d)$	content-lifetime distribution of events published by p
$I_p(t)$	publisher p's event publishing interval distribution
$b \in B$	a broker in the set of all brokers
$M_b(t)$	broker b's event processing time distribution
$l \in L$	a directed communication link
r_l	link l's transmission reliability
$D_l(t)$	link l's successful transmission delay distribution
$in(l)$	link l's sink node
$out(l)$	link l's source node

Table 1. Model variables' notation

time that e is delivered to s). Thus, the real-time reliability at a subscriber s, denoted by R_s, can be expressed as

$$R_s = \frac{|\{e \in F_s(\mathbb{E}_s) : d_e^s \leq d_e\}|}{|F_s(\mathbb{E}_s)|}$$

In the other word, R_s is the fraction of all messages matching s's interest that are delivered to s on time. However, we would like to estimate R_s for each subscriber s without actually running the system, which leads to the subscriber real-time reliability estimation problem.

Subscriber Real-time Reliability Estimation Problem: Given a publish/subscribe network $G = (B \cup P \cup S, L)$, find the estimated value of R_s, denoted by R'_s, for each subscriber $s \in S$.

In the next section (Section 4), we will discuss how to use the proposed analytical model, along with probability theory and queuing theory, to analytically solve the subscriber real-time reliability estimation problem.

4 Subscriber Real-Time Reliability Estimation

In this section, we present the subscriber reliability estimation algorithm. The estimation algorithm takes the publish/subscribe graph $G = (B \cup P \cup S, L)$ along with the statistical information of each component as the input and estimates the real-time reliability value R_s for each subscriber $s \in S$. To do so, it is necessary to estimate the end-to-end delivery delay distributions and path reliability distributions of all s's matching events when they arrive at s. Hence, we introduce another set of variables in Table 2 for the purpose

Symbol	Definition
f_l	union of all subscription filters propagated via link l
λ_l	estimated event flow rate through link l
λ_p	estimated event flow rate from publisher p
$C_l(a)$	estimated content distribution of events through link l
$up(l)$	upstream links of link l (Equation (2))
λ_b	estimated event flow rate to broker b
μ_b	estimated event processing rate at broker b
q_b	estimated queuing delay at broker b
$D_b(t)$	estimated total delay distribution at broker b
$C_l(a,d)$	estimated content-remain time distribution of events through link l

Table 2. Analysis variables' notation

of the analysis. These variables are not parts of the problem definition, but are defined as intermediate variables in order to solve the estimation problem. The overall estimation process, depicted in Figure 3, consists of four steps : propagating subscription filters, calculating per-link event flow rate, calculating broker queuing/processing delay, and calculating per-link content-remain time distribution.

4.1 Subscription Filter Propagation

In this step, the subscription filters are propagated from subscribers to each broker in the system in the same manner as subscription propagation process discussed in Section 2.1. As shown in Figure 3(a), each subscription is propagated in the reverse direction of the event flow direction (i.e. reversed to the direction of the arrows). When a subscription filter f is propagated to a broker b via one of b's outgoing links l, the subscription will be propagated to all other incoming links of b. At the same time, the subscription filter f will be merged into l's *filter set*, denoted by f_l. That is, for each new filter f that is propagated to link l, $f_l = f_l \cup f$. The process continues until all subscriptions are propagated to all brokers the system.

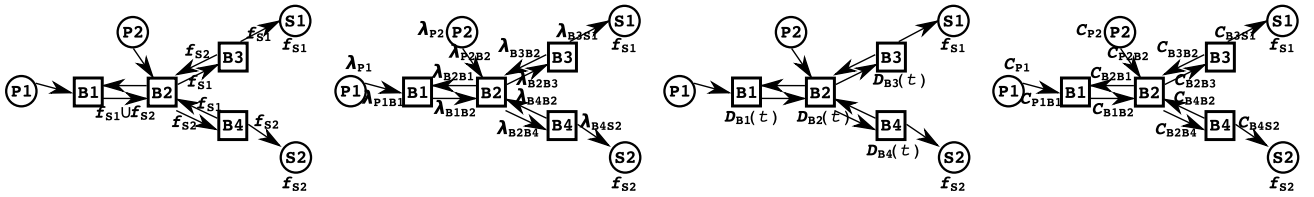
The filter set f_l can be viewed as the union of all subscriptions that are propagated through link l and hence represents the content space of the events that should be forwarded to link l. At the beginning of this step, each link l has its filter set empty (i.e. $f_l = \emptyset$). At the end of this step, if any link l's filter set still remains empty, then it means that there will be no event sent over that link.

