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

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

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 expressiveness nature of content-based publish/subscribe, 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. Furthermore, the paper leverages the proposed prediction algorithm to implements heuristic-based subscriber admission control algorithms to maximize system utility when the system cannot support all subscribers. The evaluation results yields good prediction accuracy and admission rates.

I. Introduction

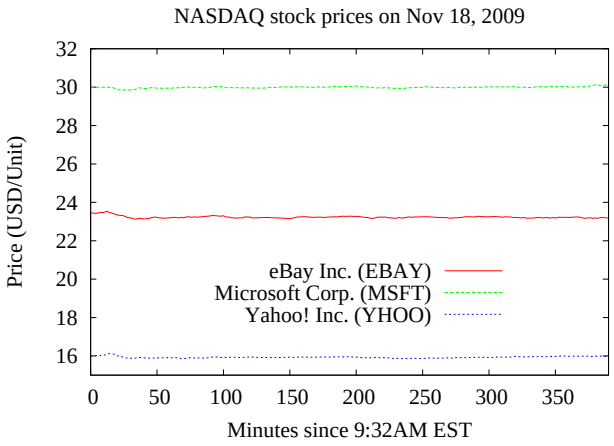
Over the past few years, publish/subscribe systems have recently become an emerging paradigm for large-scale information dissemination. The nature of publish/subscribe where the producers of the information (i.e. publishers) and the consumers of the information (i.e. subscribers) are interacted via intermediaries (i.e. brokers) allows both sides of the communication to be decoupled in space, time, and synchronization [1]. 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 [2]–[7], [7]–[11]. Based on commonly accepted taxonomy [1], [12]–[14], publish/subscribe systems can be categorized into *topic-based* publish/subscribe systems [5]–[7] and *content-based* publish/subscribe systems [2]–[4], [8]–[10]. 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 topic events it wants to receive. Content-based publish/subscribe systems give more flexibility to the subscribers at the cost of increasing processing complexity at brokers.

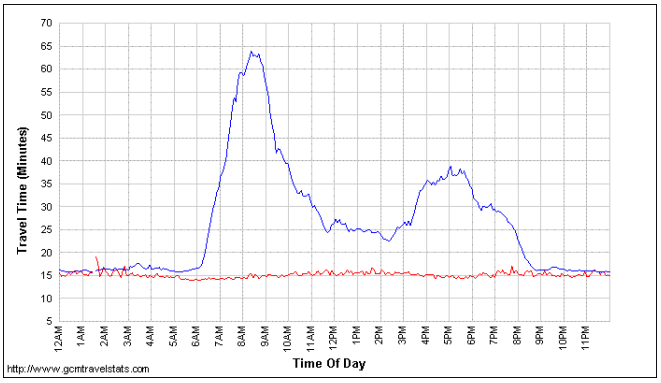
Besides the increasing complexity compared to topic-based publish/subscribe systems, another drawback of content-based publish/subscribe systems is less predictability. Since each subscriber has flexibility in choosing information it wants in fine-grained attribute level, it is also less trivial to determine event flow from each publisher to each subscriber. Hence, it is also less trivial to analyze the performance and correctness of content-based publish/subscribe 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, best-effort networks (i.e. the Internet) further decreases system determinism and predictability. Such uncertainty and complexity becomes a hindrance in applying content-based publish/subscribe systems to Internet-scale, time-sensitive applications such as stock market report [15], temperature/climate monitoring [16], and road traffic monitoring [17]. The need to solve such problem calls for a good analytical model that could accurately capture 1) applications' real-time requirements, 2) content-based publish/subscribe expressiveness, 3) uncertainty nature of underlying best-effort networks.

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, thus making it possible to model and predict behavior of the publish/subscribe system as a whole.

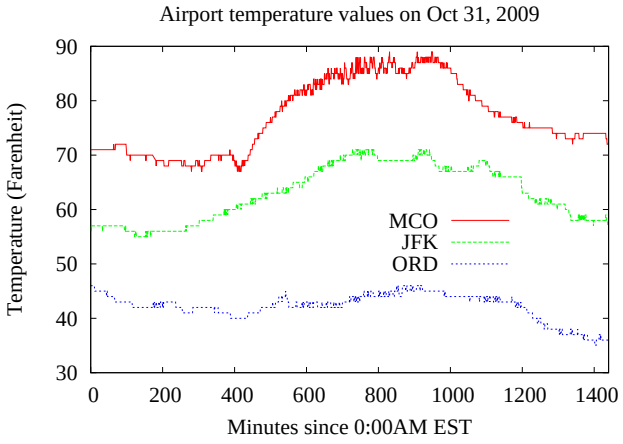
In this paper, we explore the possibility to use such imperfect information to predict event delivery delay and reliability in a distributed content-based publish/subscribe system by applying the techniques from probability theory and queuing theory. Specifically, our work has the following contributions. First, we propose a generic analytical model for *existing* distributed content-based publish/subscribe systems for the purpose of performance assessment. Second, we present the subscriber reliability prediction algorithm based on the proposed analytical model and the assumption of imperfect statistics information of each pub/sub component. Third, with the proposed prediction algorithm, we present a heuristic-based subscriber admission control protocol that provides QoS support to existing best-effort distributed content-based publish/subscribe systems. Fourth and finally, we present the simulation results of the proposed system under realistic parameters. The evaluation results yield good accuracy for the prediction algorithm and good admission rate for admission control algorithm, even when the statistics information of each component is inaccurate.



