ABSTRACT
In this effort we tackle the important issue of providing Quality of Service (QoS) guarantees to subscribers, when routing event notifications in a publish-subscribe domain. In particular, we focus on providing reliability as a QoS guarantee, which is a proportion of published events received by a subscriber. We realize this by setting up routes that guarantee subscriber requested reliability on an event broker network where every broker node is associated with a dynamically changing node reliability value. Route determination is done in an adaptive fashion via our AR (Adaptive Reliability) algorithm that uses persistent path-quality information, in the form of reliability-estimates and (1) minimizes number of messages transmitted during route-establishment (2) guarantees reliable event notification delivery to the subscribers and (3) subsequently uses reliability-estimates to refine route quality adaptively. We validate the efficacy of AR empirically using DR-SIM (Dynamic Reliability Simulator), that we have built over Hermes. DR-SIM measures dynamically changing reliability values of every broker node in the network. Our results reveal that AR guarantees reliable event notifications to subscribers, with (1) lower message complexity compared to other existing efforts in this area, (2) is scalable in terms of space consumed with increasing sizes of broker networks and increasing number of clients (3) and is able to adapt to the dynamics of varying node reliabilities in the broker network.

Categories and Subject Descriptors
C.4 [Performance of Systems]: Reliability, availability, and serviceability

Keywords
Reliability, Refining, Adaptation, Event Broker Networks

1. INTRODUCTION

Event Based Middleware (EBM) has become a paradigm of choice for large scale event dissemination systems today. The class of users using such systems is significantly large, such as network administrators, professionals and news agencies. A wide scale application of EBMs on the Internet, is the RSS feeds. RSS feeds enable users to keep up with information available on the web in an automated manner, by subscribing to information of interest and receiving a notification from the RSS reader upon any changes to the published information. EBM is a manifestation of the publish-subscribe paradigm, composed of publishers and subscribers and a network of broker nodes lying in between, which route matching events from publishers to subscribers.

This network of broker nodes, is also known as the Event Broker Network (EBN). In a publish-subscribe domain, EBNs take the form of overlay networks, as shown in figure 1 over which events are disseminated from publishers to subscribers based on a set of subscriber matching filters. The overlay network is an application level graph composed of nodes and logical links, which form a subset of the underlying nodes and links in the physical network. The physical network consists of routers and links and the overlay becomes a P2P routing substrate. Each node in the overlay serves as a broker node that routes inbound events to subscribers.

Events are routed between publishers and subscribers, using various routing algorithms provided in the literature (e.g. [4], [13]). However providing service guarantees to clients when routing events is rare and most existing efforts in this area [4], [5], [2], [3], [6] provide either limited or no support for application level quality of service guarantees. A best-effort...
In EBNs, reliability as a QoS metric, is defined as a proportion of the notifications of a particular event type received by a subscriber from the total number of published events of the type. This metric in turn depends on the reliability of individual broker nodes forming a route, which is governed by the number of event drops occurring at every broker node. A detailed survey in [10], clearly highlights the need for further work in this area. In this paper, we explore the challenges and implications for reliable delivery of events to subscribers in the broker network. We propose a novel algorithm AR(Adaptive Reliability), that adapts routes in a broker network, to the dynamically changing reliabilities of broker nodes. AR maintains path-quality information in the form of reliability estimates, used for establishing routes and subsequently for adaptively refining route quality.

The efficacy of the goodness of route establishment by AR, can be measured in terms of the current utility. The current utility enables us to determine, if the system meets the QoS requirement (reliability threshold in this case) of the subscriber and how well does the system guarantee QoS (i.e., the reliability of the path along which the notification travels). Our aim is to ensure the best possible reliability to the subscribers with subscriptions currently alive in the system, and we validate our claims through experimental results, using the Hermes Event Based Middleware simulator.

### 1.1 Challenges and Contributions

Our task of sending reliable notifications to subscribers, involves establishing reliable routes between publishers and subscribers. This depends on the reliability of the individual broker nodes forming the route. The reliability of each broker node, depends on the number of events it is able to buffer. Each time an event is dropped, the reliability of the broker node decreases. Besides, the overlay network is dynamic in nature, i.e., the reliabilities of the nodes keep varying over time which leads to a variation in the overall route reliability. There are many ways of achieving reliable delivery. The brokers may be persistent in nature and send acknowledgement for every event received, which is expensive in terms of latency of event delivery. We work with buffered brokers with no acknowledgement messages, and propose a routing algorithm that guarantees subscriber requested reliability.

Our primary challenge here is, how do we determine a set of routes between the publisher and subscriber, which meet the reliability requirement of the subscriber? An obvious answer would be to flood the entire network and determine the best route(s), i.e., the route(s) meeting the subscriber specified reliability threshold. However, this increases the latency incurred during route establishment along with the number of messages introduced in the system. Hence the main question which arises is how do we determine reliable routes, with partial information of the network dynamics, with minimal latency and message complexity?