4.2 Per-link Event Flow Rate Calculation

After each link's filter set is identified, the next step is to calculate each link l's average event flow rate λ_l. This step starts by calculating the average event generation rate at each publisher p, denoted by λ_p, as the inverse of average inter-event generation time $I_p(t)$ as follows.

$$\lambda_p = \frac{1}{\mathrm{E}[I_p(t)]} = \frac{1}{\sum_{t:I_p(t)>0}(t.I_p(t))}$$

The average event flow rate of a publisher-broker link l



(a) Propagating subscription (reversed direction of arrows)

(b) Calculating link event rate

(c) Calculating broker total delay distribution

(d) Calculating per-link content-remain time distribution

Figure 3. The steps for subscriber reliability estimation

is then equal to the event flow rate of l's source publisher, multiplied by the link's reliability r_l as follows.

$$\lambda_l = r_l.\lambda_p \tag{1}$$

, where $p = out(l)$

The process continues until the event flow rates of all publisher-broker links are determined. Then, the event flow rates of the other links (i.e. broker-broker links and broker-subscriber links) are calculated. To do so, the *content distribution* of each publisher-broker link is needed. The content distribution of a link l, denoted by $C_l(a)$, is the probability distribution of the event content that passes through link l. For each publisher-broker link l that connects a publisher p, the content distribution is equal to the content projection of the p's content-lifetime distribution as follows.

$$C_l(a) = \sum_{d>0} C_p(a, d)$$

, where $p = out(l)$

A link is called *resolved* if its average flow rate and content distribution are identified. Hence, after all publisher-broker links are resolved, the other links' average flow rates and content distributions are then calculated as follows. We defined the upstream links of a link l, denoted by $up(l)$, as the set of incoming links to l's source broker except l's reversed link. In the other words,

$$up(l) = \{l' \in L : in(l') = out(l) \wedge out(l') \neq in(l)\} \tag{2}$$

That is, $up(l)$ refers to all l's adjacent links from which events potentially flow to l. For example, in Figure 3, link *B2B1*'s upstream links are link *P2B2*, *B3B2*, and *B4B2*.

Any broker-broker or broker-subscriber link l is defined as *resolvable* if and only if all l's upstream links are resolved. For each resolvable link l, its average flow rate λ_l and content distribution $C_l(a)$ can be calculated by the following equation.

$$\lambda_l = r_l.\lambda.\sum_{a \in f_l} C'(a) \tag{3}$$

and

$$C'_l(a) = \frac{r_l.\lambda.C'(a)}{\lambda_l}, \forall a \in f_l$$

where λ and $C_(a)$ are the total rate and total content distribution of all l's upstream links. Specifically,

$$\lambda = \sum_{l' \in up(l)} \lambda_{l'} \tag{4}$$

and

$$C(a) = \frac{\sum_{l' \in up(l)} \lambda_{l'}.C_{l'}(a)}{\lambda}$$

That is, l's average flow rate λ_l is calculated from the total rate of all l's incoming event flows that match the filter set f_l. The content distribution $C_l(a)$ is then calculated in the same manner.

Once a resolvable link's flow rate and content distribution is identified, that link then becomes a resolved link. The process then continues to resolve the remaining links until all links are resolved. Since we assume the broker network to be acyclic, it is guaranteed that the process always find a new resolvable link until all links are resolved.