(a) NASDAQ stock prices (Source: Google Finance [15])



(b) Chicago highway daily traveling time (Source: GCM Travel [17])



(c) Airport temperature report (Source: NCDC [16])

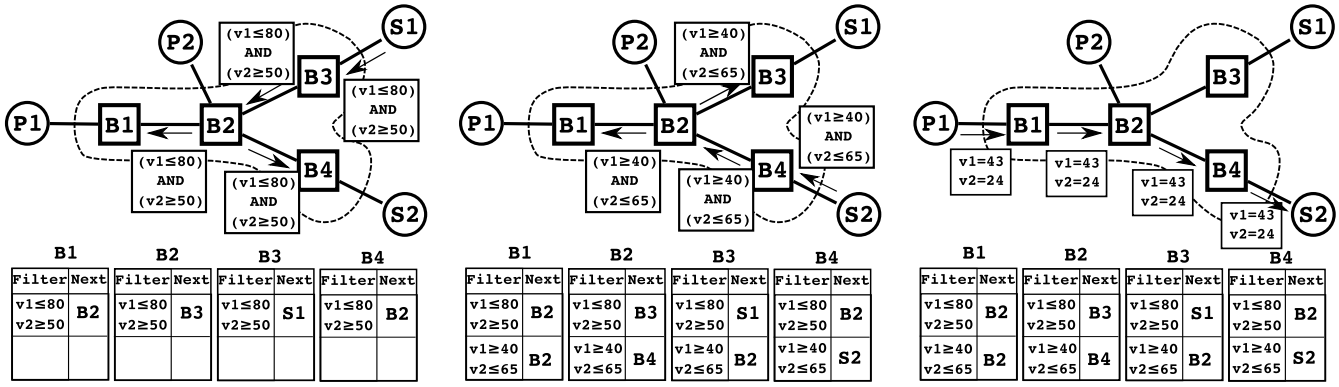
Fig. 1. Example of real-world event streams and their temporal locality

This paper is organized as follows. Section II discusses the model of distributed, content-based publish/subscribe system used in this work. Section III propose the mathematical model of real-time content-based publish/subscribe system along with the subscriber reliability prediction and admission control problem formulation. Section IV presents the analytical model to predict subscriber real-time reliability. Section V presents utility-based subscriber admission control algorithms for overloaded publish/subscribe systems. Section VI presents the evaluation results of the proposed systems. Section VII discusses related works in quality of service and modeling of real-time publish/subscribe systems. Finally, Section VIII suggests future directions of the work and concludes the paper.

II. 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 in the described model.

A. Soft Real-time Distributed Publish/Subscribe



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

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

In this work, we assume generic acyclic publish/subscribe tree model commonly adopted in existing works [2]–[4], [8] as follows. A publish/subscribe system consists of a group of *subscribers* (information consumers) and *publishers* (information providers) connected via a network of *brokers* (information intermediaries). We assume *acyclic* broker 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 called *home broker*. 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 system 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 content 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 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 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 [2]–[4], [8].

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

B. Publish/Subscribe Quality of Service

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.

[1]In Figure 2(a), each predicate filter is in conjunctive form consisting of per-attribute min-max clauses. However, our analytical model supports all possible forms of filter as long as the filter can be expressed a a subset of the content space.

C. 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, as will be discussed in Section VIII.

In the next section, we will present the formal definition of each component described in this section and the definition of subscriber reliability estimation problem.

III. Analytical Model Framework

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

A. Publish/Subscribe Entity Model

1) Events: Let $\mathbb{E}$ be the set of all events published in the system. An event $e \in \mathbb{E}$ is defined as a 3-tuple

$$e = (id_e, a_e, d_e)$$

, 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 k-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 $k-1$ 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 and D 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.

2) Subscribers: 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 event set E with respect to subscriber s as

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

3) Publishers: 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), $I_p(t)$ is the inter-event publishing time distribution, and id_p is the publisher's identifier. Thus

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

and

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

4) Brokers: Each broker in the system has a single event queue that is used to store and match event in 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 $M_b(t)$ 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.

B. Network-level Entity Model

We model the publish/subscribe network as a directed acyclic graph $G(B \cup P \cup S, L)$, where $B \cup P \cup S$ 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 among nodes in the system. Each link can be categorized into either *publisher-broker* link (direct link from a publisher to a broker), *broker-broker* link (direct link from a broker to another broker), or *broker-subscriber* link (direct link from a broker to a subscriber). Each link l has reliability r_l and link delay distribution $D_l(t)$. We define $out(l)$ and $in(l)$ as the source and the sink of link l respectively.

