Università della Svizzera italiana

Faculty of Informatics

USI Technical Report Series in Informatics

Probabilistic FIFO Ordering In Publish/Subscribe Networks

Amirhossein Malekpour, Antonio Carzaniga, Giovanni Toffetti Carughi, Fernando Pedone

Faculty of Informatics, University of Lugano, Switzerland

Abstract

In a best-effort publish/subscribe network, publications may be delivered out of order (e.g., violating FIFO order). We contend that the primary cause of such ordering violations is the parallel matching and forwarding process employed by brokers to achieve high throughput. In this paper, we present an end-to-end method to improve event ordering. The method involves the receiver (and minimally the sender) and otherwise uses the broker network as a black box. The idea is to analyze the dynamics of the network, and in particular to measure the delivery delay and its variation, which is directly related to out-of-order delivery. With these measures, receivers can determine a near-optimal latch time to defer message delivery upon the detection of a hole in the message sequence number. We evaluate the performance of this ordering scheme empirically in terms of the reduction in out-of-order deliveries, the delay imposed by the latch time, and its self-adjustment with variable network conditions and input loads.

Report Info

Published
April 2011

Number
USI-INF-TR-2011-2

Institution
Faculty of Informatics
Università della Svizzera italiana
Lugano, Switzerland

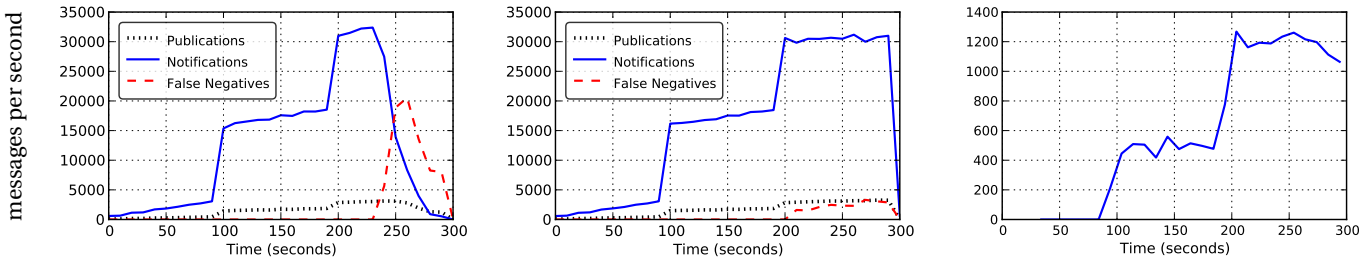
Online Access
www.inf.usi.ch/techreports

1 Introduction

In the content-based publish/subscribe communication model, or simply content-based communication, the addressing of messages is implicit and controlled by the receivers. Receivers express their interests through subscriptions that state conditions on the content of messages, while senders simply publish messages without any set address. Each message is then delivered to all receivers whose interests match the content of the message. Content-based communication has a variety of applications, such as system monitoring and management, information dissemination, resource discovery, stream processing, and distributed simulation.

In spite of substantial efforts to devise and implement robust and efficient content-based publish/subscribe systems [1, 2, 3, 4], a few proposals have considered the issue of message ordering and its most basic FIFO form: two messages published by the same sender must be delivered to a receiver in the same order they were published. FIFO ordering is typically implemented using sequence numbers set on the sender side to reflect the sending order, and checked on the receiver side to enforce the same order for delivery [5]. When the network delivers a message with a higher-than-expected sequence number, the receiver must decide whether to wait for the missing message or to proceed by delivering the message it has received. However, because of the implicit addressing induced by the content-based model, the receiver does not know whether the hole in the sequence is due to a message that was delayed along the delivery path, or to a message that does not match the receiver's interests.

In this paper we present a probabilistic method to achieve FIFO ordering in content-based communication. We illustrate this method using B-DRP, a high-throughput content-based network [6]. B-DRP implements is "best-effort" with respect to ordering and reliability, does not store messages at intermediate brokers, and does not use acknowledgments to confirm delivery. Also, B-DRP's design is intended to achieve high deliv-



(a) ActiveMQ: Throughput (b) B-DRP: Throughput (c) B-DRP: FIFO Violations

Figure 1: Throughput and FIFO violations of ActiveMQ and B-DRP in an 8-broker network.

ery rates thanks to an efficient routing scheme as well as highly parallelized matching and processing within brokers. Thus, B-DRP is arguably an ideal testbed to experiment with message ordering. Yet, the method is generic, as it applies to end-points (publishers and subscribers) and treats the whole network as a black box.

Intuitively, FIFO violations are caused by short-term variations of the end-to-end delay of messages, which may occur in the presence of different delivery paths or if the forwarding process is parallelized and therefore does not itself maintain FIFO ordering. At a high-level, our approach is to measure the delay variations, and then compensate for their effect.

To understand and measure delay variations, we study the dynamics of an actual content-based network. We show that the end-to-end delay of messages along a specific path follows a *hypoexponential* distribution. We also develop a way to measure the parameters of this distribution dynamically, and therefore a method to calculate the probability of a FIFO violation upon the observation of a hole in the sequence numbers. We also use the same model and technique to estimate the necessary latch time (i.e., deferring the delivery of messages to the application) to reduce the probability of a FIFO violation. We then enhance the receiver's decision algorithm with a method to estimate the relevance of missing messages, to prevent the unnecessary holding of a message when none of the missing messages matches any local interest.

We have fully implemented and experimentally evaluated our technique. Extensive experiments with networks of up to 46 brokers and 2500 clients reveal that our model reflects the dynamics of the network in various working conditions, and is able to avoid more than 95% of FIFO violations while keeping the extra delay caused by latching to a minimum.

In Section 2 we begin by motivating this work and overviewing the problem and our proposed solution. We then detail the model and our probabilistic FIFO ordering algorithm in Section 3. We present an experimental evaluation of the proposed algorithm in Section 5. We review related work in Section 6, and offer some concluding remarks in Section 7.

2 Overview of Problem and Solution

We motivate this work through an experimental comparison between B-DRP and Apache ActiveMQ.[1] Our purpose here is to exemplify the problem at a high level, so we give only a cursory description of the experiment, focusing on the comparative analysis of the two systems. Later in Section 5.1 we discuss this and other experiments in greater depth. The experiment measures the throughput and the rate of FIFO violations in a content-based publish/subscribe network of 8 brokers subjected to increasing message traffic. The network topology is a graph of diameter 3. Each broker runs on a dedicated machine and serves 50 clients running on the same machine. All 400 clients are subscribers, but only 50 also act as publishers. In order to simulate a wide area network, we apply transmission delays and bandwidth limits on inter-broker links. In particular, the bandwidth limit is 10Mbps and the transmission delay is 50ms with a dynamic runtime variability of ±5ms which is typical of the Internet, based on different Internet measurements.[2]

Clients generate a synthetic workload of subscriptions and publications, and the generation algorithm is parametrized so as to induce an intense stream of messages to a few subscribers combined with a steady but slower flow to all other subscribers. The experiment modulates the publication rates over a period of 300

[1] ActiveMQ (http://activemq.apache.org/) is a very popular and reportedly very efficient messaging system that also implements a content-based publish/subscribe service as part of the Java Messaging Service.

[2] For example see measurements by RIPE Network Coordination Centre (RIPE) available at http://www.ripe.net/data-tools/stats/ttm/ttm-data

seconds using two groups of high-rate and low-rate publishers, respectively. Two high-rate publishers, each one publishing 1200 messages per second, join the network at 90 and 190 seconds, respectively, and cause the two noticeable increases in the aggregate input and output rate. The remaining 48 are low-rate publishers, with a publication rate that slowly ramps up from about 1.5 messages per second at the beginning of the experiment to a maximum of 25 messages per second at the end of the experiment. This mixed workload is intended to show, to the extent possible in a single experiment, the general reaction of the broker network to abrupt as well as gradual increases of publication rates, and also to congestion, since the workload is also designed to reach the maximum (collective) delivery capacity of the network for both systems.

