Reliable Routing of Event Notifications over P2P Overlay Routing Substrate in Event Based Middleware

Shruti P. Mahambre and Umesh Bellur
KReSIT, IIT Bombay,
Mumbai, India
{shruti,umesh}@it.iitb.ac.in

Abstract

Event broker networks (EBN) are a scalable incarnation of the publish subscribe paradigm for building asynchronous systems. These take the form of overlays of broker nodes and several routing schemes exist that deliver events from publishers to subscribers efficiently on different overlay structures. However qualities of service based routing schemes are rare and our work addresses this gap. Specifically we look into the prospect of routing events based on reliability requirements of subscribers for an event type being delivered via the EBN. In this paper, we propose a multiplicative model to measure reliability of the P2P overlay routing substrate and an algorithm based on this model, to deliver event notifications to the client. We employ a technique called 'pruning' by which we restrict flooding the entire overlay routing substrate, when finding a reliable path. The complexity analysis of our algorithm shows that it finds a reliable path with a lower message complexity, as compared to the flooding approach. Our algorithm also determines a path with higher reliability than the path established by Hermes [6]. We present initial simulation results, using the Hermes middleware simulator.

1 Introduction

Event based middleware is a powerful and scalable paradigm, capable of building large scale distributed systems. It extends the notion of publish-subscribe, and also facilitates the construction of an overlay routing substrate over the publish-subscribe brokers. Overlay routing substrates, provide decentralized, application-level routing, that helps disseminate events from publishers to subscribers. Middleware alleviates the heterogeneity of large scale distributed applications, by enabling interoperability amongst its components.

In an event based middleware, information provided by the producers is distributed in a timely manner to the consumers. A common service interface provided by event based systems is the publish-subscribe (pub-sub) [11]. In a pub-sub paradigm, the producers publish information, consumers subscribe to this information, and receive notifications for the same. This information is encapsulated in a structure termed as the event. This model relies on an event notification service, provides storage and management for subscriptions and efficient delivery of events. There is anonymity amongst both, the senders and the receivers, and the communication between them is asynchronous. The fully decoupled nature in terms of time, space and synchronization [11], makes this model highly suitable for large scale distributed applications.

![Figure 1. Layers in the Broker Network Architecture](image-url)

Pub-Sub systems are deployed on overlay networks. An overlay network [7] constructs a user level graph on top of an existing networking infrastructure, such as the Internet, using a subset of the available network links and nodes. An overlay link is a virtual link in this graph and may consist of many actual links in the underlying network, i.e., a single hop in the overlay network, could result in multiple hops in the physical network. Overlay nodes run applications in
a distributed manner providing middleware functionalities. Most event based middleware are implemented using the two layers, as shown in Figure 1. The bottom layer represents the physical network comprising of routers and links. The overlay network layer represents the P2P overlay routing substrate which provides an abstraction of the distributed hash table [8]. A node in this P2P overlay routing substrate, may be mapped to an event broker, which forms a part of the event dissemination tree, used for routing events.

The main contribution of this paper, is a multiplicative model that is proposed to measure reliability of a P2P overlay routing substrate, and an algorithm based on this model, to reliably route event notifications to client. The paper is organized as follows. In Section 2, we provide a formal definition of reliability over a P2P routing substrate in a publish-subscribe domain. We compare our work with existing work in this field in Section 3. Section 4 provides a formal definition of the problem. In order to determine reliable paths for routing event notifications, we propose a multiplicative model for reliability measurement in Section 5. In Section 6 we propose a Pruning algorithm, which relies on the reliability model and finds a reliable path on the overlay routing substrate. We compare our algorithm, with a flooding algorithm in Section 6.2 and present the complexity analysis of our algorithm in Section 7. We present initial simulation results, in Section 8 that verify our claims and conclude with a discussion regarding future work in Section 9.

2 Reliability in Publish Subscribe

Quality of Service [4] or QoS refers to the ability of a system to provide network and computation services, such that the users expectations for all non-functional criteria, such as timeliness and performance quality are met. Some of the quality of service parameters, which an event based middleware may support are minimum bandwidth requirement, order of event notifications, reliable delivery of event notifications, delivery semantics of an event, latency and availability.