C. Quality of Service Model

1) Subscriber Reliability: Let $\mathbb{E}_s$ be the set of all events that are published during the period that a subscriber s is in the system. Hence, $F_s(\mathbb{E}_s)$ is the set of all events of s's interest during its stay 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 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. We believe the defined reliability metric is good enough to represent quality of service, as it combines both reliability and delay, which are two important metrics in soft real-time publish/subscribe applications. 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 defined in Section III-D1.

2) Publish/Subscribe Utility Model: Let each subscriber s has its own real-time reliability requirement R_s^*, a subscriber s is said to have its requirement satisfied if $R_s \leq R_s^*$. We define the set of *satisfied subscribers* with respect to the publish/subscribe network G, denoted by $S'(G)$, as the set of subscribers in G that have their reliability requirements satisfied (i.e. $S'(B \cup P \cup S, L) = \{s \in S : R_s \leq R_s^*\}$). We define the utility of the publish/subscribe network $G = (B \cup P \cup S, L)$, denoted by $U(G)$ as the number of satisfied subscribers. That is $U(G = (B \cup P \cup S, L)) = |S'(G)|$.

With nature of proposed utility function $U(G)$, it is more beneficial not to admit the whole subscriber set S into the system if we know in advance that some subscribers will not meet their requirements, since those unsatisfied subscribers will only waste system resources without adding any benefit to the system. Instead, a subscriber s should be admitted to the system only when it is likely to have its requirement satisfied.

D. Problem Definition

Based on the previously defined model, this section formulates the two problems to be solved by this work, the subscriber reliability estimation problem and subscriber admission control problem.

1) Subscriber Reliability Estimation: In Section III-C1, we formally define subscriber reliability and utility as quality of service indicator for each subscriber in the system. However, we would like to predict reliability R_s for each subscriber s in advance before actually running the system. This leads to the subscriber real-time reliability estimation problem.

Definition *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 $s \in S$.

Based on the proposed analytical model, this work presents a subscriber reliability estimation algorithm in Section IV.

2) Subscriber Admission Control: As mentioned in Section III-C2, admitting all subscribers in the systems may result in bad system utility. Thus, the system should pick only a subset of subscribers that will maximize the system utility. Hence, we define subscriber admission control problem as follows.

Definition *Subscriber Admission Control Problem:* Given a publish/subscribe network $G = (B \cup P \cup S, L)$, find the largest subscriber subset $S^* \subseteq S$ that maximize the utility of the system (i.e. $S^* = \arg\max_{S' \subseteq S} U(B \cup P \cup S', L - (B \times (S - S')))$).

It is trivial that the subscriber admission control problem is NP-Hard problem, as the problem can be specialized to other NP-hard optimization problems such as multicast admission control or multi-commodity flow problems. However, this work discusses a set of greedy, heuristic-based algorithms to solve the subscriber admission control problem in Section V.

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
R^*_s	subscriber s's requested real-time reliability
$U(R_s)$	subscriber s's utility
$U(G)$	publish/subscribe network G's utility
S^*	subscriber subset that maximize system utility
$p \in P$	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 I
MODEL VARIABLES' NOTATION

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 incoming 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-remaintime distribution of events through link l

TABLE II
ANALYSIS VARIABLES' NOTATION

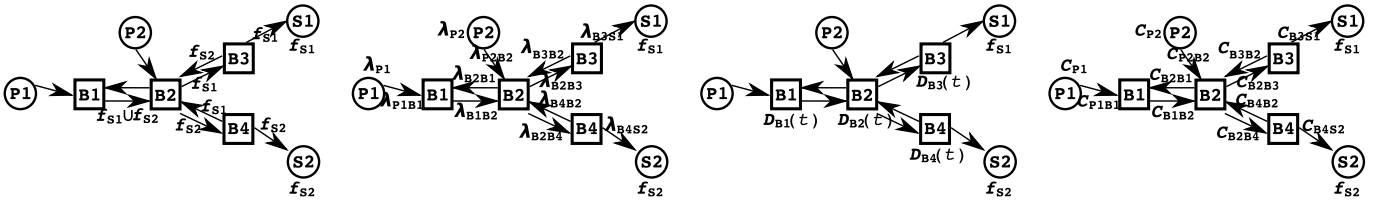
IV. SUBSCRIBER REAL-TIME RELIABILITY ESTIMATION

A. Estimation Algorithm

In this section, we present how to calculate the estimated real-time reliability R'_s at each subscriber s. To do so, it is necessary to estimate the end-to-end delivery delay and path reliability distributions of all s's matching events when they arrive at s. Hence, we introduce another set of variables in Table II for the purpose 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 subscriptions, calculating per-link event flow rate, calculating broker queuing/processing delay, and calculating per-link content-lifetime distribution.

1) Subscription 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 II-A. 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 b's outgoing link l, the subscription will be propagated to all other incoming links of b. At the same time, the subscription filter f will be included into l's *filter set*, denoted by f_l. That is, for each filter f that propagates via 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 l.



(a) Propagating subscription (reversed direction of arrows)

(b) Calculating link event rate

(c) Calculating broker total delay distribution

(d) Calculating per-link content-remaintime distribution

Fig. 3. The steps for subscriber reliability estimation

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 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-only 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 considered *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. 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.