The main contribution of our work is the AR algorithm, that (1) finds the best possible route amongst the routes established using only local information provided by neighboring nodes (2) adapts the routes in a broker network, to the dynamically changing reliabilities of broker nodes (3) reduces message complexity once the established routes have been refined and a stable reliability is attained. Our experimental evaluation shows that AR establishes reliable routes for event notifications, with a significantly smaller message complexity and route establishment latency without flooding the entire network.

The rest of the paper is organized as follows. Section 2 provides a qualitative comparison of the related work in this area. Section 3 defines the problem. Section 4 discusses our approach and the algorithm based on this approach is presented in Section 5. Section 6 discusses the performance metrics for the experimental evaluation. We present the simulation setup and DR-Simulator, which we have implemented in the Hermes Event Based Middleware in Section 7. The experimental results are presented in Section 8 and we conclude with pointers to future work in Section 9.

### 2. RELATED WORK

Routing in overlay networks [12] mainly focuses on improving the application performance by selecting a higher quality path through the network, in comparison to the path chosen by the Border Gateway Protocol. The internet provides best-effort service and in order to enhance this model, there are many proposed seminal architectures such as Intserv [17] and Diffserv [17]. Realizing QoS in these architectures is not entirely feasible, since routes chosen are rarely optimized for application performance. As a result overlays have emerged as an alternative mechanism for providing value added services in the form of scalability, fault-tolerance and security.

Overlay networks are usually fault tolerant which implies the existence of multiple paths between two overlay nodes. This allows them to handle individual node failures by routing along redundant paths in the presence of node failures. TAROM [15] is one such overlay, which determines a secondary path on the overlay network, by minimizing the joint failure probability for a given primary overlay path between a source and destination. Path quality of overlay paths is inferred based on underlay information in RON [1], that is then used to route along the more effective paths. However both RON and TAROM considers only link quality without taking node quality into consideration, which is an important factor in determining reliable routes. Our reliability model is based on the reliability of the nodes occurring in the route from source to destination.

Hermes is an Event Based Middleware which uses the Pastry [13] overlay routing substrate for dissemination of events in the broker network. Hermes also defines a reliability model, which deals with robustness - i.e., middleware failures, and QoS guarantees which deals with reliability requirements explicitly specified by its clients. Hermes takes care of node failures by using the routing algorithms provided by Pastry for patching the routing tables of those nodes that were connected to the failed node to build other links that guarantee
a connected network. However Hermes does not consider the notion of event loss or client requested reliability based on event loss at brokers. We focus on the second part of the reliability model stated by Hermes - providing application level quality of service guarantees explicitly specified by the client.

Another effort in this area, [9], discusses the Pruning algorithm, which aims at establishing reliable routes for sending event notifications by employing partial flooding. As shown in the figure 2, every node floods all its neighbors, till a pre-defined level of flooding is achieved, and messages are then routed to the destination using the routing algorithm of the overlay. In case no single route exists satisfying the QoS guarantee, the routes are combined to improve the reliability and notifications are sent along multiple routes. While the Pruning algorithm does take event loss into consideration to determine reliability, it fails to adapt to dynamically changing network conditions and focuses instead on establishing a reliable path based on statically gleaned values of node reliability. The message complexity of the algorithm also increases with increase in the connectivity of the overlay nodes and size of overlay network. In this effort, we overcome these issues by proposing an adaptive algorithm, which refines its reliability-estimate of the routes established, between publishers and subscribers over a period of time, thus leading to a decrease in the overlay message complexity and the route establishment latency of the algorithm.

3. PROBLEM DEFINITION

Reliability, attained at a subscriber, in a publish-subscribe domain, has been defined in [9] as the ratio of the number of notifications received by the subscriber for an event type, against the number of publications of an event type. Assume \( n(e^2_\tau) \) is the total number of notifications received by a subscriber \( S \) for an event type \( \tau \) and \( p(e^2_\tau) \) is the total number of publications of an event of type \( \tau \) matching the subscriptions of subscriber \( S \). The reliability \( R \) which lies in \([0,1]\), attained at subscriber \( S \) for an event of type \( \tau \) is given by equation 1, which is reproduced here for convenience from [9].

\[
R[e^2_\tau] = \frac{n(e^2_\tau)}{p(e^2_\tau)} \quad (1)
\]

Using equation 1, reliability is measured at different levels in an event based system such as Per Subscriber Per Event Type, Per Subscriber All Event Types, All Subscribers for a single Event Type and Entire System as seen in [9].

