Minimizing unwanted traffic in a global messaging system: spam, denial-of-service attacks, and edacious subscribers

Dmitrii Zagorodnov
Dept of Computer Science
University of Tromsø, Norway
dmitrii@cs.uit.no

1 Introduction

One of the most attractive properties of client-initiated pull-based communication is that the consumer of the content – the one who the bits are destined for and the one who pays for their arrival – is in control of what is being sent at all times. To stop wasting time or money on the content, the user simply has to stop asking for it. Even when the user is away from the computer, only the data specifically requested (e.g. via a file sharing application) are downloaded.

Push-based communication, on the other hand, is inherently driven by the content producer. While pushing expedites delivery, it shifts to the producer the control over what data are sent and when. When this control is abused – as happens in any sufficiently large system – recipients of the *unwanted* content waste time and money. Spam and DoS attacks can be viewed as manifestations of this problem, in the sense that they both produce unwanted traffic by exploiting vulnerabilities in the design of underlying protocols. It is only prudent to ask how the new push-based paradigm – publish/subscribe messaging – fares in respect to these dangers. So far, researchers mainly focused on total access control to the messaging system [1, 2, 3, 4, 5]. While many useful solutions have been proposed, their viability in a global, Internet-scale system is uncertain at best, owing to the difficulties of deploying a global public-key infrastructure.

One additional problem, perhaps uniquely important in the pub/sub domain, appears to have been overlooked so far. Namely, the problem of users not "keeping up" with the incoming events. When we subscribe to more magazines than we have time for, some of them end up thrown away unread, even though we wanted them. While this practice is tolerated with physical and electronic mail, the widespread adoption of pub/sub messaging has the potential to escalate the amount of this *vain* traffic to unacceptable proportions. Events from content filters are of particular concern, since their arrival rate is in theory bounded only by the narrowness of the query. (The word "the" is in nearly every page indexed by Google, so presumably subscribing to new pages with this word would in theory return almost every new page on the Internet!) Even if client software can hide outdated events from the user and sanity checks can prevent "the"-queries, overall network utilization, especially in the vicinity of subscribers, has much to gain from a system-wide mechanism for minimizing vain traffic.

The main purpose of this paper is to illuminate two types of unwanted traffic in a pub/sub system – malicious (spam, DoS attacks) and vain – and suggest a general mechanism for minimizing their effects. We do this by augmenting the classic pub/sub interface with volume-limiting parameters – a combination of attributes assigned to events by publishers and thresholds specified by subscribers – and consider the implications of this interface on the unwanted traffic and on the routing infrastructure. Notably, we observe that this mechanism can minimize unwanted traffic *without* total access control if the routing substrate supports two properties: flow control and routing integrity.

We do not claim to have found the silver bullet for spam or an overflowing mailbox after a vacation, but we *do* claim that if unwanted traffic is not considered at the early design stages of large-scale pub/sub systems, we may never get the chance to fix them later.

2 Design Principles

This section presents the key principles behind our solution, based on our observations of global-scale systems of today:

Beware of cryptography. Throwing public-key cryptography at a problem is all too often the knee-jerk reaction of both researchers and developers of distributed systems in which malicious participants may be involved. Sender authentication has been touted as the panacea for email spam and approaches that rely on a public-key infrastructure are now infiltrating many pub/sub research projects. While we believe that asymmetric cryptography has important roles to play in pub/sub systems – for example, in securing communication between subscribers and local event brokers or between subscribers and publishers – we are skeptical regarding the feasibility of a global-scale PKI. The challenges of deploying one, which nobody has explicitly surmounted to date in practice or in theory, include efficient handling of certificate revocations in the face of arbitrary failures and designing a theoretically-sound and cost-effective procedure for mitigating the effects of security breaches.

Beware of routing complexity. Although powerful and appropriate for many distributed applications, content-based pub/sub systems are arguably too complex for deployment as a global one-to-many messaging substrate. Expressive power comes at a cost in scalability, so if a less expressive subscription scheme is sufficient for most applications – and we believe it is – then that simpler scheme is ultimately more viable. Furthermore, it is not clear how to protect a global content-based messaging system from abuse.