Figures 1a and 1b show the throughput measurements. In particular, we plot the rates of publication, message deliveries (i.e., notifications) and false negatives for both ActiveMQ and B-DRP. (A false negative occurs when a subscriber does not receive a published message that matches its subscriptions, and is typically caused by congestion.) Both networks gracefully handle gradual and sudden increases of the aggregate input load during the first 200 seconds of the experiment. At this point B-DRP reaches its delivery limit of about 30000 messages per second, and from then on it starts dropping messages, as evidenced by a raise in the rate of false negatives in the diagram. ActiveMQ handles the growth of the input rate up to time 240, when it delivers 32500 messages per seconds. However, after this peak point, it exhibits a sudden reduction in delivery rate. Also, about 20 seconds after this congestion point, ActiveMQ brokers start blocking publishers, presumably to counter congestion.

As for ordering, ActiveMQ delivers all its notifications in FIFO order—as it should, according to the Java Messaging Service specification—while B-DRP, which is designed as a best-effort network, does not prevent FIFO violations, and in fact incurs a rate of violations that is roughly proportional to the delivery rate (see Figure 1c).

In summary, we observe that ActiveMQ maintains FIFO ordering at all times but suffers an almost catastrophic reaction to congestion, while B-DRP reacts more gracefully to congestion but incurs a significant number of FIFO violations. Our goal in this paper is to combine the best behaviors of these two systems. In particular, we argue that FIFO ordering can be best supported as an optional, end-to-end service implemented on top of a best-effort publish/subscribe system.

2.1 FIFO ordering

FIFO is a simple ordering condition defined for each sender/receiver pair: considering a sender s and a receiver r, for every pair of messages m_1 and m_2 sent by s and received by r, a FIFO violation occurs whenever s sends m_1 before m_2 but r receives m_2 before m_1. It is also useful to express this condition in terms of the total travel time of each message: let $departure(m)$ be the departure time of a message m, and assume $\delta = departure(m_2) - departure(m_1) > 0$; let $arrival(m)$ be the arrival time, and $delay(m) = arrival(m) - departure(m)$ the total travel time of a message m. Then, a FIFO violation occurs when $delay(m_1) - delay(m_2) > \delta$.

Furthermore, the total travel time of a message can be expressed as the sum $delay(m) = delay^*(m) + vardelay(m)$ of a nominal delay, $delay^*(m)$, representing the long-term-average link, queuing, and processing delays, plus a short-term-variable delay $vardelay(m)$. This distinction is useful because, ignoring pathological cases, FIFO violations occur only when the departure interval δ is small, and therefore when the long-term-average delays of m_1 and m_2 can be reasonably considered constant. This means that FIFO violations are essentially a function of the short-term-variable delays and the departure interval δ. Specifically,

$$\text{FIFO violation} \Leftrightarrow vardelay(m_1) - vardelay(m_2) > \delta \tag{1}$$

Equation (1) expresses the essence of the problem as well as the idea upon which we develop a solution.

Our guiding principle is to address FIFO violations with an end-to-end solution. This means that we propose to detect FIFO violations and perform the necessary reordering on the receiver side and independently of the underlying network. This mechanism can be incorporated into the client's middleware or be part of the application logic. This design has multiple advantages, the most important of which is that it is applicable to virtually every publish/subscribe system, regardless of their architectures, routing protocols, and broker technologies. Moreover, maintaining FIFO ordering within brokers can be memory intensive and would delay all messages without distinction. By contrast, when ordering is handled by end-points, it is up to the client to decide the right balance between strictness of the ordering and cost in terms of additional delivery delay.

FIFO ordering is typically implemented with sequence numbers attached to each message by the sender to reflect the sending order, and used by receivers to follow the same order for delivery. One might try to

apply the same sequencing technique to content-based communication. However, in this case, holes in the sequence (e.g., receiving m_7 immediately after m_5) must be treated differently. Specifically, because of the implicit addressing of the content-based model, the receiver does not know—and in some cases it can not know—whether a hole in the sequence is due to a message being delayed along the delivery path (e.g., due to its longer processing time) or whether that message was not supposed to be delivered at all because it does not match the receiver's interests.

Therefore, in order to avoid (or minimize) FIFO violations, we must solve two problems: first, a receiver must decide whether or not to wait for a missing message; second, if the missing message is determined to be likely to arrive, the receiver must determine an appropriate buffering time (or "latch" time) for the message(s) received out of order. One might argue that the receiver can always wait until the missing messages arrive. However, due to the best-effort nature of the service, those messages may get lost, leaving the receiver in a live-lock condition. So eventually, the receiver must timeout after a certain latch time and drop or deliver the buffered messages. The next section details our solution to each one of these two problems.

Our approach is to give receivers a way to answer these questions on the basis of the condition defined in Equation (1). In particular, we propose to carry with each message its departure time, the departure time of the preceding message, and a summary of the content of some previous messages. With departure times (time-stamped by senders) and arrival times (recorded locally) a receiver can continuously measure the distribution of end-to-end short-term-variable delays. Also, upon receiving message m_2 in the absence of the preceding m_1, a receiver can use the time stamps on m_2 to compute the sending interval δ between m_1 and m_2. Then, with δ and the measured distribution of delays, the receiver can decide, up to a set error probability, whether m_1 may have been delayed and, using the content summary carried by m_2, whether m_1 may arrive. If so, the receiver determines, also based on δ and the delay measurements, how long to hold m_2 so as to avoid a FIFO violation without delaying the delivery of m_2 excessively.

3 Probabilistic FIFO Ordering

The method we propose is probabilistic in nature, since it is based on a probabilistic model of delay variations. We now detail this model and how we use it to reduce FIFO violations.

3.1 Model of end-to-end delay

We model the end-to-end delay of a message m as the sum of a long-term average delay plus a short-term variation $vardelay(m)$. Since we are interested in comparing the end-to-end delay of pairs of messages sent by the same publisher within a short interval, we consider the long-term-average component of these delays to be the same. Thus, we focus on the delay variation $vardelay(m)$. In particular, we model $vardelay(m)$ as a random variable with a probability distribution whose parameters are also constant during the short interval that separates two consecutive messages.

In general, the variable component of the processing time (including queuing) and the transmission times at each hop in the publish/subscribe network contribute to the end-to-end variable delay. Typically, a broker has a set of tables to store subscriptions and routing information, and forwarding a message involves comparisons against the entries of the subscriptions table and/or a lookup in the routing table. As a result, the processing time may vary according to several factors, including the number of subscriptions, their constraints, the number of attributes in the message, and the matching algorithm, which might itself be randomized.

Furthermore, in a typical modern implementation on multi-processor hardware, the forwarding process is usually parallelized for maximum throughput, at a minimum with each message handled by a separate thread, and possibly with finer-grained parallelism. Therefore, since forwarding incurs minimal (if any) contention on shared data, the processing times for two different messages are mostly independent. Similar considerations apply to the transmission time, although typically with much less variability, to the point that transmission time for two messages published withing a small time frame can be considered equal.

In summary, considering two messages m_1 and m_2 that might give rise to a FIFO violation, we model their short-term variable delays $vardelay(m_1)$ and $vardelay(m_2)$ as two independent and identically distributed random variables whose distribution depends essentially on the processing time in brokers. (We validate this model experimentally and discuss our findings in Section 5.1.) Thus, our goal is to characterize this distribution in general, and then to measure and parametrize it at run-time.

3.2 Measuring delay differences

Measuring end-to-end delays with a significant precision requires synchronized clocks, and therefore is not practical outside of a tightly controlled environment. On the other hand, the *difference* between the delays of two messages m_1 and m_2 can be readily computed, without synchronized clocks, using the time stamps associated with messages. In practice, a receiver stores the departure time $departure(m_i)$ stamped on m_i by the sender, records its arrival time $arrival(m_i)$, and also records departure and arrival times, $departure(m_{i-1})$ and $arrival(m_{i-1})$, for the previous message m_{i-1}. With this information, it computes $delay(m_i) - delay(m_{i-1}) = [arrival(m_i) - arrival(m_{i-1})] - [departure(m_i) - departure(m_{i-1})]$. The crucial point here is that, by subtracting a departure time from a departure time, and an arrival time from an arrival time, the result is not affected by the lag between the two clocks. We do assume though that the imprecision due to clock drift during delivery is negligible.

