Introducing Publiy: A Multi-Purpose Distributed Publish/Subscribe System

Reza Sherafat Kazemzadeh
University of Toronto
reza@eecg.utoronto.ca

Hans-Arno Jacobsen
University of Toronto
jacobsen@eecg.utoronto.ca

ABSTRACT

We have recently witnessed widespread adoption of the publish/subscribe (pub/sub) communication paradigm in the development of large-scale distributed systems. Existing and anticipated use cases demand the pub/sub middleware to support a variety of capabilities ranging from reliability and fault-tolerance to high throughput mass-scale publication delivery, or bulk data dissemination (*e.g.*, software patch distribution). This paper introduces Publiy, our multi-purpose distributed content-based pub/sub system that features the aforementioned capabilities in its different modes of operation. Publiy is developed in Java as an open-source project.[1]

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; H.3.4 [**Systems and Software**]: Distributed systems

General Terms

Algorithms, Design, Reliability

Keywords

Publish/subscribe, message oriented middleware, content-based routing, fault-tolerance, reliability, availability

1. INTRODUCTION

Publish/Subscribe (pub/sub) middleware systems have proven instrumental in the development of many large-scale distributed applications. A pub/sub system provides a high-level, flexible, and ready-to-use solution that relieves developers from the hassle and complexities of dealing with low-level communication protocols at scale. The widespread applicability of the pub/sub model to many application scenarios implies that its middleware implementations are often expected to support a variety of different requirements. In this paper, we focus on three types of requirements: *(i)* a mission critical systems (*e.g.*, an air traffic control system) demands guaranteed event delivery despite occurrence of occasional failures and disconnections; *(ii)* an Internet-scale feed processing system (*e.g.*, a mass-scale news distribution service) requires only best-effort event delivery but places a premium on overall system scalability and throughput; *(iii)* yet another class of applications that can benefit from the pub/sub model involves push-based and selective dissemination of bulk content publications (*e.g.*, distribution of software updates based on users' software installations).

This paper introduces the Publiy pub/sub system that is developed in Java as an open-source project and supports the aforementioned requirements in different modes of operation. A distinguishing characteristic of Publiy is its adoption of the notion of *overlay neighborhoods* introduced in Section 2. Section 3 discusses the use of overlay neighborhoods in each mode of operation. Section 4 gives an overview of the Publiy's broker architecture and our live demonstration.

2. OVERLAY NEIGHBORHOODS

A distributed pub/sub system is composed of a number of application-layer routers, *a.k.a. broker*s, that form an overlay. We refer to the initial system overlay as the *primary network*. Publications are sent in this overlay towards subscribers based on their registered subscription filters. In conventional systems, a broker is only aware of its *immediate neighbors* in the primary network with whom it communicates directly. In Publiy, on the other hand, we augment brokers' knowledge of the overlay to a neighborhood that includes all nearby brokers located within a configurable distance. Using the overlay neighborhood knowledge, a broker can identify and map the primary network interconnections within its neighborhood, and also anticipate propagation path of publications that flow within its neighborhood (*i.e.*, how a publication is forwarded between nearby brokers). The following section presents how Publiy brokers use their neighborhood knowledge to achieve different goals.

3. MODES OF OPERATION

Reliability and fault-tolerance: A Publiy deployment can be configured to preserve reliability and availability and to prevent publication loss and overlay partitioning in the face of *multiple concurrent* broker crash [4] or link failures [2]. For this purpose, brokers that process publications temporarily cache the messages in an in-memory message queue and retain them until a safety condition is met. During this interval brokers are prepared to re-transmit the publications as needed. The safety condition ensures that a publication is discarded from an upstream Broker B's message queue only if all matching subscribers located within B's downstream neighborhood have successfully delivered the message, or a sufficiently large number of intermediate downstream brokers on the way towards subscribers outside of B's neighborhood have acknowledged receipt of the publication (the number of concurrent failures that the system

[1]Publiy is pronounced `/pəbli/`. The project pages are located at `http://msrg.org/project/Publiy`.

Middleware Demos and Posters '12 Montreal, Quebec Canada

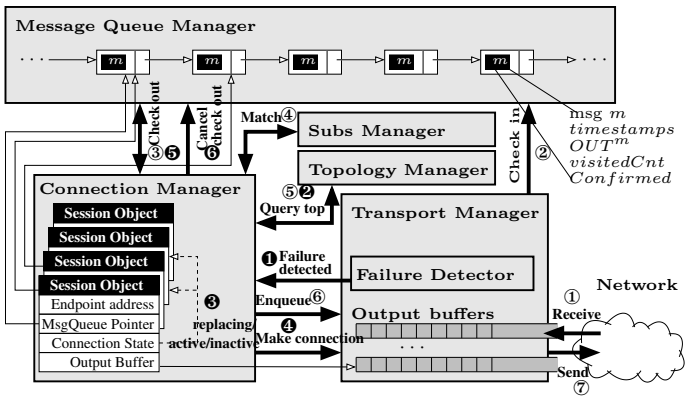

Figure 1: Internal architecture of a Publiy broker.

