2011 IEEE International Symposium on Network Computing and Applications

Probabilistic FIFO Ordering In Publish/Subscribe Networks

Amirhossein Malekpour, Antonio Carzaniga, Giovanni Toffetti Carughi, and Fernando Pedone
Faculty of Informatics, University of Lugano, Switzerland
Email: {malekpoa, antonio.carzaniga, giovanni.toffetti.carughi, fernando.pedone}@usi.ch

***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, but 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 automatic adaptability to variable network conditions and input loads.**

I. 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 form, FIFO ordering: 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 "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 delivery 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 II 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 III. We present an experimental evaluation of the proposed algorithm in Section IV. We review related work in Section V, and offer some concluding remarks in Section VI.

978-0-7695-4489-2/11 $26.00 © 2011 IEEE
DOI 10.1109/NCA.2011.12

II. Overview of Problem and Solution

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,

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

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. The next section details our solution to each one of these two problems.

III. 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.

A. 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 IV-A.) Thus, our goal is to characterize this distribution in general, and then to measure and parametrize it at run-time.

B. 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, for example, Chebyshev's inequality. In more specific cases, when the broker-hop count is known (e.g., from a hop-count 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.

Due to space limitations, we do not elaborate further on the use of probabilistic inequalities in the general case. Instead, we focus on a more accurate characterizations of the distribution of delay differences, and how to measure its parameters.

C. End-to-end delay distribution

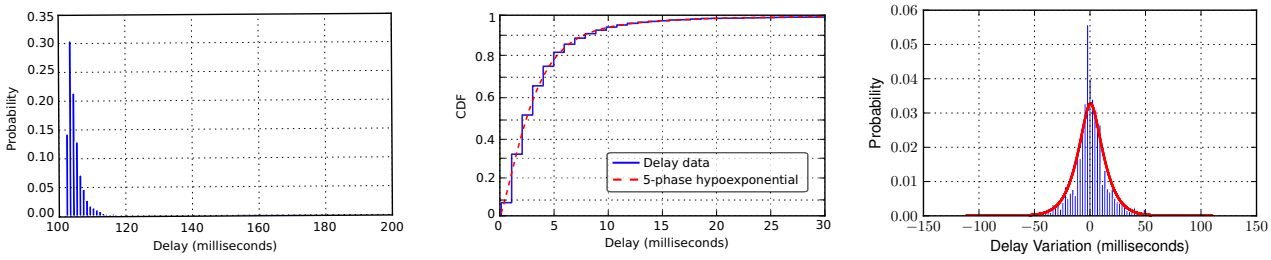
In order to model the difference between end-to-end delays, which is the observable distribution for a receiver, we start from the distribution of end-to-end delays. In the context of IP networks, researchers have proposed different models to describe end-to-end packet delays. For instance, Zhang et 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.

In our experiments with many various combinations of parameters (e.g., network size and topology, subscription and publication patterns and rates, and link delays) we observed that neither Gamma nor power-law distributions fit the traces of the end-to-end message delay. Figure 1a shows the delay distribution for messages received by a subscriber from a publisher through 3 brokers. All delays have a fixed component of 100 milliseconds, since the two inter-broker links have a delay of 50 milliseconds. Beyond that, this distribution has two pronounced characteristics: a long tail (low frequencies at large values) and a large density around its mean.

As mentioned in Section III-A, the variable component of the delay of a message is the sum of the processing delays at all the brokers the message goes through. So, we begin the analysis 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 are processed in a few milliseconds 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. The result is what is called a *hypoexponential distribution*, which is a member of a general class of distributions called *phase type distributions*. To test this model, we use a method described by Asmussen et al. [9] to fit the measured end-to-end delay (minus the fixed inter-broker delay component) within a hypoexponential distribution with the appropriate number of phases, where a phase corresponds to a hop in the network. We measured delays under a variety of configurations and with different numbers of brokers and topologies up to a diameter of 10 hops. In all cases, the data closely follow the theoretical distribution.

Figure 1b 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*.



(a) (b) (c)

Fig. 1. (a) End-to-end delays for a sender/receiver pair 3 brokers apart. (b) Cumulative distribution of end-to-end delay samples fitted to 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.

D. 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 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. 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 1c 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.