In summary, a receiver can measure the distribution of the difference between end-to-end delays, and can then use it as the basis for the estimation of the probability of FIFO violation and the estimation of the optimal latch time. In our model, any two messages that a subscriber receives from a specific publisher go through the same route and hence the same number of brokers. Consider two messages m_x and m_y received by a subscriber that is k brokers away from the publisher. Subtracting the delay of two messages cancels out the constant component of the delay and the subtraction reduces to subtracting two random variables. Writing each variable in terms of its components we have:

$$\begin{aligned} &delay(m_x) - delay(m_y) = \\ &(X_1 + X_2 + \cdots + X_k) - (Y_1 + Y_2 + \cdots + Y_k) = \\ &(X_1 - Y_1) + (X_2 - Y_2) + \cdots + (X_k - Y_k) \end{aligned} \tag{2}$$

where X_i and Y_i, $1 \leq i \leq k$, are the independent and identically distributed random variables representing the processing time of messages m_x and m_y at each broker i. Observe that in the last form of Equation (2) each term of the summation (i.e., $X_i - Y_i$) is itself the difference between two independent and identically distributed random variables and hence is a symmetric random variable with a mean of zero. As such, without making any further assumption about any of the random variables involved in this equation we can find probabilistic bounds on the value of the above delay difference using probabilistic inequalities such as Chebyshev's inequality. In more specific cases, when the broker-hop count is known (e.g., as a field in the message header similar to IP header) we can also use Bernstein inequalities or Hoeffding's inequality to find better bounds. This is indeed of great advantage because as we will detail later, to find the latch time we need to find the probability of delay difference being more than a given value.

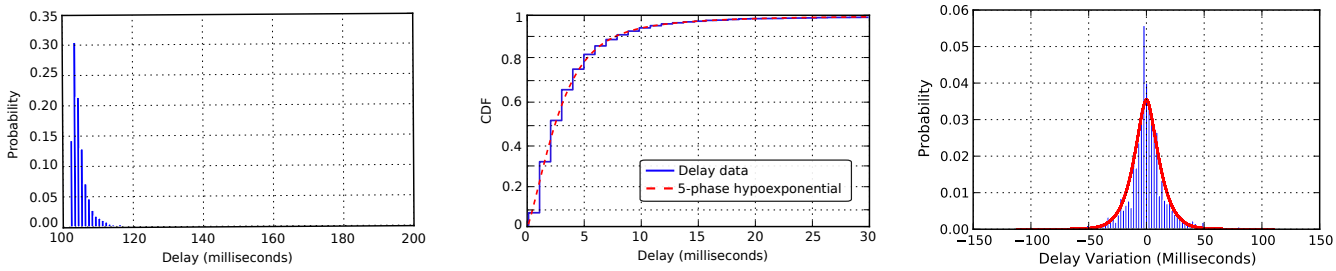
Unfortunately, space limitation does not allow us to elaborate more on the use of probabilistic inequalities in the general case. Instead, we focus on finding a more accurate characterizations of the delay difference distribution and how to measure its parameters. To this end, finding the distribution of end-to-end delays will enable us to find the distribution of delay differences. In the next section we will analyze the distribution of end-to-end delays in more detail.

3.3 End-to-end delay distribution

In order to model the difference between end-to-end delays, which is the observable distribution for a receiver, we start from the distribution of end-to-end delays. In the context of IP networks, various researchers have proposed different methodologies and distributions to model end-to-end IP-level packet delay. For instance Zhang et al. found that a power-law distribution offers a good model [7], while Mukherjee [8] reported that Internet packet delays can be represented by a shifted Gamma distribution whose shape and location factor depend on traffic load and path length.

To extend some of these results to the case of content-based publish/subscribe systems, and to study the dynamics and distribution of end-to-end delays, we conducted experiments with a variety of parameters such as network size and topology, subscription and publication patterns and rates, and link delays. We then synchronized publishers and subscribers via NTP up to a clock difference of less than one millisecond, which enabled us to accurately measure the end-to-end delay of messages with a negligible error.

Our first observation is that neither Gamma nor power-law distributions properly fit the traces of the end-to-end message delay. Figure 2a shows the delay distribution of messages received by a subscriber from a publisher thorough 3 brokers. The inter-broker links have an assigned delay of 50ms. The links that connect



(a) (b) (c)

Figure 2: (a) End-to-end delays for a sender/receiver pair 3 brokers apart. (b) Cumulative distribution of end-to-end delay samples fitted in a 5-phase hypoexponential distribution. (c) Histogram of the delay difference for a sender and receiver separated by 5 brokers. The thick line is the approximation with the sum of two Laplacian random variables.

subscriber and publishers to their local brokers have no delay. Since there are two inter-broker links, all the delays have a constant component of 100 milliseconds. This distribution has two pronounced characteristics: a long tail, which is composed of low frequencies at large values, and a large density around its mean.

As mentioned in Section 3.1, the variable component of the delay of a message is the sum of the processing delays at all the brokers it passes through. So, we start the analysis of the distribution of end-to-end delays (Figure 2a) with a specific experiment to measure the distribution of processing times in a single broker. In a typical setup with a few brokers and 10 clients per broker, we observed that most messages take a few milliseconds to be processed while a few of them need longer processing times. More specifically, in the case of our subject system B-DRP, measurements with different combinations of workload parameters (number and sizes of subscriptions, number and sizes of messages) reveal that the processing time is best fit by an exponential distribution.

We therefore proceed to model the variable component of the end-to-end delay as the sum of n exponentially distributed random variables, where n is the number of brokers between the publisher and the subscriber. This distribution is called a *hypoexponential distribution* which is a member of a general class of distributions called *phase type distributions*. To test this modeling hypothesis, we used the method described by Asmussen et al. [9] to fit the measured end-to-end delay in a hypoexponential distribution with the appropriate number of phases, where a phase corresponds to a hop in the network. Before fitting each data set into the distribution, we removed the constant component of the samples (i.e., the delay caused by the inter-broker links). We performed the sampling and fitting process with a variety of configurations and different number of brokers and topologies with diameters of up to 10. In all cases, the data closely follow the theoretical distribution.

Figure 2b shows the cumulative distribution function of 6500 samples of end-to-end delay measurements, for a given publisher-subscriber pair, fitted into a hypoexponential distribution with 5 phases (in the experiment, the publisher and the subscriber were 5 brokers apart). Next, we detail how we use this model to find the distribution of delay *differences*.

3.4 Distribution of delay differences

We established that the end-to-end delay of a message is a hypoexponential random variable, resulting from the sum of exponentially distributed random variables, each representing the processing time at a broker.

Therefore each term in the second form of the Equation 2 (i.e., $X_i - Y_i$) is the difference of two identically distributed exponential random variables, which is known as a *Laplacian* random variable. It follows that the distribution of differences of end-to-end delays is the sum of independent Laplacian random variables. The probability density function of a Laplacian random variable is $f(x) = \frac{1}{2b} e^{\frac{-|x-\mu|}{b}}$ where μ is the mean of the distribution and b is its scale parameter. Due to the linearity of expectation, μ is zero for all of the above Laplacian random variables that are the result of subtracting two exponentially distributed random variables (e.g., $X_i - Y_i$). This is because all the brokers run the same forwarding algorithms, thus we can assume that b is similar for all the Laplacian random variables.