Figure 2: Failure timeline view (left) and topology view (right) in our command and control GUI tool.

is configured to cope with influences the number of brokers whose acknowledgements are needed in this process).

Furthermore, in order to ensure liveness of publication delivery and prevent partitioning of the primary network in the face of failures, brokers monitor their neighbors and rely on their neighborhood knowledge to reconnect the overlay after failures. Transformation of a partitioned primary network takes place in a controlled manner with no need to re-propagate registered subscriptions. This reduces the time needed to re-establish overlay forwarding paths and therefore eliminates long interruptions to the publication flows.

Opportunistic multipath forwarding: Publiy features an efficient and best-effort publication dissemination mode. The operation of conventional pub/sub systems commonly relies on setup of *end-to-end* forwarding paths between publishers and subscribers in the overlay network. Publications are sent along these paths towards matching subscribers regardless of whether they are of interest to intermediate brokers along the way, *i.e.*, whether they match a local subscriber. In content-based pub/sub systems that feature selective predicate-based subscription matching, this form of propagation results in a large number of *pure forwarding brokers* who simply relay passing publications, therefore unnecessarily consuming bandwidth and processing resources.

Publiy mitigates this problem by having brokers selectively create additional communication links, known as *soft links*, in their neighborhoods [1]. The collection of all soft links in the system constructs a highly connected overlay mesh that is superimposed atop the initial primary network. The large number of diverse end-to-end paths created in the overlay mesh gives brokers a chance to make fine-grained forwarding decisions for individual publications and opportunistically bypass nearby pure forwarders. This improves the system's efficiency, scalability and overall throughput.

Bulk content dissemination: Conventional pub/sub systems are mainly designed to disseminate small-sized publications, *a.k.a.* events. However as mentioned earlier, the pub/sub model is also applicable to usage scenarios in which publications have large chunks of payload data that involve hundreds of megabytes of content (*e.g.*, a recently released episode of a weekly TV show). We have extended Publiy with capabilities to scalably distribute bulk content using a hybrid peer-assisted approach [3]. In our design, clients register their subscriptions at pub/sub brokers and await delivery of their content of interest. Once content is released by a publishing source, brokers forward its metadata information in the pub/sub overlay and query for non-local subscribers interested in the content. This takes place in the *control layer*. Brokers then coordinate matching subscribers and *guide* them to engage in direct exchange of blocks of content. This takes place in the *data layer* and uses network coding to improve dissemination. Our hybrid peer-assisted approach is highly scalable and we have devised dissemination strategies to further enhance its performance. The combination of the strategies employed by coordinating brokers and direct peer-to-peer dissemination has proven to be effective and compared to a purely peer-to-peer solution (*e.g.*, BitTorrent) Publiy improves dissemination time by 30%.

4. BROKER ARCHITECTURE AND LIVE DEMONSTRATION

Figure 1 depicts the internal architecture of a Publiy broker. Notable components are the *message queue manager* that caches publications and implements different publication queuing strategies, the *subscription manager* that stores routing tables and incorporates a matching engine to identify matching subscriptions, the *topology manager* that maintains the broker's local view of its overlay neighborhood, and the *connection and transport managers* that handle link establishment, failure detection and overlay transformation.

Publiy's source code distribution comes with an integrated command and control GUI tool (depicted in Figure 2) which can be used to deploy a pub/sub overlay and execute a visualized *failure timeline* (a series of timestamped failure injections and recoveries). Our tool also features other capabilities such as overlay composition and live monitoring.

In our live demonstration, we present Publiy's reliable and fault-tolerant mode of operation. We execute a failure timeline and visualize how the pub/sub overlay transforms after injecting failures and how reliability of publication delivery is preserved despite multiple concurrent failures.

5. REFERENCES

[1] R. S. Kazemzadeh and H.-A. Jacobsen. Opportunistic multipath forwarding in content-based publish/subscribe overlays. In *Middleware '12*.

[2] R. S. Kazemzadeh and H.-A. Jacobsen. Partition-tolerant distributed publish/subscribe systems. In *SRDS '11*.

[3] R. S. Kazemzadeh and H.-A. Jacobsen. Publiy$^+$: A peer-assisted publish/subscribe service for timely dissemination of bulk content. In *ICDCS '12*.

[4] R. S. Kazemzadeh and H.-A. Jacobsen. Reliable and highly available distributed publish/subscribe service. In *SRDS '09*.