3) Broker Queuing/Processing 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

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

and

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

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, denoted by $D_b(t)$ as

$$D_b(t + q_b) = M_b(t)$$

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 VI 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 [18]. One approach is to model a broker as a G/G/1 queue, which is presented in Section IV-B.

4) Per-link Content-remaintime 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-remaintime 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 IV-A2, except that both content and lifetime are now considered in the calculation. Specifically, for each publisher-broker link l, the content-remaintime distribution $C_l(a, d)$ is calculated as

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

, 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 IV-A2, 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-remaintime distribution as follows.

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

, where

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

, where λ is calculated from Equation (4).

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

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

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 $G = (B \cup P \cup S, L)$.

B. Improved Reliability Estimation with G/G/1 Queue Model

So far, the load estimation at each broker presented in Section IV-A 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 a modification to the 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

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

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,

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

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 SectionIV-A but with one additional step between the step in Section IV-A2 and the step in Section IV-A3 in order to calculate link and broker flow burstiness. Hence, the subscriber reliability estimation process with G/G/1 broker model consists of five steps : propagating subscriptions, calculating per-link event flow rate, calculate per-link event flow burstiness, calculating broker queuing/processing delay, and calculating per-link content-lifetime distribution. The details of all steps are the same as the ones described in Section IV-A1 through Section IV-A4 except the new step to calculate per-link event flow burstiness and the modified step to calculate broker queuing delay, which are described as follows.

1) Per-link Event Flow Burstiness Calculation: The process of per-link event flow burstiness calculation starts after the process of per-link event flow rate calculation (Section IV-A2) is done. After the flow rate calculation process, the per-link event flow rate λ_l and content distribution $C'_l(a)$ is known for each link l. Also, the per-publisher event flow burstiness z_p^2 for each publisher p and per-broker event matching burstiness z_b^2 for each publisher b are known via equation (12) and (13) respectively. The per-link event flow burstiness calculation process aims to calculate per-link event flow burstiness z_l^2 for each link l. The techniques used in the calculation are adopted from traditional queuing network theory [18].

The process starts by calculating z_l^2 for each *publisher-broker* link l using the asymptotic method [18] as follows.

$$\forall l \in L : out(l) \in P, z_l^2 = r_l.z_p^2 + 1 - r_l \tag{14}$$

, where $p = out(l)$

To calculate per-link event flow burstiness for *broker-broker* and *broker-subscriber* links, a set of linear equations must be solved according to the following set of rules.

Incoming Flow Superposition: we define *per-broker incoming flow burstiness*, denoted by z_{bi}^2 for each broker b, as the burstiness of the total event flow coming from all b's incoming links. Using the superposition rule and the asymptotic method, the per-broker incoming flow burstiness is the convex combination of each per-link flow burstiness as follows.

$$\forall b \in B, z_{bi}^2 = \sum_{l \in L: in(l) = b} \left(\frac{\lambda_l}{\lambda_b} . z_l^2 \right) \tag{15}$$

, where λ_b is the total incoming event flow rate at broker b calculated from Equation (6).

Equation (15) takes place at each broker $b \in B$ in the system. Hence, there are $|B|$ incoming flow equations.

Broker Incoming-Outgoing Flow Transformation: we define *per-broker outgoing flow burstiness*, denoted by z_{bo}^2 for each broker b, as the burstiness of the total event flow going out from broker b to all b's outgoing links. Using Marshall's formula [18], the per-broker outgoing flow burstiness z_{bo}^2 is a function of total incoming flow burstiness z_{bi}^2, total incoming flow rate λ_b (Equation (6), broker average event matching rate μ_b (Equation (7)), broker event matching burstiness z_b^2 (Equation (13)) as follows.

$$\forall b \in B, z_{bo}^2 = (\rho_b^2 . z_b^2 + (1 - \rho_b^2) . z_{bi}^2 \tag{16}$$

, where $\rho_b = \frac{\lambda_b}{\mu_b}$

Since Equation 16 takes place at each broker $b \in B$, there are $|B|$ incoming-outgoing flow equations.

Broker Outgoing Flow Splitting: after a broker fetches the incoming event from the head of the queue, it routes the event to each outgoing link with the subscription that matches the event. Hence, the per-link event flow burstiness of each outgoing link z_l^2 is a function of its source broker's incoming traffic rate λ_b (From Equation (6)) and its own traffic rate λ_l (From Equation (3)) as follows.

$$\forall l \in L : out(l) \in B, z_l^2 = \frac{\lambda_l}{\lambda_b} . z_{bo}^2 + 1 - \frac{\lambda_l}{\lambda_b} \tag{17}$$

, where $b = out(l)$

From the three equations (Equation (15), (16), and (17)), there are three forms of unknown variables (z_{bi}^2,z_{bo}^2, and z_l^2). All other variables are known from previous calculations. Since each unknown variable z_l^2 can be written in a linear form of some variable z_{bo}^2 using Equation (17) and each unknown variable z_{bo}^2 can be written in a linear form of some variable z_{bi}^2 using Equation (16), there are $|B|$ unknown variables left, which are in the form of z_{bi}^2. Also, there are $|B|$ equations left (Equation (15)). Since there are $|B|$ unknown variables left with $|B|$ linear equations, each variable z_{bi}^2 for each broker $b \in B$ can be solved by using standard matrix operations. Once variables in the form of z_{bi}^2 are solved, other unknown variables in the forms of z_{bo}^2 and z_l^2 are also solved using Equation (16) and (17). However, only variables in the form of z_{bi}^2 are needed in the next step to calculate the queuing delay at each broker.