So, our analysis shows that the difference between end-to-end delays for a given publisher/subscriber pair can be modeled as the sum of k Laplacian random variables, where k is the number of hops between the publisher and the subscriber. However, unfortunately, determining an analytical expression of the distribution

of the sum of $k > 2$ Laplacian random variables with different scale parameters is still an open problem [10]. So, as an approximation, we use the statistical properties—namely, the probability distribution, cumulative density, and quantile functions—of the sum of *two* Laplacian random variables. Even though this model is not mathematically rigorous, we have empirical evidence that the two-sum distribution also fits reasonably well the sum of up to 8 Laplacian random variables. The sum of two independent and identically distributed Laplacian random variables with mean $\mu = 0$ and scale factor b has probability density $f(x) = \frac{(|x|+b)}{4b^2} e^{-\frac{|x|}{b}}$ and cumulative distribution $F(x) = \Pr[X < x]$ for $X > 0$

$$F(x) = 1 - \frac{(2b+x)}{4b} e^{-\frac{x}{b}} \qquad (x > 0) \tag{3}$$

The estimation of the scale factor b based on a set of n samples is possible with maximum likelihood estimation, which yields $b = \frac{2}{3n} \sum_{i=1}^{n} |x_i|$. Figure 2c shows the histogram of delay differences for messages received 5 hops away from the sender. The thick line represents the sum of two Laplacian random variables whose parameter is estimated from the data. The sharp spike around zero, falls outside of the approximate distribution because of the approximation of sum of 5 random variables to only two. In other words, as the number of broker-hops increases, the density of the real distribution increases around the mean and the tails become shorter, while the approximation is less dense around zero but has longer tails. This does not cause a problem though, since in determining the latch time, the likelihood of the extreme values of delay difference is used (i.e., the tails of its distribution) which we will detail next.

3.5 Determining the latch time

Based on the model we developed, we now go back to Equation (1) to estimate the probability that m_1 and m_2 are received out of order (a FIFO violation). This probability is a function of the difference between their departure times, $\delta = departure(m_1) - departure(m_2)$. In particular,

$$\begin{aligned} &\Pr[\text{FIFO Violation}] = \Pr[delay(m_1) - delay(m_2) > \delta] \\ &= 1 - \Pr[delay(m_1) - delay(m_2) < \delta] = 1 - F(\delta) \end{aligned}$$

where $F(\cdot)$ is the cumulative distribution of delay differences.

Whenever the receiver detects a gap in the sequence numbers, it can virtually increase δ by latching the messages whose delivery would cause the FIFO violations so that the probability of a FIFO violation drops below a given threshold. More precisely, we would like to determine a latch time τ that reduces the FIFO violation probability below a given threshold P_t for a pair of messages m_1 and m_2 published δ time units apart from each other. Thus

$$\tau = F^{-1}(1 - P_t) - \delta \tag{4}$$

where F^{-1} is the quantile function of the delay variation. Intuitively, τ is the *minimum* amount of time that the receiver has to hold m_2 and wait for the missing message m_1 based on the sampled delay difference. We call P_t the *FIFO violation coefficient*. Higher values of P_t map to smaller latch times and more FIFO violations. F^{-1} is the inverse of Equation (8) and corresponds to

$$F^{-1}(p) = b[\omega(4e^{-2}(p-1)) + 2] \qquad (0.5 \le p \le 1) \tag{5}$$

where $\omega(\cdot)$ is the *Lambert Omega Function*, and can be efficiently computed using several existing numerical methods.

Now let us consider cases with more than one message missing (e.g., a receiver receives message m_6 immediately followed by m_{10}).

Let $\Phi(m,n)$ denote the occurrence of a FIFO violation for messages m and n, let $\delta_{m,n} = departure(m) - departure(n)$ denote the time difference between the publication time of two messages m and n, and let $\tau_{m,n}$ be the latch time given by Equation (5) for messages m and n. Since $\delta_{10,9} \le \delta_{10,8} \le \delta_{10,7}$ it follows that $\Pr[\Phi(m_9, m_{10})] \ge \Pr[\Phi(m_8, m_{10})] \ge \Pr[\Phi(m_7, m_{10})]$ and therefore $\tau_{9,10} \ge \tau_{8,10} \ge \tau_{7,10}$.

In words, in this probabilistic model, the latch time is independent of the number of messages in a chain of missing messages. In such cases, in order to calculate the latch time, the receiver only considers the time difference between the latest received message and the latest missing message.

Appendix A shows the process of developing the statistics of the sum of two independent and identically distributed Laplacian random variables.

3.6 Publication record

So far we have assumed that whenever there is a gap in the message sequence number, the missing messages would match the interests of the receiver. This assumption enforces the assessment of a latch time upon every message that causes a gap in the sequence, even when the missing messages are not even supposed to be received because they do not match the subscriber's interest. Obviously, this may introduce unnecessary delivery delays.

To eliminate (or reduce) this problem, we propose to attach to each message some information about previously published messages along with their publication timestamps. We call this information the *publication record* of the publisher. As a simplistic example, consider attaching to each message a copy of the previous 3 messages sent by the same publisher. In this case, a receiver receiving m_{10} right after m_6, and therefore detecting a gap of three messages, might be able to deliver m_{10} immediately after checking that none of the missing messages (attached to m_{10}) matches its subscriptions. The question then becomes how to compile a compact and yet informative publication record.

In topic-based pub/sub systems this is easily achieved by attaching the topic of the last k messages to each new publication. Things are not as simple in content-based publish/subscribe systems, although it is possible to attach a summary of the content of the previous k messages. A good encoding for this summary is a message representation based on Bloom filters that we developed for B-DRP. The salient properties of this encoding, which we detail elsewhere [6], are that it is compact and it admits to a fast matching algorithm, but it may incur false positives, meaning that an encoded message may be found to match the interests of the receiver while the original message would not. This does not compromise correctness but may lead to unnecessary delays. Nevertheless, given that in general only a small percentage of the publications of a publisher match the interests of a given subscriber, in most cases this simple method is effective in preventing unnecessary delivery delays. We call this the *enhanced mode* of the probabilistic FIFO ordering protocol as opposed to the *basic mode* in which messages do not carry any publication record.

As mentioned above at the end of Section 3.5, when the sequence number gap contains more than one message, in basic mode the receiver has to consider only the latest missing message. Instead, in enhanced mode, the receiver has to consider only the latest missing message that is found to match local subscriptions. Referring to the example where a receiver receives message m_6 immediately followed by m_{10}, if the receiver detects that m_9 does not match local interests (through the publication record attached to m_{10}) but m_8 is of interest, it takes m_8 into account to calculate δ in Equation (4) since m_9 will not be received anyway.

The size of the publication record attached to each message is controlled by the publisher. A larger record translates into shorter delivery delays, at the expense of a greater bandwidth consumption. In general, high publication rates require large publication records because the interval between publications is smaller and thus the probability of out-of-order deliveries is higher. We are currently studying ways in which we can exploit temporal locality of events to increase efficiency of our encoding scheme. Generally speaking, temporal locality of events implies that events published close to each other in time (by the same publisher) are likely to have similar contents, which could lead to additional compression in publication records and therefore to the reduction of transmission overhead.

4 Algorithmic Description

Algorithm 1 shows the core of our probabilistic FIFO ordering mechanism. A receiver (subscriber) executes a separate instance of this algorithm for each sender (publisher) from which it receives messages, and maintains a separate set of variables associated with that sender. The configurable parameters of the algorithm are the FIFO violation coefficient P_t and the size q of the ring buffer that stores the delay-difference samples.

The main algorithm (starting on line 5) is executed for each message received from the publisher. A variable called *base* stores the previously received message with the highest sequence number. The algorithm starts by computing the delay difference with respect to *base*, and uses that to update the parameters of the distribution of delay difference (line 7). If the received message m_n causes a gap in the sequence number, the algorithm assesses a latch time based on the difference in departure time between m_n and the latest missing message that is known (in enhanced mode) or assumed (in basic mode) to match the local subscriptions. In particular, in basic mode the relevant message is assumed to be the immediate predecessor m_{n-1} whose departure time is attached to m_n. Conversely, in enhanced mode, the receiver looks in the publication record (attached to m_n) for the latest missing message that matches its subscriptions (line 18). If no such message is found in the

```
1: procedure initialize
2:     base ← ⊥                                          {last in-order received message}
3:     β ← 0                                             {scale factor of the distribution of delay difference}
4:     Q ← RingBuffer(q)                                 {ring buffer of size q}

5: upon receiving m_n from publisher do
6:     if base ≠ ⊥ then
7:         x ← [arrival(m_n) − arrival(base)] − [departure(m_n) − departure(base)]
8:         update(x)                                     {update the distribution}
9:     n ← sequence(m_n)
10:    if n = 0 or n = sequence(base) + 1 then
11:        schedule(m_n, 0)                              {schedules m_n for immediate delivery}
12:    else if n < sequence(base) then
13:        schedule(m_n, 0)
14:        return
15:    else
16:        if publication_record(m_n) ≠ ⊥ then
17:            R ← publication_record(m_n)
18:            m_* ← latest message in R that matches local subscriptions and was not already received
19:            if m_* = ⊥ then
20:                m_* ← earliest message in R
21:            t_* ← read departure(m_*) from R
22:            δ ← departure(m_n) − t_*
23:        else
24:            t_{n−1} ← read departure(m_{n−1}) from m_n
25:            δ ← departure(m_n) − t_{n−1}
26:        τ ← latch_time(δ)
27:        schedule(m_n, τ)
28:    base ← m_n

29: function latch_time(δ)
30:    τ ← β[ω(4e^{−2}(−P_t)) + 2] − δ                   {(P_t ≤ 0.5)}
31:    return τ

32: procedure update(x)
33:    Q.append(x)                                       {add the sample to the ring buffer}
34:    β ← (2/3q) Σ_{x_i ∈ Q} |x_i|                      {update scale factor of the distribution}
```