4.3 Broker Total Delay Calculation

After all the links are resolved, we then determine the average queuing delay at each broker. Since we model each broker as an event matching server with a single queue, we can apply queuing theory techniques to determine broker queuing delay as follows. A broker b's average queuing delay, denoted by q_b can be calculated based on M/M/1 queuing model as follows.

$$q_b = \frac{\lambda_b}{\mu_b(\mu_b - \lambda_b)} \tag{5}$$

where

$$\lambda_b = \sum_{l \in L: in(l)=b} \lambda_l \quad (6)$$

and

$$\mu_b = \frac{1}{\mathrm{E}[M_b(t)]} = \frac{1}{\sum_{t:M_b(t)>0}(t.M_b(t))} \quad (7)$$

In the other words, λ_b is the total event flow rates from all of b's incoming links, and μ_b is b's average matching rate.

Note that if the event flow rate λ_b is more than the average matching rate μ_b, then the broker b is overloaded. In such case, the queuing delay at broker b will be equal to infinity, as the broker will never reach the stable state.

Once b's average queuing delay is determined, we then estimate b's total broker delay distribution $D_b(t)$ as

$$\forall t : M_b(t) > 0, D_b(t + q_b) = M_b(t)$$

That is, the total broker delay distribution is estimated as the event processing delay distribution plus the average queuing delay. Although the proposed approach is a simple delay distribution estimation based on the assumption of M/M/1 queue model, the evaluation result presented in Section 5 yields reasonably accurate results for other queue model as well. To further improve the delay estimation accuracy, more sophisticated techniques in queuing theory can be used [23].

4.4 Per-link Content-remain time Distribution Calculation

After the queuing and matching delay distributions at all brokers are identified, the last step is to calculate the content and lifetime distribution at each link. To do so, we define content-remain time joint distribution at each link l, denoted by $C_l(a, d)$, as the joint probability of the content and remaining lifetime of each event that passes through link l. Note that it is possible that $C_l(a, d) > 0$ when d is negative, which means that such fraction of events is already expired after they pass through link l.

As shown in Figure 3(d), the process at this step is similar to per-link event flow rate calculation described in Section 4.2, except that both content and lifetime are now considered in the calculation. Specifically, for each publisher-broker link l, the content-remain time distribution $C_l(a, d)$ is calculated as

$$C_l(a, d) = \sum_{t:D_l(t)>0} (D_l(t).C_p(a, d + t)) \quad (8)$$

, where $p = out(l)$ and $D_l(t)$ is l's link delay distribution. The reason behind Equation (8) is that once an event is transmitted via link l, its remaining lifetime is shortened by link l's transmission delay.

Here we once again use the concept of resolved link and resolvable link from Section 4.2, except that in this section, a link l is resolved when its content-delay distribution is identified. Hence, we apply Equation (8) to all publisher-broker links, making all of them resolved. We then repetitively find a resolvable link l and calculate its content-remain time distribution as follows.

$$C_l(a, d) = \frac{r_l.\lambda}{\lambda_l}. \sum_{t:D_l(t)>0} (D_l(t).C(a, d+t)), \forall a \in f_l \quad (9)$$

, where

$$C(a, d) = \sum_{t:D_b(t)>0} \frac{D_b(t). \sum_{l' \in up(l)} \lambda_l.C_l(a, d + t)}{\lambda} \quad (10)$$

, where λ is calculated from Equation (4).

Hence, the estimated reliability R'_s can then be calculated as

$$\begin{aligned} R'_s &= \frac{\text{rate of unexpired matching events delivered to } s}{\text{total rate of all events that match } s\text{'s interest}} \\ &= \frac{\lambda_l. \sum_{(a \in f_s, d>0)} C_l(a, d)}{\sum_{a \in f_s}(\sum_{p \in P}(C_p(a).\lambda_p))} \end{aligned} \quad (11)$$

where l is the link to s (i.e., $s = in(l)$)

With Equation (11), we can calculate the estimated real-time reliability R'_s at each subscriber s from publish/subscribe network $G = (B \cup P \cup S, L)$.