2) Improved Broker Queuing/Processing Delay Calculation: After the total incoming event flow burstiness z_{bi}^2 is calculated at each broker $b \in B$, a more accurate estimation of the average queuing delay q_b for each broker $b \in B$ is then a function of total incoming flow burstiness z_{bi}^2, total incoming flow rate λ_b (Equation (6), broker average event matching rate μ_b (Equation (7)), broker event matching burstiness z_b^2 (Equation (13)) as follows.

$$q_b = \frac{\rho_b . (z_{bi}^2 + z_b^2) . g(\rho_b, z_{bi}^2, z_b^2)}{2 . \mu_b . (1 - \rho_b)} \tag{18}$$

where $\rho_b = \frac{\lambda_b}{\mu_b}$ and

$$g(\rho_b, z_{bi}^2, z_b^2) = \begin{cases} \exp(-\frac{2(1-\rho_b).(1-z_{bi}^2)^2}{3\rho_b.(z_{bi}^2+z_b^2)}) & \text{if } z_{bi}^2 < 1 \\ 1 & \text{if } z_{bi}^2 \geq 1 \end{cases}$$

Thus, we replace Equation (5) with new Equation (18) to calculate the average broker queuing delay, which is then used to calculate content-remain time distribution and finally the subscriber reliability estimation as stated in Section IV-A4. Note that when the incoming event flow rate and the event matching rate of a broker are exponentially distributed (i.e. $z_{bi}^2 = 1$ and $z_b^2 = 1$), then Equation (18) is reduced to Equation (5).

The proposed G/G/1 model reliability estimation yields better estimation accuracy when compared to the M/M/1 model presented in Section IV-A. However, the G/G/1 estimation requires solving $|B|$ linear equations and thus makes it hard to do in decentralized manner. On the other hand, all calculations in M/M/1 estimation can be done locally at each broker with few messages exchanged among neighbors, making it possible to calculate in decentralized manner. The estimation result from either M/M/1 estimation or G/G/1 estimation can then be used for subscriber admission control to maximize system utility. In the next Section, we will present a heuristic-based admission control based on the presented subscriber reliability estimation to maximize publish/subscribe system utility.

V. Utility-based Subscriber Admission Control

In this Section, we propose the heuristic-based algorithm to solve the subscriber admission problem. That is, given a publish/subscribe network $G = (B \cup P \cup S, L)$, find the subset of subscriber set S, denoted by S^*, that will maximize system utility. In the other words, $S^* = \arg\max_{S' \subseteq S} U(G')$ where $G' = (B \cup P \cup S', L - (B \times (S - S')))$. This algorithm is run in a centralized fashion at a control center node, which periodically collects monitoring status from each publisher/broker entities in the network and uses such collected status to run the subscriber reliability estimation and admission control every time a new subscriber joins the system.

As mentioned, the subscriber admission problem is an NP-hard problem with respect to the number of subscribers ($|S|$). However, since we can estimate the system utility $U(G)$ for any publish/subscribe network G based on the approach presented in Section IV, we now then propose the heuristic-based, greedy algorithm framework, denoted by $A^*(G)$ to for the subscriber admission control problem (i.e. $A^*(G)$ approximates S^* for $G = (B \cup P \cup S, L)$).

Algorithm 1 Function $A^*(G = (B \cup P \cup S, L))$

```
S'' ⇐ S
S* ⇐ ∅
U* ⇐ 0
while S'' ≠ ∅ do
    s ⇐ arg max_{s' ∈ S''} φ(s')
    G' = (B ∪ P ∪ S* ∪ {s}, L − (B × (S − S* − {s})))
    if U(G') > U* then
        S* ⇐ S* ∪ {s}
        U* ⇐ U(G')
    end if
    S'' ⇐ S'' − {s}
end while
return S*
```

A. Admission Control Algorithms

Algorithm (1) presents the detail of the greedy, heuristic-based subscriber admission control algorithm A^* to approximate the maximum-utility subscriber set S^*. The basic concept of the algorithm A^* is to initially set the admitted subscriber set S^* to empty set, and then grows the set S^* progressively by including each subscriber $s \in S$ only when the addition of s can increase the system utility. The system utility can be approximated based on the analytical framework described in Section IV. The order of subscribers in the addition process is obtained on the priority function $\phi(s)$, which gives a priority value to each subscriber s. Since each subscriber is considered only once in the addition process, the priority function $\phi(s)$ must be chosen carefully to achieve near-optimum maximum-utility subscriber set.

In this work, we pick a set of heuristic subscriber priority functions $\phi(s)$ to be used with the maximum-utility subscriber admission algorithm framework A^* as follows.