In this paper our focus is on measuring reliable delivery of event notifications. The delivery reliability of an event, in the context of a pub-sub system, is defined as the ratio of the number of notifications delivered to the subscriber of the event type, against the number of publications of an event type. In a type based scheme, event subscriptions are filtered based on their types. The notion of an event kind, is matched to an event type [3]. Such a model facilitates type safety at compile time and leads to closer integration of the language and the middleware [11]. We run our simulations on the Hermes middleware, which provides type-based [6] and type-and-attribute-based [6] routing of events.

We now provide a formal definition for reliability, in the context of a publisher-subscriber system.

**Formal Definition of Reliability in a Publisher-Subscriber Model:**

Assume that \( n(e_r) \) represents the number of notifications sent for an event type \( \tau \) and \( p(e_r) \) represents the number of publications of an event of type \( \tau \). The reliability \( R \) attained at subscriber \( s \) is given by 1.

\[
R[s(e_r)] = \frac{n(e_r)}{p(e_r)}
\]

Using equation 1, reliability is measured at different levels in an event based system as follows:

2.1 Per Subscriber Per Event Type

The delivery reliability for the notification of an event type, is measured for a single subscriber subscribing for a single event type. A subscriber may subscribe for multiple event types. We can measure the reliability that the subscriber attains for each event type that it has subscribed for, as shown in equation 1

2.2 Per Subscriber All Event Types

A subscriber can measure the total reliability it has attained, by calculating the reliability it attains across all the event types for which it has subscribed. The reliability attained by a subscriber for all the event types \( e(\tau_j) \), it has subscribed for, is given by equation 2

\[
R[s(e_{\tau_j})] = \frac{\sum_{j=1}^{\tau_{max}} R[s(e_{\tau_j})]}{e_{\tau_{max}}}
\]

2.3 All Subscribers for an Event Type

The reliability offered by the entire system for a specific event type, can be calculated across all the subscribers, who have subscribed for that event type. The reliability offered by the system for an event type \( \tau \), across all the subscribers \( s_i \) is given by 3

\[
R[s_i(e_{\tau})] = \frac{\sum_{i=1}^{s_{max}} R[s_i(e_{\tau})]}{s_{max}}
\]

2.4 Entire System

The reliability of the entire system, is defined as the reliability attained across all the subscribers and for all event
types, they have subscribed for. Using equations 2 and 3 the reliability of the entire system is as given by equation 4

\[
R[s_i(\tau_j)] = \sum_{i=1}^{s_{max}} \left[ \sum_{j=1}^{t_{max}} R[s_i(e_{\tau_{ij}})] / (t_{max}) \right] \tag{4}
\]

3 Related Work

TAROM [10] finds a secondary overlay path that minimizes the joint failure probability for a given primary overlay path between a source and destination. It performs path probing to obtain loss rates in overlay paths and then applies random sampling to estimate the link loss rates. The alternative paths to two independent (i.e. paths between pairs of two non-interacting sources and destinations) primary paths may interfere. If both the primary paths fail simultaneously, the alternative paths are used and hence the common links between the alternative paths may get excess traffic leading to high losses (both in links and at corresponding nodes). Also TAROM is invoked "on demand", i.e., when a node sees the need to establish multiple paths. It is not tied to an event model, where factors such as the arrival rate of event publication, and number of events in the system are relevant. TAROM uses the secondary overlay paths only as backup paths. This approach also does not consider the node failure probabilities which may be crucial as the message queues at each node are of finite length and hence significant losses can occur at overlay nodes with high incoming traffic.

RON [2] nodes also use active probing to infer quality of virtual links, and passive observation of the traffic, and then propagate this information to other nodes. Each node uses a variety of performance metrics, and an application-specific path is selected on the basis of these metrics.

Both RON and TAROM do not take into account node failure probabilities. We propose a multiplicative reliability model, which considers both, the node and the link reliabilities when establishing reliable paths in an overlay network. Unlike TAROM, we establish secondary paths in order to achieve higher reliability in the network, by increasing redundancy, and not merely for establishing backup paths. If the primary path (i.e. the path established by the middleware) is reliable, then we send notification along the primary path to the subscriber.