4.5 Improved Reliability Estimation with G/G/1 Queue Model

So far, the load estimation at each broker presented in this section uses M/M/1 queue model, which assumes event inter-arrival time distribution and broker processing time distribution to be exponential random variables. Such assumption may not result in accurate subscriber reliability estimation as each event inter-arrival time and broker processing time may be drawn from other distributions than exponential distribution. For example, the event intern-arrival time may be deterministic (i.e. publishers with periodic sensors) or the broker event processing time may be uniform (i.e. brokers matching a random event with an array of subscriptions). To address complex time distribution for more accurate reliability estimation, this section presents the improved estimation algorithm based on G/G/1 queue model.

To model the system using G/G/1 model, we introduce additional analytical variables as follows. Apart from event flow rate λ_p at each publisher p, another variable called

event *flow burstiness*, denoted by z_p^2, is calculated from p's event inter-arrival time distribution $I_p(t)$ as

$$z_p^2 = \frac{\text{Var}[I_p(t)]}{\text{E}[I_p(t)]^2} = \frac{\sum_{t:I_p(t)>0} I_p(t).(t - \frac{1}{\lambda_p})^2}{(\frac{1}{\lambda_p})^2} \quad (12)$$

The burstiness variable z_p^2 hence represents the uniformity level of event generation interval at p. For example, $z_p^2 = 0$ when $I_p(t)$ is a uniform distribution and $z_p^2 = 1$ when $I_p(t)$ is an exponential distribution.

Also, at each pub/sub broker b, the burstiness variable z_b^2 is calculated from its event matching time distribution $M_b(t)$ in the same way z_p^2 is calculated at each publisher p. That is,

$$z_b^2 = \frac{\text{Var}[M_b(t)]}{\text{E}[M_b(t)]^2} = \frac{\sum_{t:M_b(t)>0} M_b(t).(t - \frac{1}{\mu_b})^2}{(\frac{1}{\mu_b})^2} \quad (13)$$

With the event generation burstiness variable z_p^2 at each publisher p and the event matching burstiness variable z_b^2 at each broker b, a more accurate subscriber reliability estimation algorithm can be done by the approaches presented in Section 4.1 through Section 4.4 but with one additional step between the step in Section 4.2 and the step in Section 4.3 in order to calculate link and broker flow burstiness. The broker flow burstiness will then be used to approximate the broker queuing delay. We omit the detail of the process of per-link flow burstiness calculation due to space limitation. The complete algorithm for flow burstiness calculation and improved queuing delay equation can be found in the technical report version of this paper. [18].

5 Evaluation Results

In this section, we present the evaluation results of our proposed analytical framework. The evaluation is done via simulation with realistic component parameters. Section 5.1 will describe the detail of simulation settings. Section 5.2 then presents the accuracy results of the reliability prediction algorithm proposed in Section 4. Section 5.3 presents the accuracy results of the improved G/G/1 reliability prediction algorithm presented in Section 4.5. Section 5.4 then presents the accuracy of the prediction algorithm with imprecise publisher content distribution.

5.1 Simulation Parameters

We validate our approach via simulation using ns-2 network simulator [3]. Unless explicitly specified, each simulation is run with the parameters presented in Table 3 The link delay between broker nodes are derived from Planetlab delay and bandwidth traces that were collected by Ripeanu et al [19, 22]. The event processing delay distribution at each broker is interpolated from broker performance report in publish/subscribe event matching algorithm works [11, 15]. Specifically, the average event matching time at each broker is linearly proportional to the number of subscriptions stored in that broker's routing table, with the increase rate roughly equal to 1 millisecond per 1 additional stored subscription. The processing time for each event at a broker is then drawn from either uniform distribution or exponential distribution with the computed average value. The publisher event distribution and subscription distribution are generated from Zipf-like distributions in order to reflect real-world events [1, 2].