Random Priority (random): The priority value of each subscriber is determined randomly based on its identification number (i.e. $\phi(s) = id_s$).
Requirement Priority (hi-req-first): The priority value of each subscriber is equal to the reliability requirement of itself (i.e. $\phi(s) = R_s^*$). Hence, the subscriber with higher reliability requirement will be considered before the one with lower reliability requirement in this priority function.
Inversed Requirement Priority (low-req-first): The priority value of each subscriber is equal to the inverse of the reliability requirement of itself (i.e. $\phi(s) = 1 - R_s^*$). This scheme is the opposite of the requirement priority scheme, as the subscriber with lower reliability will be considered first in this scheme.

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 III
SIMULATION PARAMETERS

Additional Content Priority (overlap-first): In this scheme, the first $\log_2|S|$ subscribers will have random priority (i.e. $\phi(s) = id_s$). However, after $\log_2|S|$ subscribers, the subscriber priority s will be calculated as the inverse of the size of additional filter space incurred by adding such subscriber (i.e. $\phi(s) = \frac{1}{|f_s - f_{s^*}|}$ where $f_{s^*} = \bigcup_{s \in s^*} f_s$).

In Section VI-D, we will evaluate and compare the effectiveness of each subscriber priority function to approximate the maximum-utility subscriber set in the publish/subscribe system.

VI. 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 VI-A will describe the detail of simulation settings. Section VI-B then presents the results regarding the accuracy of the M/M/1 reliability prediction algorithm and the improved G/G/1 reliability prediction algorithm proposed in Section IV-A and Section IV-B respectively. Section VI-D then discusses the efficiency of the subscriber admission control algorithm presented in Section V.

A. Simulation Parameters

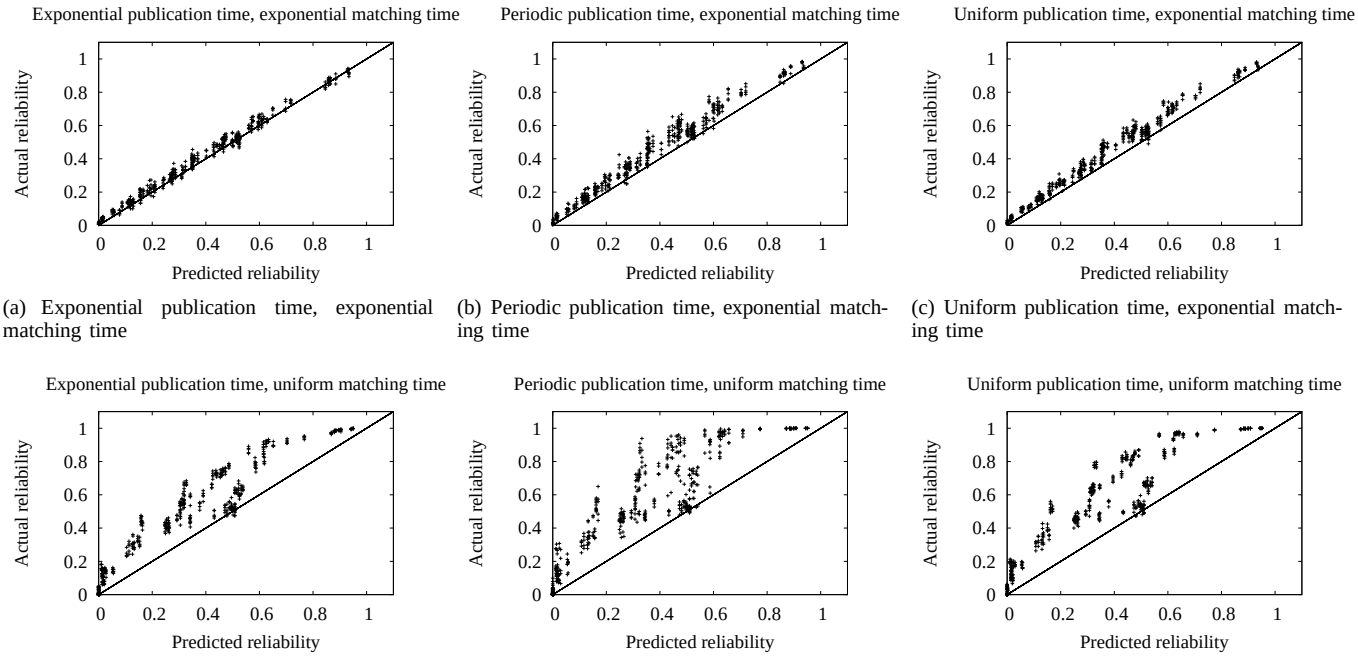
We validate our approach via simulation using ns-2 network simulator [19]. Unless explicitly specified, each simulation is run with the parameters presented in Table III The link delay between broker nodes are derived from Planetlab delay and bandwidth traces that were collected by Ripeanu et al [20], [21]. The event processing delay distribution is approximated and simplified from recent related works in event matching algorithms [22], [23]. 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.

B. Reliability Prediction

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 processing distribution between exponential and uniform matching distributions.