Hermes reliability model has two aspects - the robustness of the middleware against failures and the reliability that is explicitly demanded by clients through a quality of service(QoS) specification. Hermes uses Pastry [9] which is a routing substrate for wide area P2P applications. Pastry performs object location, and application level routing in an overlay network. Each node in Pastry has a unique numeric node identifier, and assuming a network of \(N\) nodes, Pastry can route a message in \(\log N\) steps. Hermes routing algorithms are built over Pastry and they use built-in fault-tolerance features which enable event brokers to recover to a consistent system state after failure. However Hermes does not provide support for reliability specified by the client as a service guarantee. We focus on the second aspect of the reliability model of Hermes, i.e., providing reliability, as a QoS requirement specified by the client.

4 Problem Statement

- \(p_i\) is a publisher \(i\), s.t. \(p_i \in P\), where \(P\) is the set of publishers and \(s_i\) is a subscriber \(i\), s.t. \(s_i \in S\), where \(S\) is the set of subscribers.
- \(\lambda\) is the reliability threshold requested by the subscriber for an event it has subscribed for.
- \(\tau\) is an event type
- \(p_i(\tau)\) is a publisher \(i\), s.t. \(p_i \in P\), which has published an event of type \(\tau\) and \(s_i(\tau)\) is a subscriber \(i\), s.t. \(s_i \in S\), which has subscribed for an event of type \(\tau\)
- \(n_i\) is a node \(i\), s.t. \(n_i \in N\), \(\forall i = 1..N\), where \(N\) is the set of brokers in the overlay network and \(l_{i,j}\) is a link between nodes \(n_i\) and \(n_j\), s.t. \(l_{i,j} \in L\), where \(L\) is the set of links in the overlay network.
- \(\Theta\{n_i,l_{ij}\}\) is a set of nodes and links on a reliable path established between a publisher and subscriber for an event type.
- \(R[\Theta\{n_i,l_{ij}\}]\) is the reliability of the path

Given a single subscriber \(s_i(\tau)\), single publisher \(p_i(\tau)\) and \(\lambda\), we determine a path \(\Theta\{n_i,l_{ij}\}\) s.t.

\[
R[\Theta\{n_i,l_{ij}\}] \geq \lambda \tag{5}
\]

5 Multiplicative Model for Reliability

In order to compute reliability of a path, we consider, the path of an event notification to be a series system [12], in which all components (broker nodes) are so interrelated that the entire system fails if any one of its components fail.

Consider a system with \(n\) components, i.e., \(n\) broker nodes. All these nodes lie in the path from the subscriber to the publisher for an event type. Let \(\omega_i\) be the blocking threshold \(^1\) at node \(n_i\) and \(\phi_i\) be the link loss occurring at

\(^1\)Blocking threshold refers to the threshold beyond which events are dropped from the node buffer
link \( l_{ij} \). The probability that the node \( i \), will drop a packet (i.e. the blocking probability) is given by \( P(\omega_i) \). Hence the probability that the link is reliable is \((1 - P(\omega_i))\). The probability that the link \( i \), will drop a packet (i.e. the link loss) is given by \( P(\phi_i) \). Hence the probability that the link is reliable is \((1 - P(\phi_i))\). The reliability of the entire path is given by the following equation

\[
R[\Theta\{n_i, l_{ij}\}] = \prod_{i=1}^{N}[1 - P(\omega_i)][1 - P(\phi_i)] \tag{6}
\]

Any algorithm can rely on this model, to find a reliable path in an overlay network. The broker network is responsible for ensuring the reliability requirement specified by the subscriber. In the next section we discuss the proposed algorithm in detail.

6 Our Approach

In this section, we propose an algorithm which leverages the Pastry [9] routing algorithm, based on the reliability measurement model proposed in Section 5. Given a threshold value for reliability, by the subscriber, our algorithm tries to determine the most reliable path\(^2\), for delivery of an event notification, with minimum possible message complexity. We can determine a highly reliable path in the overlay network, by flooding the entire network. This results in a high message complexity. The Pastry overlay substrate has the property that it is able to route a message to a given destination in maximum \( \log N \) hops, where \( N \) is the number of nodes in the overlay network. We use this property of the Pastry overlay routing substrate, in order to control the message complexity introduced by the flooding approach. We modify the Pastry routing algorithm to determine a set of paths, along with their reliabilities and send an event notification only along paths which meet the reliability requirement of the subscriber. The Pruning Algorithm described in Section 6.1, is discussed at the level of a single event broker, and is applicable to every broker in the reliable paths being established.