Parameters	Value
#event attributes (k)	21
event lifetime	1 second
event content distribution	Zipf-like
#brokers	20
#topics	4
#publishers	8
#subscribers	100
#avg publishing rate	1 message / sec
Message size	64 bytes
Simulation Time	10000 seconds
#Runs	5

Table 3. Simulation Parameters

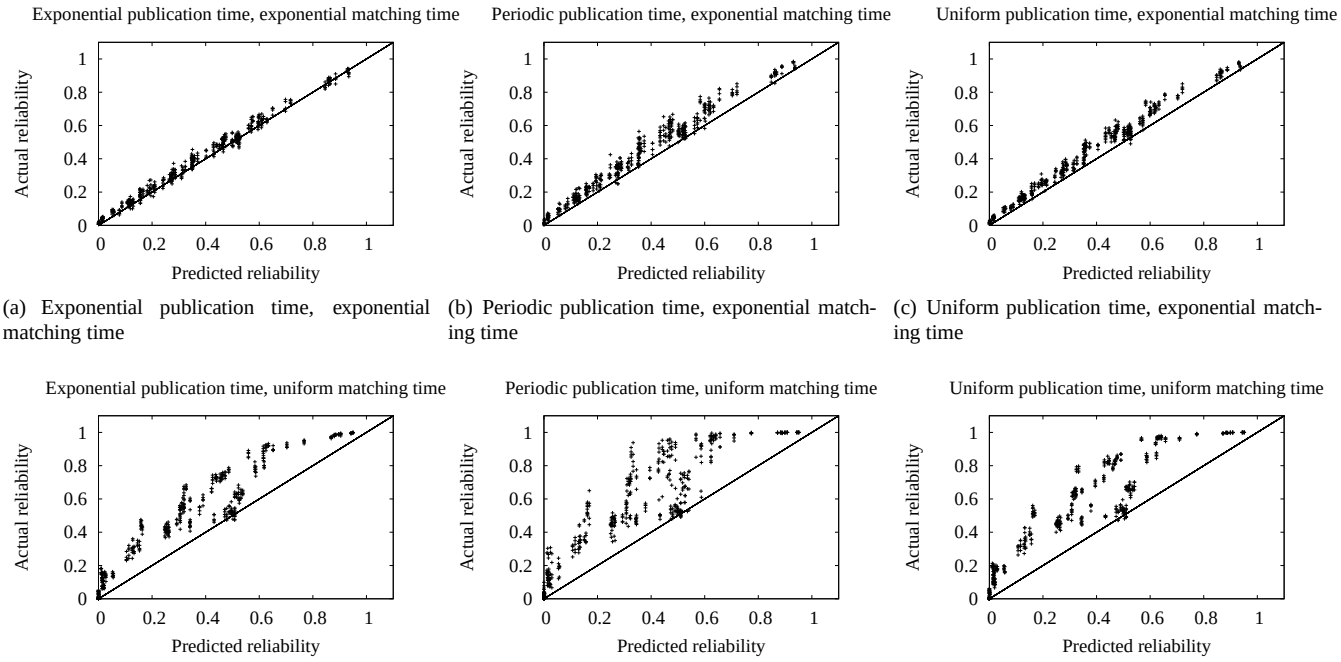
5.2 Prediction with M/M/1 Broker Model

In subscriber reliability prediction experiment, we vary publishers' publishing interval distribution between exponential, deterministic (i.e. periodic), and uniform publishing distributions. Also, we vary brokers' event matching distribution between exponential and uniform distributions.

Figure 4 presents the accuracy of the subscriber reliability estimation algorithm using M/M/1 broker model presented in Section 4 under different distributions of each publisher's publishing interval and each broker's event processing interval. The y-axis of each graph represents the values of actual subscriber real-time reliability while the x-axis of the graph represents the values of predicted real-time reliability. Each single point in each graph represents one subscriber in one run of simulation. As shown in the result, our algorithm can predict subscriber reliability values accurately in all scenarios. The prediction is most accurate in when publishing interval and event processing delay are both exponentially distributed (Figure 4(a)). While the results in non-exponential settings are less accurate, almost all predicted values are less than or equal to the actual reliability values. Hence, the prediction can still be used as reasonably tight upper bound of actual reliability.