Beware of pushers. One of the key advantages of a push-based data broadcast mechanism is that the bytes are replicated on the way to the endpoints "for free." This is also its greatest danger, since it hides from the publisher the total cost of delivering the data to subscribers, which in turn shields the publisher from responsibility for content and encourages abuse. Of all properties of push-based mechanisms, *timeliness* is the only one that cannot be achieved with a pull (without incurring inordinate amount of overhead). This motivated our decision to use push only for event notification delivery, but not for event content distribution.

3 Architecture

The architecture that has emerged from these principles is that of a topic-based system with no global access control and a combination of push and pull for content distribution. Essentially, after connecting to one or more of the event brokers, any user can subscribe to a topic or advertise a new one. The system ensures that events published on the topic reach its subscribers (although, as with other such systems, to get strong delivery guarantees, a higher-level protocol would be necessary). The notifications carry meta-data, but not the content. The content is pulled by the subscriber directly from the publisher, using the address included in the meta-data.

Upon the pull, the publisher and the subscriber may chose to authenticate each other via some out-of-band mechanism, such as the ubiquitous login-over-SSL method used by many web publishers today (in which only publishers are authenticated via the PKI), but that is purely optional. Publishers would authenticate to restrict their subscriber base and subscribers would authenticate if they are concerned about forged publications. Notably, upon authenticating a publisher at a particular IP address, further content pulled from that IP does not require signatures to be deemed authentic. Putting the publisher in control of content delivery has two fundamental and interdependent implications:

- The content does not have to enter the subscriber's host until the subscriber choses to process the notifications. Thus, if events become irrelevant before then – either because they expire or they are deemed malicious (both cases will be explained below) – they disappear without a byte wasted on transmission of content. This translates into savings of bandwidth on the links near the subscriber.

- We forfeit the ability of the routing substrate to replicate data as the publication travels towards the leaves of the broadcast tree. As a result, the links near the publisher and the publisher itself see considerably more traffic (in the form of pull requests) than in a pure push-based scheme. We argue that keeping the publisher "financially responsible" for the content is a good idea. Having said that, by caching content or distributing it via swarming, the load on the publisher can be considerably reduced.

Essentially, when only a portion of notifications causes content pulls, our design relieves the subscriber of some message load and adds to the load seen by the publisher. For this to work, a mechanism to identify irrelevant events is needed. It is presented next.

3.1 Volume limiting

Every system has a maximum rate at which it can process input, and the receiving end of a pub/sub network is no exception. That limit may be imposed by hardware limitations, by a traffic quota from the ISP, or by the user explicitly requesting a cap on the number of events presented (e.g. per day). When the arrival rate exceeds that maximum rate, some events have to be dropped sooner or later. To do better than random dropping – after all not all events may be equally important – some way of prioritizing them is needed. This is our motivation for allowing the publisher to attach two *volume-limiting attributes* to every notification:

- $\mathcal{R}$*ank* – Indication of an event's importance in relation to other events on the topic.
- $\mathcal{E}$*xpiration* – Time after which an event is no longer relevant and should be discarded from the queue.

Although publishers are not required to use these two attributes and they cannot be forced to use them correctly, it is in their interest to do so to ensure that the most important events get through to the busiest subscribers. For example, on the topic of weather updates, attaching high priority to a storm warning and ensuring timely expiration of old forecasts would give an edge to the publisher utilizing $\mathcal{R}$*ank* and $\mathcal{E}$*xpiration*.

On the subscriber's end, $\mathcal{E}$*xpiration* influences the contents of the incoming queue and $\mathcal{R}$*ank* determines the order of notifications in it. To limit the size of the queue sensibly, our system offers two complementary *volume-limiting thresholds* to subscribers:

- $\mathcal{M}$*ax* – Accept at most this many highest-ranked notifications at a time. This is a *quantitative* limit.
- $\mathcal{T}$*hreshold* – Only notifications with the rank at or above this threshold are deemed acceptable. This is a *qualitative* limit.

To illustrate, if one subscribed to a news topic, the two limits used in concert would allow one to come back from a month-long vacation and read the 30 most important stories from the past month. Implications of these limits on the routing infrastructure are considered in Section 5. Although $\mathcal{R}$*ank* is useful to most publishers, it is indispensable for at least one type of publisher: the indexer, described next.