6.1 The Pruning Algorithm

The Pruning algorithm is divided into three stages. Initially the Hermes middleware determines a path for sending an event notification. This path does not take into consideration the reliability requirement of the subscriber. Once this path is established, the next stage of the Pruning algorithm, involves partial flooding, followed by Pastry routing. We will now describe algorithm in 1 in detail.

- **stage 1**: We determine the primary path for event delivery. Primary path is the path along which the event notification travels by default using the Hermes middleware routing algorithm [6]. However instead of sending a notification to the subscriber, we return the identifier of the destination node to the event publisher source, along with the reliability value of the path. This is done, since it is possible that the primary path, may not be meeting the reliability threshold specified by the subscriber.

- **stage 2**: With this we ensure that the source is aware of the destination. Using this information, we apply the pruning algorithm to establish reliable paths between the source and destination. In step 1 of algorithm 1, we get the set of neighborhood nodes [9] for the current node, which is being flooded and perform partial flooding (i.e., flooding up to a particular level only and not the entire network). Flooding enables the establishment of multiple paths, and increases the chances of finding a path which has a reliability value meeting the threshold reliability requirements of the subscriber. In **stage 3** we will discuss a technique in which we combine multiple paths in order to meet the threshold reliability requirement of the subscriber, if no single path exists, which has a reliability value above the threshold. In step 3 we check to see if the node being flooded already exists in the path that has

---

\(^{2}\)Most reliable path is one which has a reliability value greater than or equal to the threshold specified by the subscriber.
been established so far. If the current node is in the list, then we do not add it to the list once again. This prevents cycles. In step 4, we obtain the node blocking probability for a node which is occurring on the path for the first time, and in step 5, we obtain the link loss rate. We calculate partial path reliability as given in the formula in step 6. This node now becomes a part of the partial path established so far (steps 10, 11). If a neighborhood node is not in the partial path established so far, it implies that the neighborhood node has not yet been flooded. We continue flooding, till the level of flooding that has been specified, and recursively call the Pruning algorithm (step 14).

• stage 3:- Once the flooding is complete, the message is then routed to the destination using Pastry Routing (step 15). The PartialPath routine, returns the partial path, up to the point from where we need to resume flooding (step 15). Once all the paths are established, the set of reliable paths, along with their reliability values is sent back to the publisher source node, using Pastry routing as shown in step 20. Once the reliable paths have been determined, the source node, takes a decision and sends notification along a route, whose reliability is in conformance with the reliability requirement of the subscriber. The SendNotification routine in step 21 does the following:

stage 3a:- Find a set of paths, having reliability values, which are greater than or equal to the threshold reliability requirement specified by the subscriber. Send the notification along the path having maximum reliability value, from this set.

stage 3b:- If no single path exists which meets the reliability requirement of the subscriber, then we combine paths in the system, such that the reliability of the combined paths, is greater than or equal to the threshold. Notification is then sent in parallel, along all the combined paths. We use the greedy approach when selecting paths, i.e., we select paths in descending order of reliability values. Two cases arise out of this scenario.

Case 1: Disjoint Paths - In this case all the paths being combined are disjoint, i.e., they do not have any common nodes and links. If \( R \) is the reliability of path \( \Theta_i \), and there are \( H \) paths which need to be combined in order to attain a reliability value greater than threshold, then the combined reliability of the parallel paths \( R_c \) is given by equation 7.

\[
R_c = 1 - \prod_{h=1}^{H} (1 - R[\Theta_h \{n_i, l_{ij}\}]) \quad (7)
\]

Case 2: Intersecting Paths - Here the paths being combined are non-disjoint, i.e., subsets of the paths, intersect with each other. Given \( R \) is the reliability of path \( \Theta_i \), and there are \( H \) paths which need to be combined in order to attain a reliability value greater than threshold, we obtain a union of all the paths, and calculate the reliability as shown in equation 8.