1) Prediction with M/M/1 Broker Model: Figure 4 presents the accuracy of the subscriber reliability estimation algorithm using M/M/1 broker model presented in Section IV-A 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 other 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.

2) Prediction with G/G/1 Broker Model: This section presents the accuracy of the subscriber reliability estimation using G/G/1 broker model. The experimental setting is the same as the setting in Section VI-B1 except the estimation algorithm, which includes the flow burstiness calculation described in Section IV-B. 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 IV-B.



(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

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

C. The Effect of Imprecise Publisher Information

The reliability prediction results shown in Section VI-B 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

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

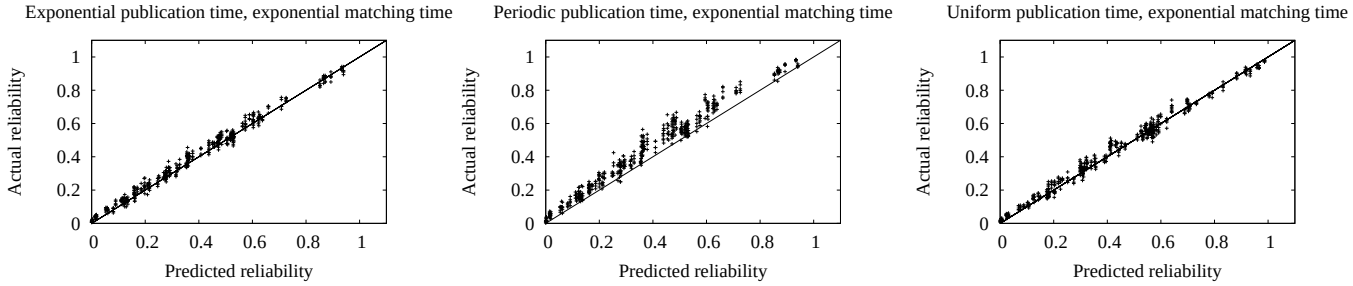
That is, the observed probability that a publisher p will publish an event with content a and lifetime d will be equal to the actual probability of such event to the power of α, normalized by the total transformed weight. Hence, $\alpha = 1$ represents the scenario of perfectly precise publisher information.

Figure 6 presents the result of subscriber reliability prediction algorithm with the same parameter configuration as Section VI-B, 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.

D. Subscriber Admission Control

We evaluate the heuristic-based admission control algorithms discussed in Section V in a smaller-scale setting due to time constraint in exhaustively exploring all possible subscriber sets to find the optimal solution. The publish/subscribe system in the setting consists of 4 brokers, 8 publishers, and 10 requested subscribers. The event publishing interval and event processing time are exponentially distributed, resulting in no difference between results from M/M/1 model and G/G/1 model.

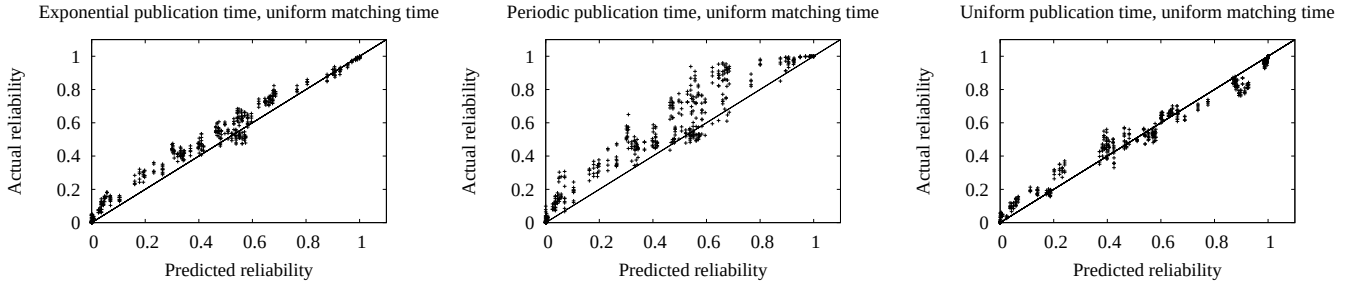
Figure 7 shows the fraction of subscribers that have their requirements satisfied. As shown from the figure, the publish/subscribe system without admission control performs the worst, since all subscribers are admitted to the system and contend for resources. On the other hand, the proposed heuristic–based algorithms give satisfaction rates that are closed to the optimal subscriber selection, yielding the effectiveness of the algorithm. Each algorithms perform closed to each other without clear extinction, although the low-req-first heuristic perform slightly better than others as the load increases.