Our system model, comprises an overlay network of brokers interconnected by logical links. Our primary goal is to send event notifications to subscribers, along reliable routes, meeting QoS requirements of subscribers. Initially we need to determine the reliability of the route along which the notification will travel to the subscriber. Each route comprises a series of broker nodes. Hence we divide the problem, (1) to calculate the reliability of an individual broker node (2) use this knowledge, to determine the reliability of the entire route. We elaborate both these issues below. The table contains notation that will be used throughout the paper.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>Total Number of Event Brokers</td>
</tr>
<tr>
<td>( N_1 )</td>
<td>Set of Neighbors for every Broker</td>
</tr>
<tr>
<td>( R )</td>
<td>Reliability of broker</td>
</tr>
<tr>
<td>( n_1, n_2 )</td>
<td>Neighbor Nodes</td>
</tr>
<tr>
<td>( s )</td>
<td>Source Node</td>
</tr>
<tr>
<td>( d_1, d_2 )</td>
<td>Destination Nodes</td>
</tr>
<tr>
<td>( R[\phi] )</td>
<td>Reliability of a path</td>
</tr>
<tr>
<td>( \phi )</td>
<td>Threshold Reliability specified by Subscriber</td>
</tr>
<tr>
<td>( M(a, b) )</td>
<td>Reliability-Estimate from node a to b</td>
</tr>
</tbody>
</table>

### 3.1 Calculation of Node Reliability

We model the broker node as a FIFO buffer and based on this, we define reliability of a broker node, as the ratio of the number of events processed by the broker node, versus the total number of events arriving at the broker. The reliability measured is event type agnostic. If \( t \) is the total number of events arriving at a broker, \( t' \) is the number of events dropped at the buffer of the broker node, then the reliability \( R \), of the broker, is given by the following equation:

\[
R = \frac{t-t'}{t} \quad (2)
\]

### 3.2 Calculation of Route Reliability

We now use the node reliability, to determine the reliability of a route established between a publisher and subscriber using the multiplicative model discussed in [9], where in the route established between a publisher and subscriber maybe viewed as a series system [16]. If \( n \) is the total number of nodes occurring in the series path to the subscriber and \( R_i \) is the reliability attained at each broker node calculated using equation 2, then the reliability of the path is given as follows:

\[
R[\Theta] = \Pi_{i=1}^{n} R_i \quad (3)
\]

We assume perfect links (i.e., there are no link losses) and focus only on measuring the node reliability, since there are existing efforts [15], [1] in overlay networks, which have proposed link reliability measurement models. The link reliability can be easily incorporated in the route reliability equation by calculating a product of the node and links occurring along the path as stated in [9].

Assume that a subscriber has subscribed for an event with a reliability requirement \( \phi \). Our task is to establish a route or set of routes, between a given publisher and subscriber, such that, \( R[\Theta] \geq \phi \). \( \Theta \) represents a path between publisher and subscriber, which is made up of a set of broker nodes. If a single route exists between a publisher and subscriber which satisfies this condition, then a notification is sent to the subscriber along this path. If no single series path exists, which meets the reliability requirement of the subscriber, then the multiple paths, which are established
by AR, are combined s.t., the reliability of the combined paths is greater than or equal to the threshold, and notification is sent to the subscriber along parallel paths. We intend to follow the same approach for calculation of parallel path reliability as discussed in [9].

Based on the above discussion, the problem we solve in this paper has 2 parts to it. (1) Initially our task is to use the partial information at the broker to make an informed decision about the next-hop during route establishment. (2) Once a reliable route is established, refine the route using partial information, so that the reliability of the route is stabilized. To solve this problem, we first come up with a route refining model, which will refine the routes established between publisher and subscriber. Next we propose AR, an adaptive algorithm, which uses this model to make informed decisions during route establishment, minimizes the message complexity and is scalable in terms of space complexity once the established routes are refined, while meeting our basic premise: \( R[Θ] \geq Φ \).

We discuss the route refining model and AR in Sections 4 and 5 respectively.

4. ROUTE REFINING MODEL
From Section 2 we have seen that, the existing approaches, do not take into account node reliability and there is performance degradation in terms of the message complexity with increase in the size of the broker network. We propose a model, which refines the quality of the route established over a period of time. The primary goal of our algorithm is to establish routes s.t., the reliability of either a single route or a combination of routes is above the subscriber specified threshold, for a given source-destination pair. Our route-establishment policy answers the question: Is the route (set of routes) chosen, attaining the best possible reliability over the threshold? We use ideas from the field of reinforcement learning [14], [8], to try to update our policy locally, i.e., we use local information available at each node, in order to take a decision when choosing a route. Using local information, reduces the number of messages in the system, as we are able to determine, the quality of the entire route, by merely obtaining information from the neighboring node. This information helps us take a decision for the choice of a next hop neighbor.