Algorithm 1: Probabilistic ordering FIFO algorithm run by a recipient for each publisher

publication record then the receiver conservatively assumes that the latest matching message was published at the same time as the earliest message in the publication record (line 20).

Having computed an appropriate latch time τ, the receiver schedules the message for delivery to the receiver application. This is done through a procedure *schedule*(m,τ) that also assures the ordered delivery of all scheduled messages. This procedure (not shown in the listing) is analogous to the mechanisms found in most sliding-window protocols. In essence, the scheduler maintains a queue of pending messages and a set of corresponding timers. A message is queued until its timer expires or all earlier messages have been delivered, at which point the message is also delivered to the application and removed from the queue.

Since the ordering protocol is probabilistic, the latch time of a set of messages may not be large enough to cover the time needed for the missing messages to arrive. In such cases, the timer associated with the latched messages expires and the messages are delivered. When the missing messages are received, they might be simply dropped or delivered to the application, thereby causing a FIFO violation (see line 13). We have implemented this treatment as a configurable parameter of the ordering protocol.

5 Evaluation

We implemented the recovery protocol as a pluggable module which integrates into any publish/subscribe application and protocol. Specifically, the publication record and other metadata that is required by the ordering protocol is attached to messages as an array of bytes, perceived by brokers as application payload.[3]

We now present the experimental evaluation of the FIFO ordering method proposed in this paper. This evaluation addresses three high-level questions. First, it validates the statistical models, developed in Sec-

[3]Most implementations of the Java Messaging Service (JMS) are capable of carrying an opaque payload.

tion 3, upon which the method is built. Second, it evaluates the benefits, costs, and scalability of the method in its basic and enhanced form. Third, it evaluates the ability of the method to respond and adapt to dynamic workloads.

We ran all our experiments on a testbed consisting of a cluster of Dell PowerEdge with two dual core 2GHz AMD Opteron processors and 4GB of main memory running Linux with a 2.6.32 kernel. Connectivity is through an isolated high-throughput Gigabit Ethernet switch. B-DRP is implemented in Java and runs on the 64-bit open-JDK VM.

5.1 Network delay model validation

This first experiment we describe corresponds to the scenario described in Section 2. However, whereas in Section 2 we compared global statistics of throughput and FIFO violations for ActiveMQ and B-DRP, here we focus on B-DRP (because we are interested in modeling best-effort delivery) and look closer at the distribution of end-to-end delays and their variations.

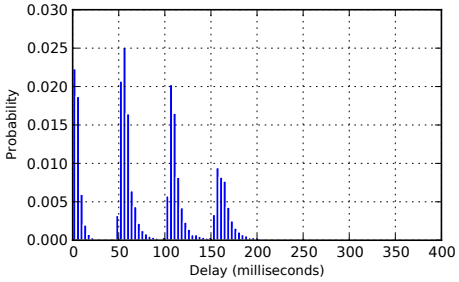


Figure 3: Distribution of end-to-end delay for all of the delivered messages during the first 90 seconds of the experiment.

Figure 3 shows a histogram of the end-to-end delay for nearly 250000 messages that were delivered across the network during the first 90 seconds of the experiment. Recall that this initial phase is characterized by a slow (and slowly growing) flow of publications. Below we also examine the later phases of the experiment when high-rate publishers cause congestion, and therefore cause a significant shift in the delay distribution. The histogram shows 4 distinct clusters corresponding to the hop-distance between publisher and subscriber. For example, messages that pass through 3 brokers (two broker-to-broker hops) have a transmission delay of around 100 milliseconds. The histogram of Figure 3 is qualitatively consistent with our model of end-to-end delay. Now, in order to validate the model more precisely, we isolate a single publisher/subscriber pair and measure end-to-end delays and delay differences.

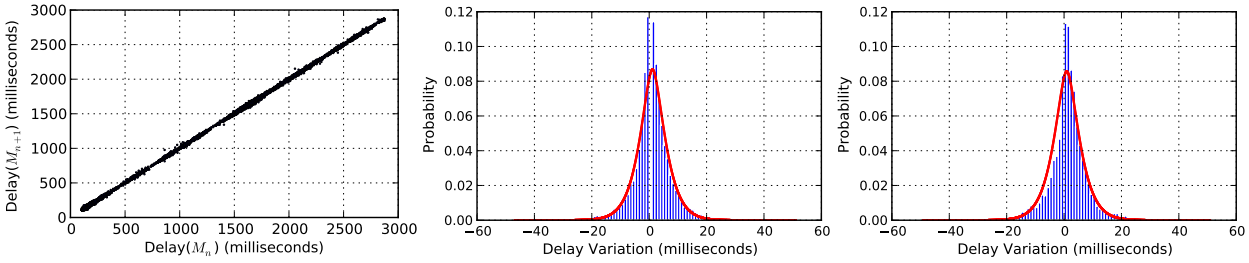
We first examine the delay of pairs of consecutive messages recorded over the entire duration of the experiment. The results are reported in the scatter-plot of Figure 4a. The plot highlights two facts. First, the delays of two consecutive messages are highly correlated; second, the delays vary significantly throughout the experiment, and since the data refers to a single sender/receiver pair, this indicates the effect of significant queuing delays. We also note that we purposely select a sender/receiver pair that experiences an intense flow of messages that ultimately causes congestion in the intermediate brokers.

We now take a closer look at the effect of delays and congestion on delay variation. In particular, we test our intuition that queuing delays do not have any substantial effect on the distribution of delay variation. To do that, we compute the distribution of delay variations between consecutive messages for pairs of messages subject to delays within two ranges, corresponding to the areas marked with dotted lines in Figure **??**. The resulting histograms, plotted in Figure 4, demonstrate that the delay variations are essentially independent from the delay.

To confirm this visual analysis, we use Wilcoxon rank-sum test whether this data is consistent with our hypothesis that the two datasets follow the same distribution. The resulted p-value of the Wilcoxon test is 0.35 which confirms that the evidence is compatible with our hypothesis with a high statistical significance.

5.2 Effectiveness of the ordering protocol

We evaluated the effectiveness of the ordering protocol through various experiments. In general, these experiments are intended to measure both the reduction in FIFO violations and the additional latency incurred by

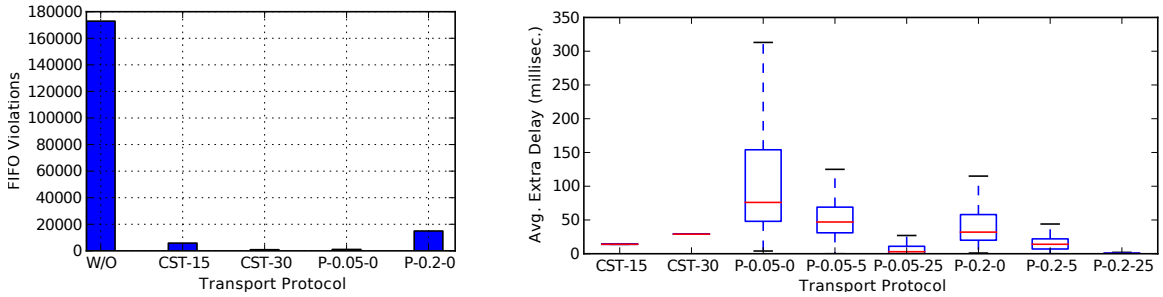


(a) (b) (c)

Figure 4: (a) End-to-end delays of every two consecutive messages for a chosen pair of sender and receiver. (b) and (c) Delay variation distribution for messages with end-to-end delay of $delay(m) \leq 1500ms$ and $delay(m) > 1500ms$ respectively.

the protocol. Specifically, to characterize the trade-offs between these benefits and costs, and also to obtain a comparative baseline, we juxtapose the performance of our probabilistic protocol with that of a simpler protocol that uses a static latch time. This protocol latches each message that creates a gap for a fixed amount of time. However, to obtain the most conservative comparison, we first select the parameters of our probabilistic protocol and measure its performance in terms of FIFO violations, and then configure the static protocol with the *optimal* latch time that achieves the same (or nearly the same) level of FIFO violations. (We determine the optimal static latch time experimentally with a trial-and-error binary search.)