(a) Exponential publication time, exponential matching time

(b) Periodic publication time, exponential matching time

(c) Uniform publication time, exponential matching time

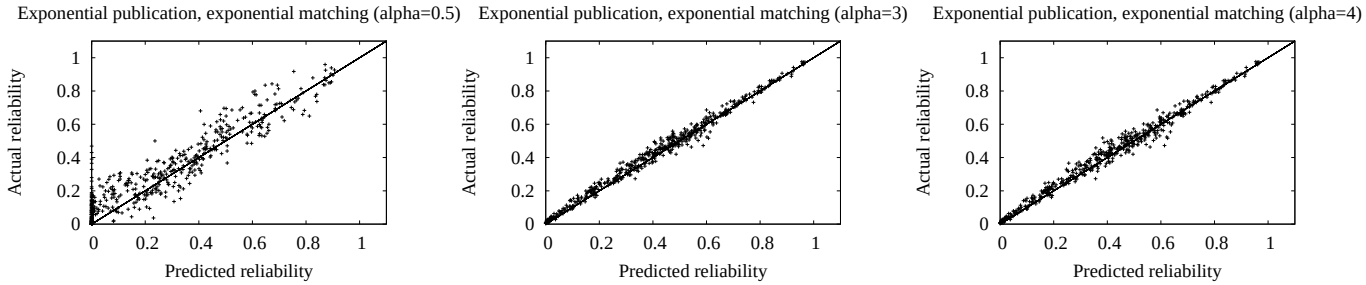


(d) Exponential publication time, uniform matching time

(e) Periodic publication time, uniform matching time

(f) Uniform publication time, uniform matching time

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



(a) α=0.5

(b) α=3

(c) α=4

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

VII. 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 [24] 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 [25] 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. He et al [26] proposed a publish/subscribe model checker based on probabilistic timed automata. However, the computational overhead associated with the automata due to state explosion may limit the usage of such approach to only small-sized problems.

Liu and Jacobsen [22] 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 [27]. The work analyzes mean delivery delay of distributed event-based system with the use of rate calculation

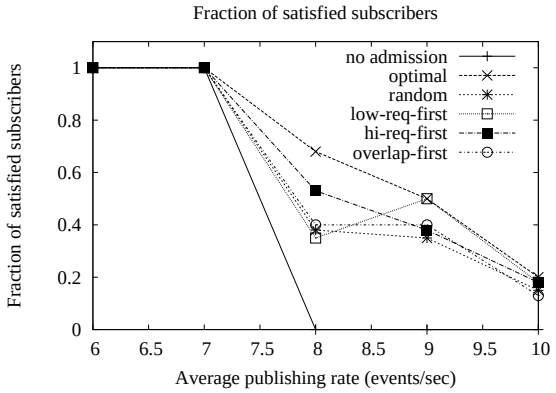


Fig. 7. Fraction of satisfied subscribers with different admission control algorithms

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. We also propose a heuristic-based admission control on top of such model.

VIII. 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. Moreover, a set of subscriber admission control algorithms based on the prediction algorithm were also proposed. Finally, the experimental results validated the algorithms' accuracy and effectiveness. Our future directions of this work include decentralized subscriber reliability estimation/admission control, mobile subscriber admission control, and admission control on cyclic-overlay content-based publish/subscribe systems.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

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

[4] G. Cugola and H.-A. Jacobsen, "Using publish/subscribe middleware for mobile systems," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 4, pp. 25–33, 2002.

[5] 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*, 2001, pp. 30–43.

[6] L. F. Cabrera, M. B. Jones, and M. Theimer, "Herald: Achieving a global event notification service," in *Proc. HOTOS '01*. Washington, DC, USA: IEEE Computer Society, 2001, p. 87.

[7] 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*. New York, NY, USA: ACM, 2001, pp. 11–20.

[8] P. R. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *Proc. ICDCSW '02*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 611–618.

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

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

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

[12] ——, "Distributed event routing in publish/subscribe communication systems: a survey," Tech. Rep., 2005.

[13] R. S. S. Filho and D. F. Redmiles, "A survey of versatility for publish/subscribe infrastructures," University of California, Irvine, Tech. Rep. UCI-ISR-05-8, 2005.

[14] Y. Liu and B. Plale, "Survey of publish subscribe event systems," Indiana University, Tech. Rep. TR574, May 2003.

[15] "Google finance," http://www.google.com/finance.

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

[17] "Gary-chicago-milwaukee corridor transportation information," http://www.gcmtravel.com/.

[18] W. Whitt, "The Queueing Network Analyzer," *Bell System Technical Journal*, vol. 62, no. 9, pp. 2779–2815, November 1983.

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

[20] 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.

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

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

[23] 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*, vol. 0, pp. 129–136, 2008.

[24] G. Muhl, "Large-scale content-based publish/subscribe systems," Ph.D. dissertation, University of Technology Darmstadt, 2002.

[25] R. Baldoni, R. Beraldi, S. Tucci Piergiovanni, and A. Virgillito, "On the modeling of publish/subscribe communication systems: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 17, no. 12, pp. 1471–1495, 2005.
[26] F. He, L. Baresi, C. Ghezzi, and P. Spoletini, "Formal analysis of publish-subscribe systems by probabilistic timed automata," in *27th IFIP WG 6.1 International Conference on Formal Techniques forNetworked and Distributed Systems (FORTE 2007), Tallinn, Estonia, June 27-29*, ser. Lecture Notes in Computer Science, vol. 4574. Springer, 2007, pp. 247–262.
[27] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann, "A methodology for performance modeling of distributed event-based systems," in *Proc ISORC '08*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 13–22.