5.3 Prediction with G/G/1 Broker Model

This section presents the accuracy of the subscriber reliability estimation using G/G/1 broker model. The exper-



(a) Exponential publication time, exponential matching time

(b) Periodic publication time, exponential matching time

(c) Uniform publication time, exponential matching time

(d) Exponential publication time, uniform matching time

(e) Periodic publication time, uniform matching time

(f) Uniform publication time, uniform matching time

Figure 4. M/M/1 model predicted subscriber reliability compared to actual reliability under different event traffic patterns

imental setting is the same as the setting in Section 5.2 except the estimation algorithm, which includes the flow burstiness calculation described in Section 4.5. Figure 5 shows the result of G/G/1 prediction. As seen from the result, the prediction accuracy with G/G/1 model is better than the one with M/M/1 model when the publication interval and matching interval are not exponentially distributed. When both publication interval and matching interval are exponentially distributed, both M/M/1 model and G/G/1 model produce the same result as explained in Section 4.5.

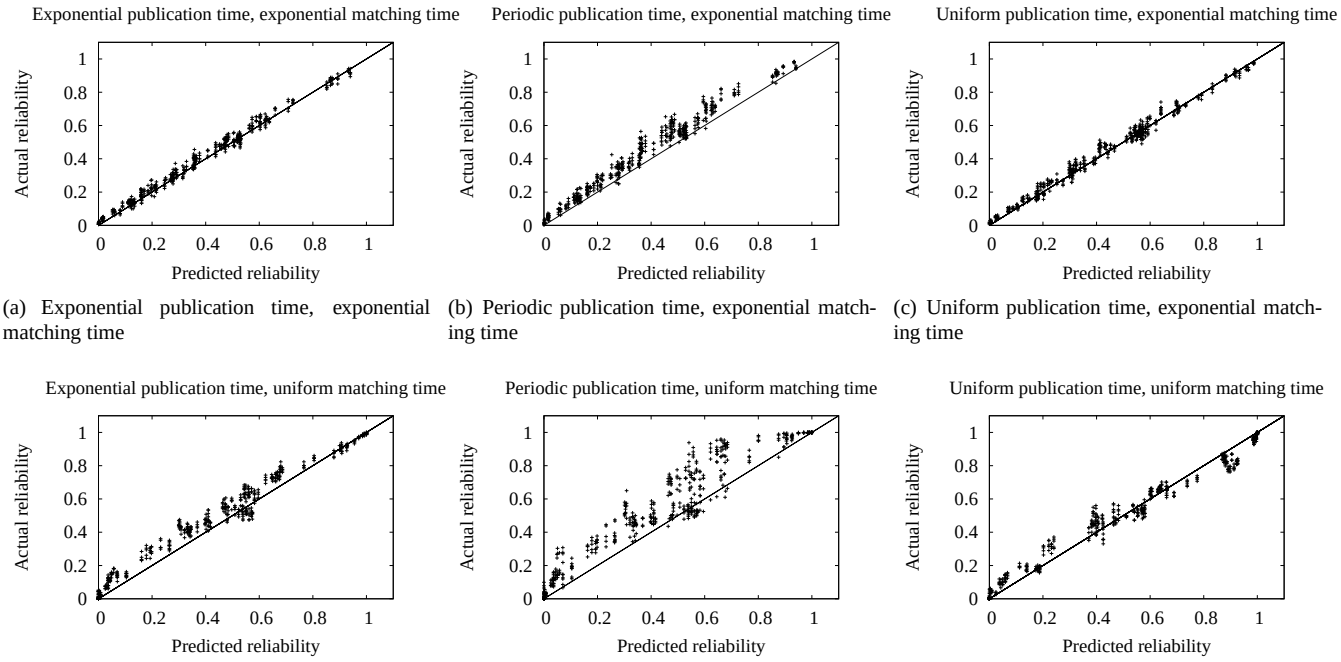
5.4 Effect of Imprecise Publisher Information

The reliability prediction results shown in Section 5.2 and Section 5.3 are based on the experiments with perfectly accurate publisher content-lifetime distributions. However, such assumption may not be true in practice as the approximation of publisher's characteristic may not be accurate. This section presents the accuracy of the subscriber reliability prediction algorithm with such imprecise publisher information. Specifically, we define *distribution skewness*, denoted by α, as the level of inaccuracy in the observed publisher content-lifetime distribution. Let $\tilde{C}_p(a,d)$ be the actual, hidden content-lifetime distribution of a publisher p, then the observed content-lifetime $C_p(a,d)$ of publisher p with skewness α is

$$C_p(a,d) = \frac{\tilde{C}_p(a,d)^\alpha}{\sum_{a \in V, d \geq 0} \tilde{C}_p(a,d)^\alpha}$$

That is, the observed probability that a publisher p will publish an event with content a and lifetime d will be equal to the actual probability of such event to the power of α, normalized by the total transformed weight. Hence, $\alpha = 1$ represents the scenario of perfectly precise publisher information.