\[
R_c = \sum_i R[\Theta_i] - \sum_i \sum_{j > i} R[\Theta_i \cap \Theta_j] + \sum_i \sum_{j > i} \sum_{k > j} R[\Theta_i \cap \Theta_j \cap \Theta_k] - \ldots
\]

\[
+ (-1)^{H+1} R[\Theta_1 \cap \Theta_2 \cap \Theta_3 \ldots \cap \Theta_H]
\]

We use an incremental approach when calculating multipath reliability, i.e., we keep on adding one path at a time to the existing set of paths, till we attain the reliability specified by the subscriber. At each step, we store reliabilities of the paths which have been combined so far. Therefore, when we add a new path, we only need to calculate the joint reliability of the new path with the existing ones as shown in equation 8. Since Pruning restricts the level of flooding in the system, we compare our approach with the Flooding approach, which we discuss in the next section.

6.2 The Flooding Algorithm

In the flooding algorithm [5], we send messages to all nodes occurring in the neighborhood set [9] of the current node. This process is continued till we reach the destination. The flooding approach differs from Pruning, since it does not restrict flooding to a particular level. Every node is flooded, till the destination node, occurs in the leaf set [9] of a node. The procedure for calculating reliability of the path, and sending notification, is similar to that discussed in the Pruning algorithm in Section 6.1.

7 Complexity Analysis of Pruning Algorithm

In this section, we determine the message complexity for the Pruning algorithm, and discuss its implications.

• The Hermes middleware determines a path for notification of the event, using the Pastry routing algorithm,
which we call as the Primary Path. This introduces a message complexity of $2(\log N)$, assuming that the network consists of \(N\) nodes. The messages generated are twice \((\log N)\), because, once the primary path is established, the destination node collects the reliability value of the path and routes it back to the source, using Pastry routing.

- If \(n\) is the number of neighborhood nodes for each broker in the network, and \(k\) is the level to which we flood, then $\sum_{i=1}^{k} n^i$ indicates the total number of messages introduced in the network, when flooding. And for each of the paths established when flooding, we use Pastry routing to route message to destination. Therefore, the total number of messages introduced in the system currently are \((\log N)n^k\). An additional \((\log N)\) messages are introduced, to route the reliability values back to the source using Pastry Routing algorithm.

- Once the paths are established, the notification is sent along the paths, which meet the reliability requirement of the subscriber. The maximum number of paths along which the event notification can be sent are \((\log N)n^k\) (in case of multipath routing).

The total number of messages \(M\) generated in the system, when the pruning algorithm is executed, is given by the following expression.

\[
M = 2(\log N) + \sum_{i=1}^{k} n^i + (\log N)n^k + (\log N) + (\log N)n^k
\]

We can infer that, the message complexity of the Pruning approach is of \(O((\log N)n^k)\).

We find that the message complexity is influenced by the number of neighbors for each node, and the level of flooding in the network. So we see that, as the levels of flooding in the Pruning algorithm increase, the Pruning algorithm, will have a message complexity similar to that introduced in the network by flooding the entire network. We verify our claims with simulation results presented in the next section.

8 Simulation Results

In this section, we present initial simulation results, obtained with an implementation of the Pruning routing algorithm. The parameters for our simulations are as shown in the table. All experiments were performed using an underlying transit-stub topology [6] with 5 autonomous systems, each having 20 nodes. This represents the physical network as shown in Figure 1. We assumed a static overlay network, and generated random values for reliabilities of nodes and links. The results of our simulations have been averaged across the values obtained for algorithm runs for 10 different publishers distributed across the overlay routing substrate for an event type. The event notification was sent to a single subscriber, subscribing for an event type. In our experiments, the main focus has been on determining the reliability of the path attained by the Pruning algorithm, and the number of messages generated when achieving this reliability as compared to flooding the entire network. The reliability attained by the subscriber, was evaluated against the reliability the subscriber originally requested in Section 8.1. We then compare the message complexity of Pruning and Flooding to determine a reliable path in Section 8.2. In Section 8.3 we take a look at how the Pruning algorithm tends to Flooding, with increase in the levels of flooding. Finally, in Section 8.4, we study the influence of the connectivity of the overlay routing substrate, on the message complexity of the Pruning algorithm.

