Soft State in the XSiena Publish/Subscribe System

Zbigniew Jerzak*
Dresden University of Technology
Systems Engineering Group
01062 Dresden, Germany

Christof Fetzer
Dresden University of Technology
Systems Engineering Group
01062 Dresden, Germany
Christof.Fetzer@tu-dresden.de

Robert Wójcicki
Silesian University of Technology
Institute of Informatics
44100 Gliwice, Poland
Robert.Wojcicki@polsl.pl

ABSTRACT

This paper presents our experiences with building of the soft state XSiena publish/subscribe system. We provide a brief overview of our approach towards the soft state in publish/subscribe systems along with the detailed discussion of the liveness and safety issues as well as the choice of the API and handling of the unsubscription and unadvertisement messages.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms

Algorithms, Design

1. INTRODUCTION

Every node in a traditional publish/subscribe system stores a certain amount of hard state [6]. Publishers, producing data, store the advertisements summarizing the content to be produced. Subscribers, willing to receive information, store filters which describe their interest. Loosely coupled brokers interconnecting publishers and subscribers maintain both advertisements and filters until a corresponding unadvertisement or unsubscription arrives.

A soft state approach [6] assumes that filters and advertisements are maintained in a hard state fashion only at the endpoints of network, i.e., by publishers and subscribers. Brokers use a lease-based approach [4] instead: every filter and advertisement is assigned a validity interval timer T_I which states for how long after the reception of the given message should it be kept by the broker. The validity interval timer is set by the publisher (in case of advertisements) or subscriber (in case of filters) and it is decremented by the broker starting from the moment of the respective message reception. When the validity interval timer of the message m reaches zero, message m is removed by the broker from the broker's state.

In order to prevent the expiration of filters and advertisements publishers and subscribers reissue respective messages before the validity interval timer T_I expires. Brokers upon the repeated reception of the same message reset the value of the validity interval timer to the one carried by the new message, thus effectively extending the lease of the message stored in their soft state.

Our approach differs from the previous work [4], in that we use the Timed Asynchronous Distributed System Model [1] (TADSM) as the foundation for our system. The TADSM does not make any assumptions regarding the bounds on the communication latencies nor the number of failures in the system. Instead, we use the fail-aware publish/subscribe system design [2] to compensate for the varying latencies on the communication links.

We implement the soft state approach on top of the XSiena[1] content-based publish/subscribe system (part of the FP7-216181 STREAM[2] project) developed in our group. The evaluation of the soft-state approach is performed in the PlanetLab environment. In the remainder of this paper we highlight noteworthy design issues which were raised during the implementation of the soft state XSiena system.

2. LIVENESS AND SAFETY

The correctness of distributed systems is often specified in terms of liveness and safety. The definitions of those properties in context of publish/subscribe systems [3] are given below.

The **safety property** states that: ($s1$) an event e is delivered to the subscriber **S** at most once and ($s2$) the subscriber **S** only receives events which have been previously published and ($s3$) the subscriber **S** only receives events for which it has subscribed.

The **liveness property** states that: subscriber **S** which subscribed with filter f and did not issue an unsubscription message for that filter, will eventually receive every event e which had been published and matches f.

The soft state XSiena system satisfies the liveness property. Moreover, following [4] it can be said that the soft state

*Author partially supported by the Polish Ministry of Science and Higher Education grant number N N516 375034. Author is currently with the SAP Research (CEC Dresden) and can be reached at zbigniew.jerzak@sap.com

[1] http://wwwse.inf.tu-dresden.de/xsiena
[2] http://www.streamproject.eu/

```
1 public interface XSiena {
2   void publish(Event e);
3   void subscribe(Filter f, Deliverable d);
4   void advertise(Filter f);
5 }
```

Listing 1: The soft state XSiena interface

```
1 public interface EXSiena extends XSiena {
2   void unsubscribe(Filter f, Deliverable d);
3   void unadvertise(Filter f);
4 }
```

Listing 2: The extended XSiena interface

XSiena system satisfies a modified version of the safety property – the **eventual safety property**, defined as: starting from an arbitrary state the system eventually satisfies the safety properties $s1$, $s2$ and $s3$.