Figure 6 presents the result of subscriber reliability prediction algorithm with the same parameter configuration as Section 5.2 and Section 5.3, but with different values of skewness (α). The results shown in Figure 6 are based on exponentially distributed publishers' publication interval and brokers' event processing delay, so both M/M/1 model and G/G/1 model produce the same results. It can be seen that the accuracy of the prediction algorithm slightly decreases when $\alpha > 1$, but significantly decreases when $\alpha < 1$. The conclusion is that $\alpha < 1$ reduces the difference of content popularity in Zipf-like distribution, and thus affects flow estimation accuracy more than when $\alpha > 1$. However, the overall prediction accuracy is acceptable.



(a) Exponential publication time, exponential matching time

(b) Periodic publication time, exponential matching time

(c) Uniform publication time, exponential matching time

(d) Exponential publication time, uniform matching time

(e) Periodic publication time, uniform matching time

(f) Uniform publication time, uniform matching time

Figure 5. G/G/1 model predicted subscriber reliability compared to actual reliability under different event traffic patterns

6 Related Works

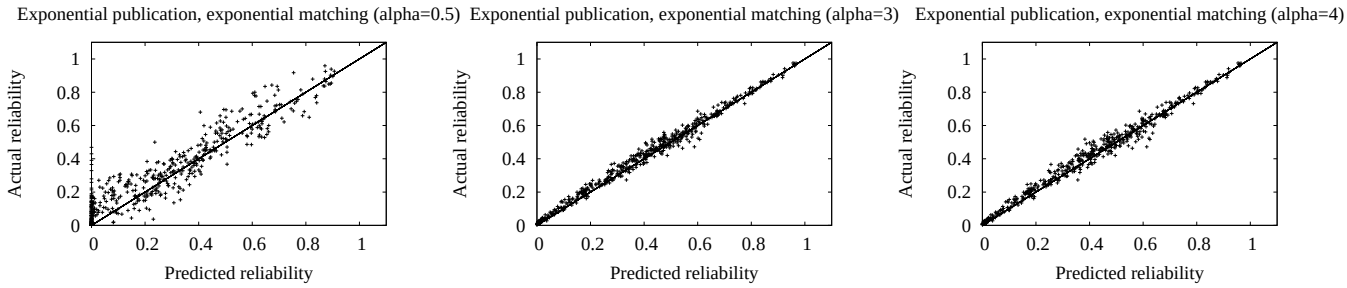
There have been significant efforts to model and analyze publish/subscribe systems along with their correctness properties and performance aspects. In his dissertation, Muhl [16] proposed a generic content-based publish/subscribe frameworks and a class of subscription/publication routing and matching algorithms with proof of correctness and performance analysis. Baldoni et al [4] also proposed correctness proof of publish/subscribe systems when subscription propagation delay is not negligible. However, both works assume reliable underlying networks and does not address event delivery timeliness aspect.

Liu and Jacobsen [15] addressed the uncertainty in terms of imprecise knowledge in subscriptions and events in content-based publish/subscribe systems. By expressing subscriptions and events in the form of fuzzy sets, the work proposes the publish/subscribe systems that allow approximate matching between subscriptions and events with vague attributes. The concept of publication uncertainty in their work can be considered equivalent to the concept of publisher content-lifetime probability distribution in our work. However, their work focus on the aspect of subscription uncertainty and correctness in event matching while our work focus on uncertainty in underlying networks, event delivery probability and timeliness.

Another work that resembles our work in modeling publish/subscribe system integration and timeliness is the work done by Kounev et al [13]. The work analyzes mean delivery delay of distributed event-based system with the use of rate calculation and queuing theory. While our work also uses the queuing theory to calculate delivery delay, our work presents the model that abstracts content-based events and subscriptions and allows fine-grained prediction of reliability and delay.