We set up and perform each experiment so that receivers run multiple instances of static and probabilistic ordering protocols with different parameters. This enables us to compare the efficiency of the protocols and the effect of different parameters in the exact same scenario.



(a) Number of FIFO violations (b) Average extra delay

Figure 5: Effectiveness of different FIFO algorithms in an 8 broker setup. (a) Total number of incurred FIFO violations with and without ordering. (b) Average extra delay caused by different ordering algorithms.

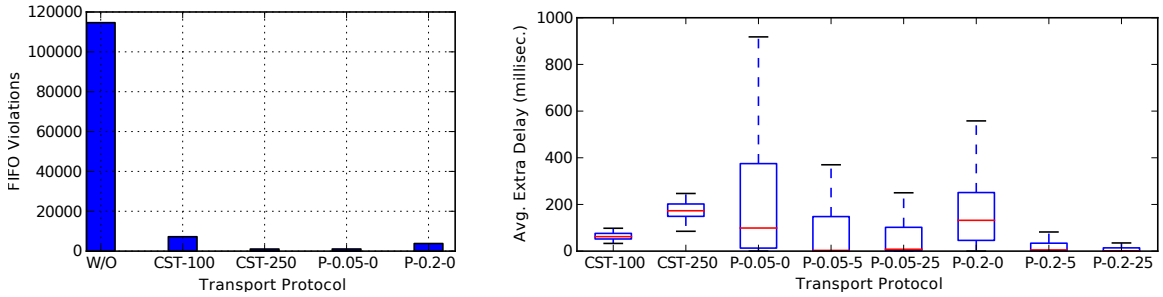
We report the results of our experiments in figures 5 and 6. Each data set corresponding to the static protocol is labeled "CST-t," where t is the constant latch time (milliseconds); and each set of the probabilistic FIFO ordering protocol is labeled "P-x-y," where x is the probabilistic FIFO coefficient P_t, and y is the number of previously published messages whose encoded Bloom filters are attached to each message (in the enhanced version of the algorithm). The probabilistic FIFO ordering protocol also uses a sample buffer Q of size 25 in all the experiments, and uses an encoding of the publication record that uses 16 bytes per message, so for example, "P-0.2-5" and "P-0.2-25" indicate experiments in which the enhancement of the publication record introduces an overhead of 80 and 400 bytes, respectively.

Figure 5a compares the total number of FIFO violations observed by 400 receivers, with and without ordering mechanisms. Note that for a given P_t, the size of the publication record does not have any effect on the performance of the protocol in terms of the number of reduced FIFO violations. Recall in fact that the use of the publication record allows the receiver to assess a reduced latch time, but does not change the behavior of the protocol in terms of FIFO violations. Hence, we only plot the number of FIFO violations for the basic mode of the probabilistic protocol. For example, in Figure 5a with P_t equal to 0.05, the number of FIFO violations for P-0.05-0 is the same for P-0.05-25; what differs is the extra delay, as we will show later. In total there were almost 173000 out-of-order receptions. The probabilistic FIFO ordering with P_t set to 0.05 mitigated more than 99% of the FIFO violations, performing as well as CST-30. With $P_t = 0.2$, the probabilistic FIFO algorithm

reduced FIFO violations by about 91%.

To demonstrate the benefit of the probabilistic FIFO ordering algorithm with respect to minimizing the overall delivery delay when compared to the static protocol, we first define a measure we call the *average extra delay*. If M is the set of all the messages that a given subscriber received from a given publisher, we define the average extra delay for the messages of that stream as $\frac{1}{|M|}\sum_{m\in M} latch\_time(m)$.

Figure 5b presents the average extra delay caused by the static and probabilistic FIFO algorithms for all of the pairs of sender and receiver. Unsurprisingly, the static algorithm with constant latch time induces the same average extra delay for all nodes. P-0.05-25 performs better than CST-30 in terms of average extra delay. P-0.05-0 (basic mode) causes much larger extra delays, but when enhanced with a publication record of only 5 messages (P-0.05-5), it incurs a considerably smaller extra delay. Note that the large average extra delay in basic mode is largely due to the intentionally extreme scenario in which two high-rate publishers are combined with subscribers configured to receive a very high portion of their publications (more than 80%). This is an extreme case that is relatively uncommon in a publish/subscribe network. In other words, whenever the publisher sends messages at lower rates, the difference in the average extra delay between the static protocol and our adaptive protocol increases significantly. This is because the adaptive protocol is sensitive to the time difference between two consecutive publications while any static protocol latches messages irrespective of the publication rate.



(a) Number of FIFO violations

(b) Average extra delay

Figure 6: Effectiveness of different FIFO algorithms in a 46-broker setup. (a) Total number of FIFO violations with and without ordering. (b) Average extra delay caused by different ordering algorithms.

To demonstrate the performance of the protocol in larger networks, in Figure 6 we present the performance of the probabilistic ordering in a network of 46 brokers where each broker serves 10 clients. We use a workload with similar characteristics to the 8-broker experiment, except that this one runs for 240 seconds. We choose a low-degree network topology with a graph diameter of 15. Due to this larger diameter, constant latch times of 30 or 50 milliseconds are not effective. Indeed, like in the previous experiments, we ran several setup experiments to find the optimal constant latch time for our comparative analysis. On the other hand, the adaptive nature of our algorithm captures very well the dynamics of larger-diameter networks. In this experiment, there are more than 115000 out-of-order receptions without any ordering algorithm in place, and our probabilistic ordering algorithm with $P_t = 0.05$ is able to prevent 99.5% of such FIFO violations. This mitigation in basic mode comes at a cost of a maximum 960 milliseconds in average additional delivery delay, while with a publication record of 25, this cost is nearly zero for more than 50% of the nodes and less than 190 milliseconds for 90% of them.

5.3 Adaptivity of the protocol

Figure 7 shows the dynamics of the FIFO-ordering protocol for a publisher/subscriber pair in response to changes in publication rate. The top frame pin-points out-of-order deliveries in the message stream; the second frame shows the publication rate of the publisher (messages per second); the third frame plots the changes in FIFO-violation probability calculated by our algorithm; and the two bottom frames show the changes in latch time when the publication record is 0 (basic mode) and 25 (enhanced mode).