However, the soft state XSiena system, unlike systems proposed in [4], does not (and cannot) guarantee an upper bound on the time for which system remains in the incorrect state. This is a direct implication of the unbounded network delays in the TADSM – the underlying model for the soft state XSiena system. Instead, it can be said that the soft state XSiena system achieves the eventual safety property assuming that: (1) network delays do not expose an infinite growth and (2) the size of the network does not expose an infinite growth either.

3. SOFT STATE API

The nature of the soft state publish/subscribe system determines the exposed API. The API presented in the Listing 1 is a direct result of the properties of the soft state XSiena system. Presented API follows the general publish/subscribe API suggested in [5]. The subscribe operation includes as parameters: (1) a new filter `CFilter` and (2) a callback interface `Deliverable`. The `Deliverable` interface is implemented by the application programmer in order to receive events matching the issued filters.

However, unlike the API proposed in [5] the validity of the filters and advertisements is not explicitly included in the corresponding `advertise` and `subscribe` API calls. In soft-state XSiena system the validity interval T_I is part of the filter and advertisement messages and can be manipulated via the `Filter` API. Specifically, the validity interval T_I is not a subject to the content-based matching. It is the opinion of the authors that such design is more natural for the soft state publish/subscribe systems, treating the validity of the filter or advertisement as an integral part of the respective message.

4. EXPLICIT STATE REMOVAL

An important aspect of the soft state XSiena system is the lack of unadvertisement and unsubscription methods in the API. Such methods are strictly speaking not necessary, as every advertisement and filter will expire within its validity interval T_I plus the broker dependent extension. However, it is possible to provide an extended API to the brokers (see Listing 2) which allows to unadvertise or unsubscribe messages prior to the expiration of their validity intervals. It is important to stress that soft state brokers do not rely on those calls, which are only a means to speed up removal of filters and advertisements with large validity intervals.

Another, related, design decision is made with respect to the removal of filters and advertisements for which the validity interval expires. Such cases are handled by the brokers via the broker internal, private unsubscribe and unadvertise methods. An important aspect of those methods is that their effects are never propagated outside of the brokers. If, for example, a filter f_1 covering filter f_2 expires than fact of the deletion of the filter f_1 and uncovering of the filter f_2 is never propagated to other downstream brokers. Even though the propagation of this information would be a correct action from the perspective of the soft state XSiena system and could increase the responsiveness of the publish/subscribe system it has been chosen not to implement this behavior.

It is important to note that the correct routing information will be eventually set up with the propagation of the refresh messages for the filter f_2. Our design decision was dictated by the fact that it is desirable to avoid the storm of unsubscription and unadvertisement messages propagated throughout the publish/subscribe network whenever a filter or advertisement expires due to the high variance in the propagation delays of the refresh messages.

5. ADDITIONAL AUTHORS

Additional authors: Grzegorz Hryń (Fluor Corporation, Prymasa Stefana Wyszyńskiego 11, 44101 Gliwice, Poland, email: `Grzegorz.Hryn@fluor.com`)

6. REFERENCES

[1] Flaviu Cristian and Christof Fetzer. The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):642–657, June 1999.

[2] Zbigniew Jerzak, Robert Fach, and Christof Fetzer. Fail-aware publish/subscribe. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pages 113–125, Cambridge, MA, USA, July 2007. IEEE Computer Society.

[3] Gero Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Technisches Universität Darmstadt, 2002.

[4] Gero Mühl, Michael A. Jaeger, Klaus Herrmann, Torben Weis, Andreas Ulbrich, and Ludger Fiege. Self-stabilizing publish/subscribe systems: Algorithms and evaluation. In *Euro-Par 2005 Parallel Processing*, volume 3648/2005, pages 664–674. Springer Berlin / Heidelberg, 2005.

[5] Peter Pietzuch, David Eyers, Samuel Kounev, and Brian Shand. Towards a common api for publish/subscribe. In *DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 152–157, Ontario, Canada, 2007. ACM.

[6] Suchitra Raman and Steven McCanne. A model, analysis, and protocol framework for soft state-based communication. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 15–25, New York, NY, USA, 1999. ACM.