7 Conclusions

In this paper, we discussed the feasibility of performance assessment of distributed, content-based publish subscribe systems in terms of event delivery probability and end-to-end delivery delay. We proposed an analytical model that abstracts expressiveness nature of content-based publish/subscribe paradigm and uncertainty in underlying overlay networks. We then proposed the use of subscriber real-time reliability as a quality of service metric that combines delivery success rate and timeliness metrics. With the proposed model, we then presented the real-time reliability prediction algorithm for the given system configuration. Finally, the experimental results validated the algorithms' ac-



(a) α=0.5 (b) α=3 (c) α=4

Figure 6. Predicted reliability compared to actual reliability with inaccurate content distribution information

curacy and effectiveness.

References

[1] Google finance. http://www.google.com/finance.

[2] National climate data center: Online climate data directory. http://lwf.ncdc.noaa.gov/oa/climate/climatedata.html.

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

[4] R. Baldoni, R. Beraldi, S. Tucci Piergiovanni, and A. Virgillito. On the modeling of publish/subscribe communication systems: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(12):1471–1495, 2005.

[5] R. Baldoni, L. Querzoni, and A. Virgillito. Distributed event routing in publish/subscribe communication systems: a survey. Technical report, 2005.

[6] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a global event notification service. In *Proc. HOTOS '01*, page 87, Washington, DC, USA, 2001.

[7] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[8] G. Cugola and H.-A. Jacobsen. Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):25–33, 2002.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[10] O. M. Group. Corba Event Service. http://www.omg.org/technology/documents/formal/event_service.htm.

[11] X. Guo, J. Wei, and D. Han. Efficient Event Matching in Publish/Subscribe: Based on Routing Destination and Matching History. *Networking, Architecture, and Storage, International Conference on*, 0:129–136, 2008.

[12] X. Guo, H. Zhong, J. Wei, and D. Han. A new approach for overload management in content-based publish/subscribe. *Software Engineering Advances, International Conference on*, 0:32, 2007.

[13] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann. A methodology for performance modeling of distributed event-based systems. In *Proc ISORC '08*, pages 13–22, Washington, DC, USA, 2008. IEEE Computer Society.

[14] G. Li, S. Hou, and H.-A. Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In *Proc. ICDCS '05*, pages 447–457, Washington, DC, USA, 2005.

[15] H. Liu and H.-A. Jacobsen. Modeling Uncertainties in Publish/Subscribe Systems. In *Proc ICDE'04*, pages 510–521, March-2 April 2004.

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

[17] P. R. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *Proc. ICDCSW '02*, pages 611–618, Washington, DC, USA, 2002.

[18] T. Pongthawornkamol and K. Nahrstedt. Towards timeliness and reliability analysis of distributed content-based publish/subscribe systems over best-effort networks. Technical Report http://hdl.handle.net/2142/14415, Department of Computer Science, University of Illinois at Urbana-Champaign, November 2009.

[19] M. Ripeanu, I. T. Foster, A. Iamnitchi, and A. Rogers. A dynamically adaptive, unstructured multicast overlay. In *Service Management and Self-Organization in IP-based Networks*, 2005.

[20] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In *In Networked Group Communication*, pages 30–43, 2001.

[21] M. Ryll and S. Ratchev. Towards a publish / subscribe control architecture for precision assembly with the data distribution service. pages 359–369. 2008.

[22] D. Surendran. Visualizing connection bandwidths and delays in planetlab. http://people.cs.uchicago.edu/ dinoj/vis/planetlab/.

[23] W. Whitt. The Queueing Network Analyzer. *Bell System Technical Journal*, 62(9):2779–2815, November 1983.

[24] Y. Zhao, D. Sturman, and S. Bhola. Subscription propagation in highly-available publish/subscribe middleware. In *Proc. ACM Middleware '04*, pages 274–293, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[25] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. NOSSDAV '01*, pages 11–20, New York, NY, USA, 2001. ACM.