E. 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 (3) and corresponds to

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

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 $\delta_{10,9} \leq \delta_{10,8} \leq \delta_{10,7}$ it follows that $\Pr[\Phi(m_9, m_{10})] \geq \Pr[\Phi(m_8, m_{10})] \geq \Pr[\Phi(m_7, m_{10})]$ and therefore $\tau_{9,10} \geq \tau_{8,10} \geq \tau_{7,10}$.

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.

F. 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 III-E, 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.

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.

IV. 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.

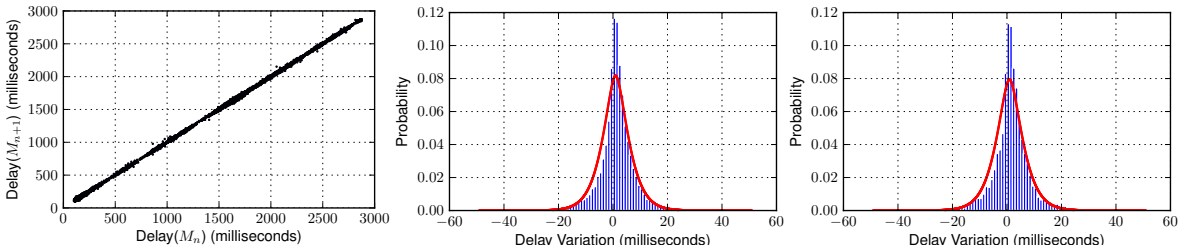
We now present the experimental evaluation of the FIFO ordering method proposed in this paper. This evaluation addresses three high-level questions. First, it validates the statistical models, developed in Section III, upon which the method is built. Second, it evaluates the benefits, costs, and scalability 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. More details on these experiments that we can not include here are available in a technical report [11].

A. Network delay model validation

This first experiment we present corresponds to a network of 8 brokers, diameter 3, in which messages are published with increasing and variable rates intended to induce low traffic, high traffic, and up to congestion in brokers. In these varied conditions, we look at the distribution of end-to-end delays and their variations.

In order to validate the model we formulated, we isolate a single publisher/subscriber pair and measure end-to-end delays and delay differences. We purposely select a publisher/subscriber pair that experiences an intense flow of messages that ultimately causes congestion in the intermediate brokers. 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 2a. The



(a) (b) (c)

Fig. 2. (a) End-to-end delays of consecutive messages for a chosen pair of publisher and subscriber. (b) and (c) Delay variation distribution for messages with end-to-end delay below and above 1500ms, respectively.

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 publisher/subscriber pair, this indicates the effect of significant queuing delays.

We then 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 consider the distributions of delay variations for delays above and below 1500ms, respectively. The two distributions, plotted in figures 2b and 2c, respectively, demonstrate that the delay variations are essentially independent from the delay. A Wilcoxon rank-sum test confirms this visual analysis with a p-value of 0.35.

B. *Effectiveness of the ordering protocol*

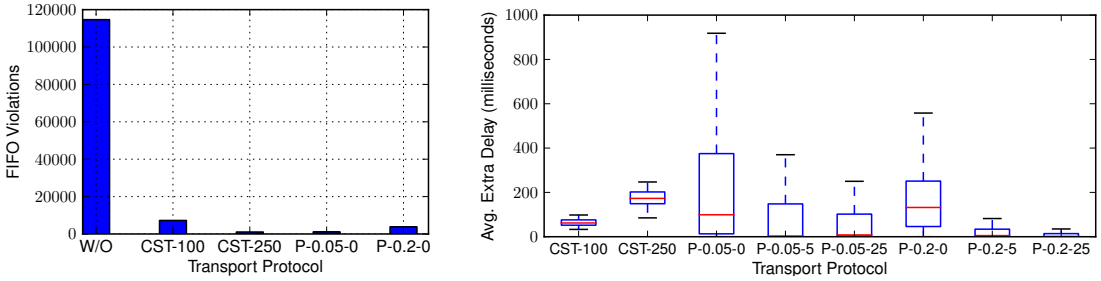
We evaluate the effectiveness of the ordering protocol through various experiments. In general, these experiments are intended to measure both the reduction in FIFO violations and the additional latency incurred by the protocol. Specifically, to characterize the trade-offs between these benefits and 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. We tested networks of 8 and 46 brokers, where each broker serves 10 clients. Here we present the results for the 46-broker network. This is a low-degree network topology with a graph diameter of 15. These and other results are reported more extensively elsewhere [11].

