A Scheme for Reliable Delivery of Events in Distributed Middleware Systems

Shrideep Pallickara and Geoffrey Fox
Community Grids Lab, Indiana University
(spallick,gcf)@indiana.edu

Abstract

Increasingly interactions that services and entities have with each other, and among themselves, are network bound. These interactions can be encapsulated in events. We describe a scheme for the reliable delivery of events in the presence of link and node failures.

1. Introduction

Events can encapsulate, among other things, information pertaining to transactions, data interchange, system conditions and finally the search, discovery and subsequent sharing of resources. The routing of these events is generally managed by a middleware. In this paper we describe our scheme for the reliable delivery of events within NaradaBrokering [1, 2], which is a distributed messaging middleware supporting a variety of event driven interactions – from P2P interactions to audio-video conferencing applications. This reliable delivery guarantee holds true in the presence of four conditions.

1. Broker and Link Failures: The delivery guarantees are satisfied in the presence of individual or multiple broker and link failures. The entire broker network may fail. Guarantees are met once the broker network (possibly a single broker node) recovers.
2. Prolonged Entity disconnects: After disconnects an entity can retrieve events missed in the interim.
3. Stable Storage Failures: The delivery guarantees must be satisfied once the storage recovers.
4. Unpredictable Links: Events can be lost, duplicated or re-ordered in transit over individual links.

The remainder of this paper is organized as follows. Section 2 describes the reliable delivery scheme with section 3 outlining our augmentation to GridFTP. We include experimental results in section 4. Finally in section 5 we outline our conclusions and future work.

2. The reliable delivery scheme

To ensure the reliable delivery of events (conforming to a specific template) to registered entities three distinct issues need to be addressed. First, there should be exactly one Reliable Delivery Service (RDS) node that is responsible for providing reliable delivery for a specific event template. Second, entities need to make sure that their subscriptions are registered with RDS. Finally, a publisher needs to ensure that any given event that it issues is archived at the relevant RDS. In our scheme we make use of both positive (ACK) and negative (NAK) acknowledgements. We may enumerate the objectives of our scheme below.

- Storage type: Underlying storages could be based on flat files or relational/XML databases.
- RDS instances: There could be multiple RDS instances. A given RDS instance can manage reliable delivery for one or more templates.
- Autonomy: Individual entities can manage their own event templates. This would involve provisioning of stable storage and authorization of entity constraints.
- Location independence: A RDS node can be present anywhere within the system.
- Fast Recovery schemes: The recovery scheme needs to efficiently route missed events to entities.

2.1. The Reliable Delivery Service (RDS)

RDS can be looked upon as providing a service to facilitate reliable delivery for events conforming to any one of its managed templates. To accomplish this RDS provides four very important functions. First, RDS archives all published events that conform to any one of its managed templates.

Second, for every managed template, RDS also maintains a list of entities for which it facilitates reliable delivery. RDS may also maintain information regarding access controls, authorizations and credentials of entities that generate or consume events targeted to this managed template. Entity registrations could either be user controlled or automated.

Third, RDS also facilitates calculation of valid destinations for a given template event. This is necessary since it is possible that for two events conforming to the same template, the set of valid destinations may be different. RDS maintains a list of the profiles and the encapsulated constraints (subscriptions) specified by each of the registered entities. For each managed template the service also hosts the relevant matching engines, which computes entity destinations from a template event's *synopsis* (content descriptors).

Finally, RDS keeps track not only of the entities that are supposed to receive a given template event, but also those entities that have not explicitly acknowledged receipt of these events.

RDS also archives information pertaining to the addition, removal and update of constraints specified by registered entities. For every archived event or entity profile related operations, RDS assigns monotonically increasing sequence numbers. These sequence numbers play a crucial role in error detection and correction, while also serving to provide audit trails.

Publishing entities make use of *companion events* and a series of *negotiations* to ensure delivery of published events to the relevant RDS. Publishing entities need information regarding generation of companion events. This information is maintained both at the publishing entity and RDS. During recovery this information can be retrieved from RDS.

Entities within the system use *invoice events* (which can encapsulate both ACKs and NAKs) to detect and fix errors in delivery sequences. Associated with every entity within the system is an *epoch*. This epoch is advanced by RDS and corresponds to the point up until which that entity has received events. Since the epoch can be used to determine order and duplicate detection it is easy to ensure guaranteed exactly once delivery of events.

3. Applications

We have augmented (details in Ref [3]) GridFTP to exploit this scheme. Here, we had a proxy collocated with the GridFTP client and the GridFTP server. This proxy, a NaradaBrokering entity, utilizes NaradaBrokering's fragmentation service to fragment large payloads (> 1 GB) into smaller fragments and publish fragmented events. Upon reliable delivery at the server-proxy, NaradaBrokering reconstructs the original payload from the fragments and delivers it to the GridFTP server.

4. Experimental Results

Our experiments involved 3 brokers (see Figure 1). We compared the performance of NaradaBrokering's reliable delivery algorithms with its best effort approach. For best effort, all entities/brokers within the system communicate using TCP, while in the reliable delivery approach we setup communications to be based on UDP. The publishing/subscribing entities, brokers and RDS are all hosted on separate machines (1GHz, 256MB RAM) with each process running in a JRE-1.4 Sun VM. Currently, RDS supports flat-file and SQL based archival. The results reported here correspond to RDS utilizing MySQL–4.0 for storage operations. We found that archival overheads were between 4-6 milliseconds for payloads varying from 100b–10 KB. We have computed the delays associated with the delivery of events in the best-effort and reliable delivery schemes.

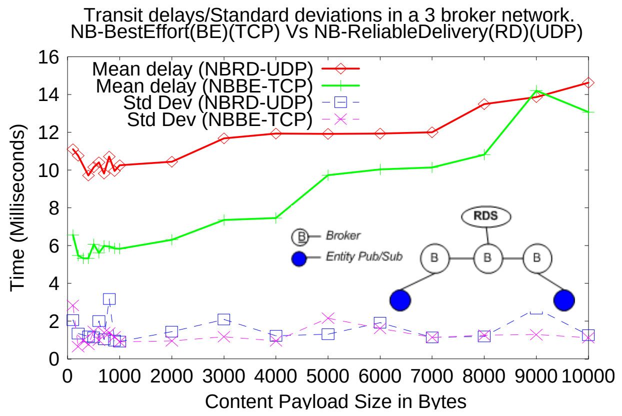

Figure 1: Experimental Setups

Figure 1 depicts our results. In the reliable delivery case there is an overhead of 4-6 milliseconds (depending on payload size) associated with the archival of the event, with an additional variable delay of 0-2 milliseconds due to *wait()-notify()* statements in the thread which triggers archival. These factors, in addition to retransmissions triggered by the subscribing entity, due to lost packets, contributed to higher delays in the reliable delivery case. Note that we can have an optimistic delivery scheme which does not wait for archival notifications prior to delivery; this would be even faster.

5. Conclusions & Future Work

In this paper we described our scheme for the reliable delivery of events. We are currently incorporating support for WS-ReliableMessaging [4] within NaradaBrokering.

6. References

[1] NaradaBrokering Project http://www.naradabrokering.org

[2] Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of ACM/IFIP/USENIX International Middleware Conference. 2003.

[3] G. Fox, S. Lim, S. Pallickara and M. Pierce. Message-Based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services. (To appear) Journal of Future Generation Computer Systems.

[4] Web Services Reliable Messaging Protocol. March 2003. From IBM, Microsoft.