The FIFO violation probability and the latch time follow the trend in the changes of publication rate. This is the result of delay variation and change of time gap between two consecutive messages. Observe that, in the basic version of the protocol, where there is no publication record attached to messages, the latch time spikes

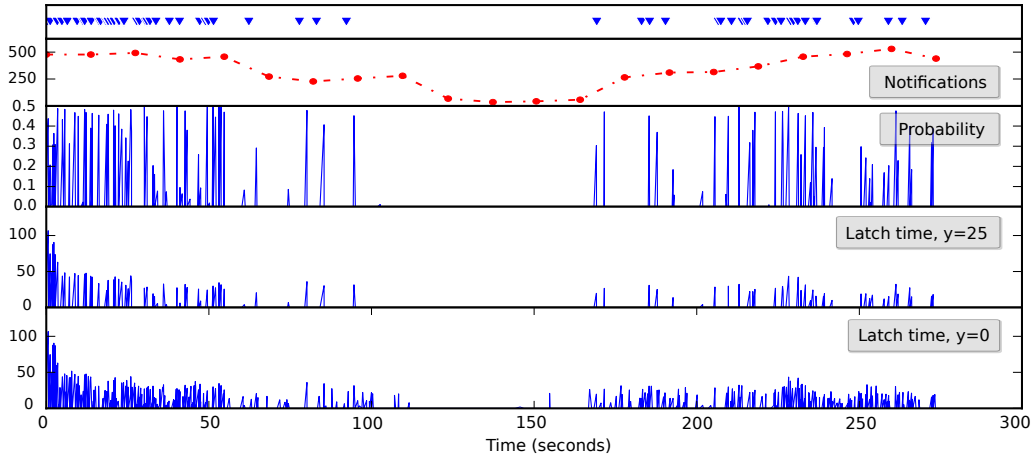


Figure 7: From top to bottom: timestamps of out of order receptions; publication rate; probability of a FIFO violation; latch time in enhanced mode with a publication record of size 25; latch time in basic mode.

more frequently, for there are many cases when the missing message does not match the subscriptions of the receiver but the ordering algorithm latches the messages for that specific time frame.

6 Related Work

The out of order reception of messages due to parallelism and queuing complexities has been acknowledged and studied by the networking community. In particular, Bennet et al. suggest that IP packet reordering is not a pathological behavior but rather, an inevitable outcome of highly parallelized processing [11, 12, 13].

As for content-based communication, systems can be generally divided into two categories with respect to their message ordering guarantees: those that provide an ordered delivery service and those that provide a best-effort service. Systems in the first category are designed to offer a safer abstraction for applications, and are typically implemented with a store-and-forward mechanism. Systems in the second category work under the assumption that ordering violations are reasonably rare, and/or applications can tolerate them, and favor a design that enhances throughput.

Bhola et al. [14] propose a form of store-and-forward mechanism in which publishers and subscribers together with brokers form a tree called "knowledge graph." Soft-state messages labeled "knowledge" and "curiosity" flow downstream and upstream on the tree, and ensure ordered one-time delivery even in the presence of failure. This method can guarantee FIFO and total order, but it introduces complexities in the implementation of the broker and does not easily integrate with existing broker technology.

Aguilera and Storm [15] propose another form of store-and-forward network that guarantees deterministic, uniform total order of messages. In this network, some of the nodes act as *merger* nodes, each one responsible for a subset of the subscribers. All messages go through a sequence of merger nodes to be ordered in a globally uniform manner before they are forwarded to the subscribers. This ordering algorithm assumes that publishers have access to synchronized clocks and that they have a known publication rate. Although this algorithm has the interesting ability to determine an upper bound on the delivery delay, it is prone to substantial delays and has limited scalability. Furthermore, the scheme requires a balanced assignment of subscribers to merger nodes to prevent overloading of some mergers.

The main pitfall of the store-and-forward design is that it induces high delivery delays. Moreover, when the publication rate is high, logging messages onto disk might induce congestion. On the contrary, best-effort systems do not generally log messages onto stable storage, nor they require acknowledgments, and in general do not include any reliability mechanism within the broker network [16, 2, 17]. This results in a streamlined processing of messages that yields high delivery rates and reduces the failures caused by congestion. This difference is evidenced by an experimental comparison between B-DRP, ActiveMQ, and WebSphereMQ [6].

7 Final remarks

In this paper we presented our approach to enhance FIFO ordering in best-effort, content-based publish/subscribe networks. Our general idea is to implement an end-to-end, probabilistic algorithm to avoid FIFO violations. More specifically, first we studied and modeled the causes of FIFO violations, and showed experimentally that the major cause of FIFO violations is the variation in end-to-end delays. Then, based on a simple analytical model of the end-to-end delay, we developed a method to quantify its variation, which we also validated experimentally. This allowed us to devise an algorithm to estimate the probability of a FIFO violation whenever there is a gap in the sequence number of an incoming message stream. The same estimation also allows us to find an adequate latch time for some of the received messages in order to reduce the FIFO-violation probability below a desired threshold. Through experiments, we showed that this method can mitigate up to 99.5% of the FIFO violations while keeping the unnecessary delivery delay to a minimum.

The work presented in this paper is part of a larger project to develop what amounts to a transport layer for a content-based network. Our current and future plans are to develop other traditional functions of a transport layer, such as a method to increase reliability and a method to prevent or control congestion. In this pursuit, we can of course draw from the extensive literature and technical progress in traditional networking. However, we argue that the content-based communication model—and in particular its lack of explicit addresses and therefore its lack of identifiable end-to-end connections—poses interesting and challenging problems also for these other transport-layer functions.

A Statistics of the sum of two Laplacian random variable

The probability density function of a Laplacian random variable is of the following form:

$$f_X(x) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}} \tag{6}$$

where b is the scale factor and μ is called the location factor of the distribution. Assume $Y = X_1 + X_2$ is a random variable defined as the sum of two independent and identically distributed Laplacian random variables with μ equal to zero and equal scale factor of b. We will derive probability density function, cumulative distribution function and quantile function of Y. Also we derive the parameter estimator of b. Note that due to linearity of expectations, $E[Y]$ is zero.

A.1 Probability Density Function

In order to find the probability density function of Y we note that X_1, X_2 and Y can assume negative and positive values. Hence, we need to separate cases when Y is positive from when it is negative. Here we show how $f_Y(y)$ can be derived for positive values of y. To do that, we need to consider three cases: when $X_1 > 0$, $X_2 > 0$, when $X_1 > 0$, $X_2 < 0$ and when $X_1 < 0$, $X_2 > 0$. Now, assuming $Y > 0$, $X_1 > 0$, $X_2 > 0$, we have $X_2 = Y - X_1$ and $X_1 \in [0, Y]$. Since X_1 and X_2 are two independent and identically distributed random variables we have

$$f_Y(y) = f_{X_1,X_2}(x_1, x_2) = f_X(x_1) \,.\, f_X(x_2) \quad such\ that \quad x_1 + x_2 = y$$

and from there

$$f_Y(y) = \int_0^y f_X(x_1) \,.\, f_X(y - x_1)\, dx = \int_0^y \frac{1}{2b}\, e^{-\frac{x_1}{b}} \,.\, \frac{1}{2b}\, e^{-\frac{(y-x_1)}{b}}\, dx = \frac{1}{4b^2}\, e^{-\frac{y}{b}} \int_0^y dx = \frac{y}{4b^2}\, e^{-\frac{y}{b}} \quad (X_1 > 0\,,\, X_2 > 0)$$

Now we find $f_Y(y)$ for cases when $Y > 0$, $X_1 > 0$, $X_2 < 0$. Similar to what we have above, writing X_2 as $X_2 = Y - X_1$ implies that $X_1 \in [Y, \infty)$ and hence

$$f_Y(y) = \int_y^{\infty} f_X(x_1)\,.\,f_X(y-x_1)\,dx = \int_y^{\infty} \frac{1}{2b}\,e^{-\frac{x_1}{b}}\,.\,\frac{1}{2b}\,e^{\frac{(y-x_1)}{b}}\,dx = \frac{1}{4b^2}\,e^{\frac{y}{b}}\int_y^{\infty} e^{-\frac{2x}{b}}\,dx =$$
$$\frac{1}{8b}\,e^{-\frac{y}{b}} \quad (X_1 > 0\,,\ X_2 < 0) \tag{7}$$

Due to symmetry of the Laplace distribution, the third case where $X_1 < 0, X_2 > 0$ yields the same result as Formula 7. Now $f_Y(y)$ is the sum of three cases:

$$f_Y(y) = \frac{1}{8b}\,e^{-\frac{y}{b}} + \frac{1}{8b}\,e^{-\frac{y}{b}} + \frac{y}{4b^2}\,e^{-\frac{y}{b}} = \frac{(b+y)}{4b^2}\,e^{-\frac{y}{b}} \quad (y \geq 0)$$