3.2 Content-based filtering

Topics restrict a subscriber to the set of publishers that contribute to them. To allow one to search for events based on their content, regardless of a topic, our system relies on *indexers*. Just as web crawlers index the content of websites to allow others to find them, an indexer parses all incoming events and notifies subscribers whose queries match the content of the event. To obtain events, the indexer can subscribe to other topics and re-publish their content; also, it can open an interface for anyone to submit their events to (not unlike users notifying a crawler of a new web page).

Subscribers, for their part, submit out-of-band queries to the indexer (in the query language of its choice), which responds with the name of a custom topic that they should subscribe to. When the first party is interested in a particular query, the indexer creates a new topic, but otherwise the subscribers are instructed to join an existing one. For example, if I ask the indexer called Eoogle to notify me when there are new events matching words "weather" and "Norway," then Eoogle may create a new topic eoogle.query.norway.weather and suggest that I subscribe to it. After doing so, I will receive all events that match that query. Other subscribers with an equivalent query will join that topic.

The volume-limiting thresholds are crucial for indexer queries, which aggregate content from multiple publishers and can effectuate an event arrival rate beyond the processing capacity of the subscriber. Web indexers return results in small chunks, ordered by relevance; thresholds can be viewed as the equivalent of that mechanism for event queries.

3.3 Feedback and ranking of content

Ranking of content, especially in the presence of malicious publishers, is the crux of an indexer. Experience with web indexers shows that using feedback from humans – in the form of links – is very effective (although not fool proof). Lacking links, event indexers need other means of getting feedback from subscribers. To that end, our system offers two feedback channels:

- *Re-publishing an article* comes closest in spirit to the idea of linking on the web. In our system a notification may reference another notification, allowing indexers to deduce the boost in the popularity of the original. For example, a blog post referenced by other blogs is an example of re-publishing.
- *Explicit feedback* from the user in the form of a rating is the ultimate and most prompt indicator of quality. As rating-enabled websites (e.g. Amazon) illustrate, many people are willing to spend the time to send feedback when they know it will improve the quality of the result. The ability to submit ratings is thus a feature of publisher's API in our system.

Publishers, including indexers, may use the feedback as they see fit. It could be used to adjust the reputation of the content author (useful for for computing ranks of future events) or it could be used to dynamically change the rank of the article itself. In the latter case, as the rank of the article changes, the publisher may send updates to the previous notification.

To limit the effects of insincere feedback, both from re-publishing and ratings, publishers may authenticate contributors (even when they do not authenticate readers) to keep track of their reputation. While there will be, no doubt, plenty of opportunities for cheating – either by defeating the authentication or by building up reputation – the publishers have the incentive to keep such problems to a minimum.

4 Malicious Traffic

We expect attempts from malicious parties to subvert the system. Traffic that results from their exploits is defined as *malicious* and falls into two broad categories distinguished by intended result: *spam* tries to reach the eyes of subscribers not interested in it and a *DoS attack* tries to slow down or disable the system or a part of it.

4.1 Spam

Topic subscriptions are spam-proof in that unsubscribing is sufficient to silence a rogue publisher. Forgeries are weeded out when the publisher is authenticated upon pulling content and thus never reach the eyes of the subscriber. Spam can be a problem, however, when multiple parties publish on the same topic. This can happen to a topic created by an indexer in response to a query, or to a topic whose publisher accepts out-of-band submissions from subscribers (this is how a bulletin board can be built with our system). The solution we envision relies on the same mechanisms that we use to limit vain content:

First, $\mathcal{M}ax$ and $\mathcal{T}hreshold$ ensure that if the publisher is successful at assigning spam notifications a lower $\mathcal{R}ank$, the chances of users seeing it are reduced. Second, even if the publisher is unsuccessful at identifying spam at first, by decoupling notification from content delivery we increase the window of vulnerability for spammers, so subscribers who notice the spam early and submit negative feedback can prevent others from pulling it (by causing the publisher to recompute the rank for the event and send an update to subscribers). Thus, volume limiting is a general technique for minimizing unwanted traffic.