The results of these experiments are summarized in Figure 3. 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 3a compares the total number of FIFO violations observed 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. The results show that our probabilistic ordering algorithm is very effective. For example, with $P_t = 0.05$, the algorithm prevents 99.5% of all FIFO violations. Furthermore, as we will now show, this reduction of FIFO violations comes at a minimal cost in terms of delay.

To analyze delays, and specifically to demonstrate the benefit of the probabilistic FIFO ordering algorithm when compared to the ideal static protocol, we measure the *average extra delay*. A message m is held at the receiver waiting for missing messages until all missing messages arrive, or until the latch time expires. This is the extra delay of m. Figure 3b presents the average extra delay (over all received messages) incurred by the static and probabilistic FIFO algorithms for all of the pairs of publisher and subscriber. The box plots show the quartiles of those averages over all publisher/subscriber pairs. We first observe that the maximum extra delays incurred by the static algorithms are limited by the fixed latch time of the algorithm, whereas our adaptive protocol may incur higher delays. However, if we compare each static algorithm with the corresponding parametrization of our adaptive algorithm that achieve the same level of reduction of FIFO violations (e.g., compare CST-250 with P-0.05-*) we observe that the adaptive



(a) Number of FIFO violations

(b) Average extra delay

Fig. 3. 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.

algorithm has a better median delay even in basic mode (no publication record). Also, adding a publication record increases the advantage of the adaptive algorithm significantly. For example, with a publication record of 25 entries (P-0.05-25) the average extra delay is nearly zero for more than 50% of the nodes and less than 190 milliseconds for 90% of them.

C. Adaptivity of the protocol

Figure 4 shows the dynamics of the FIFO-ordering protocol for a publisher/subscriber pair in response to changes in publication rate. The top frame pinpoints 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 more frequently. This is because there are many cases in which the missing message does not match the subscriptions of the receiver but the ordering algorithm still latches the messages for the computed time frame.

V. 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 [12], [13], [14].

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. [15] 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 [16] 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 [17], [2], [18]. 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].

VI. 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, proba-

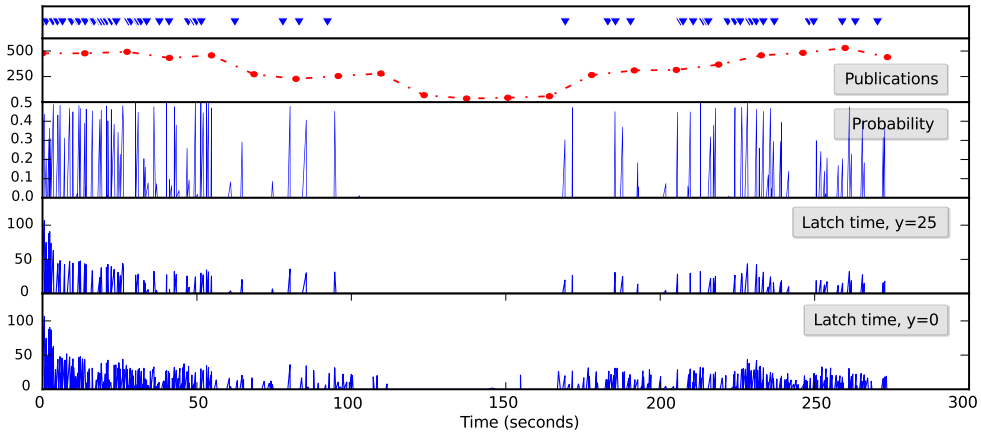


Fig. 4. From top to bottom: timestamps of out of order receptions; publication rate; probability of a FIFO violation; latch time in enhanced mode with a publication record of size 25; latch time in basic mode.

bilistic 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.

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] A. Malekpour, A. Carzaniga, G. Toffetti Carughi, and F. Pedone, "Probabilistic fifo ordering in publish/subscribe networks," Faculty of Informatics, University of Lugano, Tech. Rep. USI-INF-TR-2011-2, Apr. 2011.

[12] 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.

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

[14] 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.

[15] 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.

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

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

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