By following the same method for cases when $y < 0$ we can find the general formula of the density function for positive and negative values of y which is the following

$$f_Y(y) = \frac{(b+|y|)}{4b^2}\,e^{-\frac{|y|}{b}}$$

A.2 Cumulative Density Function and Quantile Function

To find the cumulative density function denoted by $F_X(x)$, we can conveniently utilize the symmetry of the density function. Therefore, we begin by finding $F_X(x)$ for cases when x is negative. We have

$$F_X(x) = \int_{-\infty}^{x} \frac{(b-x)}{4b^2}\,e^{\frac{x}{b}}\,dx = \frac{1}{4b^2}\int_{-\infty}^{x} (b-x)\,e^{\frac{x}{b}}\,dx$$

Using integration by parts, the above integral yields

$$F_X(x) = \frac{1}{4b^2}[(b-x).(b\,e^{\frac{x}{b}}) + \int_{-\infty}^{x} b\,e^{\frac{x}{b}}\,dx\,] = \frac{(2b-x)}{4b}\,e^{\frac{x}{b}} \quad (x \leq 0)$$

and from there we can derive $F_X(x)$ for positive values of x which yield the follwoing formula

$$F_X(x) = 1 - \frac{(2b+x)}{4b}\,e^{-\frac{x}{b}} \quad (x \geq 0) \tag{8}$$

The general form of $F_X(x)$ for both negative and positive values of x can be written as the following

$$F_X(x) = 0.5\,[1 + sgn(x)(1 - \frac{(2b+|x|)}{2b}\,e^{-\frac{|x|}{b}})\,]$$

where $sgn(x)$ is the sign of x. In order to to find $F^{-1}(p)$, the quantile function, we note that for $p \leq 0.5$, the value of $F^{-1}(p)$ is negative and for $p \geq 0.5$ it is positive. Now assuming that $p \geq 0.5$, we find the inverse of the Equation 8. Denoting $F_X(x)$ by p we have

$$\begin{aligned}
&p = 1 - \frac{(2b+x)}{4b}\,e^{-\frac{x}{b}} && thus\\
&1-p = \frac{(2+\frac{x}{b})}{4}\,e^{-\frac{x}{b}} && so\\
&-(2+\frac{x}{b})\,e^{-\frac{x}{b}} = 4(p-1) && which\ can\ be\ written\ as\\
&-t\,e^{-t} = 4\,e^{-2}\,(p-1) && where\ \ t = \frac{x}{b}+2
\end{aligned}$$

The above equation is solved by ω, the *Lambert Omega Function.* This function solves the equation $y = x\ e^x$ by $x = \omega(y)$. As such, the anwser to the above equation is

$$t = -\omega(4\ e^{-2}\ (p-1))$$

By substituting t back, we get the final form of the quantile function

$$F^{-1}(p) = -b\ [\omega(4\ e^{-2}\ (p-1)) - 2)\ ] \quad (0.5 \leq p \leq 1)$$

A.3 Parameter Estimation

Having a set of n samples $x_1, ..., x_n$, the parameter of the probability density function can be found trough maximum likelihood estimation. We denote the estimator of b by β. Given that the values of each sample is independent of the other samples we have

$$p(x_1, ..., x_n|\beta) = \prod_{i=1}^{n} p(x_i|\beta) = \frac{1}{(4\beta^2)^n} \prod_{i=1}^{n} (\beta + |x_i|)\ e^{-\frac{|x_i|}{\beta}}$$

The objective is to find the value of β that maximizes the preceding function. Since it is a positive value (the product of some probabilities) it follows that maximizing its natural logarithm is equal to maximizing the function itself. Taking natural log from the function we have

$$\begin{aligned} &\ln(p(x_1, ..., x_n|\beta)) = \ln(\frac{1}{(4\beta^2)^n} \prod_{i=1}^{n} (\beta + |x_i|)\ e^{-\frac{|x_i|}{\beta}}) = \\ &-n\ln(4\beta^2) + \sum_{i=1}^{n} (\beta + |x_i|) - \sum_{i=1}^{n} \frac{|x_i|}{\beta} \end{aligned} \tag{9}$$

Formula 9 can be seen as a function of β. To find the value of β that maximizes the value of this function, we take its derivative with respect to β. Equating the resulted function to zero and solving for β, yields the answer. Thus, after taking deravative and equating to zero we have

$$\begin{aligned} &-\frac{2n}{\beta} + \sum_{i=1}^{n} \frac{1}{(\beta + |x_i|)} + \sum_{i=1}^{n} \frac{|x_i|}{\beta^2} = 0 \quad so \\ &\sum_{i=1}^{n} |x_i| + \beta^2 \sum_{i=1}^{n} \frac{1}{(\beta + |x_i|)} - 2n\beta = 0 \end{aligned} \tag{10}$$

The above equation does not have a close form answer. In order to approximate its root we approximate the second term of the left side of the equation as seen below. Showing expected value of the random variable X by $E[X]$ we see that

$$\begin{aligned} E\ [\frac{1}{(\beta + |x|)}\ ] = \int_{-\infty}^{\infty} \frac{1}{(\beta + |x|)}\ f_X(x)\ dx = 2\int_{0}^{\infty} \frac{1}{(\beta + x)}\ f_X(x)\ dx = \\ 2\int_{0}^{\infty} \frac{1}{(\beta + x)}\ \frac{(\beta + x)}{4\beta^2} e^{-\frac{x}{\beta}}\ dx = \frac{1}{2\beta^2} \int_{0}^{\infty} e^{-\frac{x}{\beta}}\ dx = \frac{1}{2\beta} \end{aligned}$$

Therefore

$$E\ [\sum_{i=1}^{n} \frac{1}{(\beta + |x|)}\ ] = \frac{n}{2\beta}$$

Substituting this value back in Equation 10 we will have

$$\sum_{i=1}^{n}|x_i| + \frac{\beta n}{2} - 2n\beta = 0 \qquad that\ gives$$
$$\beta = \frac{2}{3n}\sum_{i=1}^{n}|x_i|$$

References

[1] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg, "Content-based publish-subscribe over structured overlay networks," in *ICDCS '05*. Washington, DC, USA: IEEE Computer Society, 2005.

[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, 2001.

[3] E. Fidler, H.-A. Jacobsen, G. Li, and S. Mankovski, "The padres distributed publish/subscribe system," in *In 8th International Conference on Feature Interactions in Telecommunications and Software Systems*, 2005.

[4] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: a high performance publish-subscribe system for the world wide web," in *NSDI'06*. Berkeley, CA, USA: USENIX Association, 2006.

[5] V. Hadzilacos and S. Toueg, *Fault-tolerant broadcasts and related problems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1993.

[6] A. Carzaniga, G. Toffetti Carughi, C. Hall, and A. L. Wolf, "Practical high-throughput content-based routing using unicast state and probabilistic encodings," Faculty of Informatics, University of Lugano, Tech. Rep. 2009/06, Aug. 2009.

[7] H. Zhang, A. Goel, and R. Govindan, "An empirical evaluation of internet latency expansion," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, 2005.

[8] A. Mukherjee, "On the dynamics and significance of low frequency components of internet load," *Internetworking: Research and Experience*, vol. 5, 1992.

[9] S. Asmussen, O. Nerman, and M. Olsson, "Fitting phase-type distribution via the em algorithm," *Scandinavian Journal of Statistics*, vol. 23, 1996.

[10] S. Nadarajah and S. Kotz, "On the linear combination of laplace random variables," *Probab. Eng. Inf. Sci.*, vol. 19, no. 4, 2005.

[11] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," in *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*, 1993.

[12] V. Paxson, "End-to-end internet packet dynamics," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, 1997.

[13] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, 1999.

[14] S. Bhola, R. E. Strom, S. Bagchi, Y. Zhao, and J. S. Auerbach, "Exactly-once delivery in a content-based publish-subscribe system," in *DSN '02*. Washington, DC, USA: IEEE Computer Society, 2002.

[15] M. K. Aguilera and R. E. Strom, "Efficient atomic broadcast using deterministic merge," in *PODC 2000*, 2000.

[16] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Internetworking," in *SIGCOMM '09*, 2009.

[17] A. C. Snoeren, K. Conley, and D. K. Gifford, "Mesh-based content routing using xml," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, 2001.