4.2 Denial of Service

Pub/sub systems share DoS vulnerabilities with other network services and resolving those problems is an open research area. Of bigger concern to designers of pub/sub systems are the risks that the messaging mechanism itself introduces. In the most general sense, the question is whether a small subset of participants (subscribers, publishers, or event brokers) can use the pub/sub interface to overload or disrupt the operation of the system. There are two approaches to overloading, which an attacker may use:

- *Insert frivolous data* – such as well-formed publications, subscriptions, and advertisements – without the intention of using the system. We claim that this type of behavior fundamentally cannot be distinguished from legitimate behavior. As such, it can be restricted only as part of regulating all flows of data in the system.

- *Insert forged data* – i.e. publications or subscriptions on behalf of someone else – directing unwanted traffic towards one or more victims. Preventing this type of abuse requires some degree of integrity from the routing substrate, namely the ability to limit the propagation of messages with spoofed IDs. (More on this below.)

5 Implications on routing

Introduction of volume-limiting parameters on the edges of the network, which helps minimize vain and spam traffic, also opens doors to optimizations inside the pub/sub network. Consider the path travelled by a notification from a publisher to a subscriber. In an ideal setting, in which routing tables are unlimited and routing operations are free, the volume-limiting thresholds of the subscriber can be passed back along the path all the way to the publisher. Hence, every node in the broadcast tree can compute the maximal $\mathcal{M}ax$ and the minimal $\mathcal{T}hreshold$ for each branch below. Unwanted traffic can then be minimized within the network if every node only forwards notifications so as to satisfy the thresholds of its children. This is essentially a form of inter-node *flow control* that allows downstream nodes to put backpressure on their parents.

A realistic implementation of flow control would not have the complete global state at its disposal. Event brokers would have to rely on approximate information about downstream thresholds. Furthermore, the effective forwarding rate may be subject to overall traffic restrictions established between individual nodes: for example, a subscriber may be restricted by a contract with the border event broker; similarly, event brokers in different domains may have mutual service agreements on cross-domain traffic, etc. Precise accounting can, in theory, even make insertion of frivolous data, as described in the previous section, an unprofitable enterprise. Essentially, those who pay a fair price for bandwidth can do whatever they want with it!

Unfortunately, forged data that is allowed to propagate through the network can wreak havoc with flow control and accounting. For example, forged *publications* with high rank may "steal" room from legitimate publications inside the window established by the thresholds. With enough forgeries in the pipe, the subscriber may not receive anything at all on a topic after forgeries are filtered out by its client. Similarly, forged *subscriptions* may use up the overall traffic quota that the subscriber was granted by the border event broker. Therefore, the routing substrate needs machinery to guarantee *integrity* of the forwarding paths, which includes restricting propagation of forgeries.

6 Summary

To minimize unwanted traffic in a pub/sub system, we advocate the use of volume-limiting parameters at the edges and a combination of flow control and integrity in the routing mechanism. Although the system we sketched out is topic-based, we believe our conclusions hold for a wide range of pub/sub systems, with different routing schemes. We hope that others will join us in investigating this question.

References

[1] L. Opyrchal and A. Prakash. Secure Distribution of Events in Content-Based Publish Subscribe Systems. In *Proc. 10th USENIX Security Symposium*, pp 281–295, Washington, DC, August 2001.

[2] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems. In *Proc. 35rd Hawaii Intl Conf. on System Sciences (HICSS)*, p 303, Big Island, Hawaii, January 2002.

[3] Z. Miklós. Towards an Access Control Mechanism for Wide-Area Publish/Subscribe Systems. In *Proc. 1st Intl Workshop on Distributed Event-Based Systems (DEBS)*, pp 516–524, Vienna, Austria, July 2002.

[4] A. Belokosztolszki, D. M. Eyers, P. R. Pietzuch, J. Bacon and K. Moody. Role-Based Access Control for Publish/Subscribe Middleware Architectures. In *Proc. 2nd Intl Workshop on Distributed Event-Based Systems (DEBS)*, pp 1–8, San Diego, CA, June 2003.

[5] L. Fiege, A. Zeidler, A. Buchmann, R. Kilian-Kehr, and G. Mühl. Security Aspects in Publish/Subscribe Systems. In *Proc. 3rd Intl Workshop on Distributed Event-Based Systems (DEBS)*, Edinburgh, Scotland, UK, May 2004.