8.1 Actual Reliability Attained at Subscriber

![Figure 2. Actual v/s Expected Reliability for a Subscriber](image)

In this experiment, we plot the reliability value specified by the subscriber (Expected Reliability) versus the reliability that our algorithm is able to attain (Obtained Reliability).
As expected, Hermes, does not take into account, the reliability requirement of the subscriber, hence the reliability of the path established by Hermes remains constant. For a given reliability threshold, Pruning chooses a path that has maximum reliability amongst those paths which have a reliability value greater than or equal to the threshold. It is observed that, as we increase the reliability threshold, both Pruning and Flooding combine paths in order to meet the reliability requirement of the subscriber. Combining paths, sometimes results in a higher reliability than the specified threshold, as seen in the graph. Our experiment shows that with an increase in the reliability threshold, Pruning and Flooding always manage to attain a path with higher reliability than the path established by Hermes and that the Pruning algorithm manages to achieve the threshold reliability even with a flooding level of 2. We now compare the number of messages generated by Pruning and Flooding when determining reliable paths.

8.2 Message Complexity of Pruning and Flooding

Here we compare Pruning and Flooding in terms of their message complexities for determining reliable paths, given a reliability threshold requirement by the subscriber. We infer from this experiment that, although both Pruning and Flooding attain the reliability requested by the subscriber, the Pruning algorithm is far more efficient, since it does so with a much lower message complexity as compared to the Flooding algorithm. In the next section, we evaluate the performance of the Pruning algorithm in terms of message complexity with increasing levels of flooding.

8.3 Pruning Performance

In this experiment, for a given value of reliability for a subscriber, we increase the flooding levels for the Pruning algorithm and determine the number of messages generated at each level. The graph 4 depicts the percentage increase in the number of messages of pruning in terms of flooding. It is observed that pruning performance degrades with increasing levels of flooding. We conclude from this experiment, that the Pruning algorithm is efficient only if we find a reliable path with an average level of flooding.

8.4 Message Complexity of Pruning with respect to Connectivity of the Overlay Nodes

The message complexity of the algorithm is also affected by the number of nodes in the neighborhood set of each node. In this experiment, we increase the connectivity of the overlay network nodes (neighborhood set sizes 2, 3 and 4), and study the message complexity of the pruning algorithm for increasing levels of flooding. It is observed that the overlay network connectivity directly affects the message complexity of the algorithm i.e., the message complexity increases with the size of the neighborhood set for each level of flooding. Hence, we conclude that the higher the connectivity of the graph, lower the level of flooding that is required to determine reliable paths, and vice versa.
9 Conclusion and Future Work

In this paper, we have presented a multiplicative model for determining reliability of paths in a P2P overlay routing substrate, in order to route event notifications. We have also proposed an algorithm which leverages the Pastry routing algorithm, to determine a path having reliability, greater than or equal to the threshold, between a pair of overlay broker nodes. We have tested this algorithm using the Hermes middleware simulator. The algorithm has a higher message complexity in comparison to the Hermes routing algorithm, however unlike Hermes, it ensures a reliable event notification guarantee to subscribers. We have compared our routing algorithm with a Flooding approach. The simulation results show that our algorithm, determines a more reliable path as compared to Hermes, and one with a lower message complexity as compared to the Flooding approach. As part of the future work, we plan to test the algorithm for multiple subscriptions and event types and for different topologies of the P2P routing substrate. We also plan to study the effect of hierarchical and compositional relationships between the event types on reliable event notification. As discussed in Section 5, we consider the reliability of both nodes and links, when calculating the reliability of a path. We are currently building an analytical model to measure the reliability of a node on an overlay routing substrate. In our model we will measure the node reliability, and assume the reliability values for links, obtained from the link reliability measurement models in [1], [10], and [2]. We plan to incorporate this model to measure reliability of the nodes in an overlay substrate, and test the algorithm, in a system with dynamically changing values for reliability of nodes and links.

References