In our model, every broker node, keeps a reliability-estimate of the path, from its neighbors to the destination broker node, as a matrix of neighbor-destination pairs. The reliability-estimate matrix at each broker, looks as follows:

\[
\begin{bmatrix}
    d_1 & d_2 & d_3 & \ldots \\
    n_1 & M(n_1, d_1) & M(n_1, d_2) & M(n_1, d_3) & \ldots \\
    n_2 & M(n_2, d_1) & M(n_2, d_2) & M(n_2, d_3) & \ldots \\
    \vdots & \vdots & \vdots & \vdots & \ldots \\
\end{bmatrix}
\]

Assume \( s \) is the source broker node and the figure depicts the matrix stored at broker node \( s \). \( d_1, d_2, d_3 \) are the destination broker nodes, and \( n_1 \) and \( n_2 \) are neighbors of node \( s \). The matrix for \( s \) stores reliability estimates for the destinations, via all it’s neighbors. The destinations, represent subscribers, and the broker node \( s \), could be a source node or an intermediate node in the route to the destination. Since the reliability estimate value is refined every time a route is established, each entry in the matrix \( (e.g. \text{M}(n_1, d_1)) \), contains a set of reliability estimate values. When a broker is queried for it’s reliability-estimate for a particular neighbor-destination pair, it returns the average of the reliability-estimate values attained over a period of time.

The calculation of the \( M \) matrix value is dependent on three factors. (1) The Refining Rate parameter, which captures the variations in the reliability-estimates of the routes over a period of time. (2) The Biasing Factor which assigns weights to the choice of next hop node. We choose two next-hop nodes at each step, in order to introduce redundant routes. One is chosen on the basis of the reliability-estimate and the other is chosen at random. (3) The Reward earned at each hop. In our model, the reward is the reliability attained at each broker node. We now elaborate each of these:

1. Refining Rate: The refining rate \( µ \), s.t. \( 0 \leq µ \leq 1 \) is used to capture the variations in the reliabilities of the routes established, in a broker network with dynamically changing values of reliabilities of broker nodes. If \( µ \) is closer to zero, the broker node will tend to consider only the immediate rewards, when deciding to choose a route, where as, if \( µ \) is closer to one, the broker node will consider future rewards occurring in the route with greater weight, willing to delay the reward. Since the broker network, comprises of dynamically varying reliabilities of nodes, we need to keep the route-estimate information (i.e. \( \text{M}(n_i, d_i) \)) updated at each broker node for every neighbor-destination pair. The refining rate parameter \( µ \) captures these changes in the reliability-estimate of routes established over a period of time. This is because, each broker refines its reliability-estimate with experience, and we obtain a steady state for values for every \( M \) matrix.

2. Biasing Factor (Degree of Randomness - DoR): When a source broker node \( s \) has to route a message to a destination broker \( d \), it will choose the next hop neighbor, which will ultimately be a part of a route to the destination. The obvious method is to choose the neighbor node, having the highest reliability-estimate, i.e., a neighbor having the highest value of the \( M \) matrix, amongst all the neighbors. However choosing an immediate neighbor with the highest reliability-estimate may not be the ideal approach, since the reliability-estimate of the nodes occurring later in the route, may cause a drop in the overall route reliability. Hence we use a biasing factor \( α \), s.t., that \( 0 \leq α \leq 1 \), for choosing a node with the highest value of reliability-estimate amongst all the neighbor nodes, and a factor of \( (1-α) \) for choosing another node at random amongst all the neighbors (except the one with the maximum reliability-estimate) when routing the message. As \( α \) tends to 1, our approach turns greedy, since we choose a neighbor node having maximum reliability-estimate
with higher weightage, and as \( \alpha \) tends to 0, we give higher weightage to a random selection of a neighbor node.

3. Reward (Reliability) Attained: Every node in the event broker network, has a reward associated with it. The reward for a broker node, is the reliability attained at the broker node, which is calculated as described in Section 3.1. The reward attained at every broker node is a part of the calculation of the \( M \) matrix (i.e., calculation of the reliability-estimate). And every broker node will take a decision for route establishment, based on the reliability estimate stored in the \( M \) matrix.

Assumption- The reliability-estimate calculation is based on the assumption that the estimate from the chosen neighbor to the destination is more accurate as compared to the estimate from the source to the destination, assuming that the chosen neighbor is closer to the destination than the source.

Next we discuss how the reliability-estimate is calculated, and how each broker node updates its \( M \) matrix. Given \( 0 \leq \alpha \leq 1 \) and \( 0 \leq \mu \leq 1 \), if \( M(s,d) \) represents a source-destination pair in the event broker network, and \( R(n_1) \) and \( R(n_2) \) are the reliabilities (or rewards) earned by node \( s \), when visiting its neighbors \( n_1 \) and \( n_2 \), s.t. \( n_1, n_2 \in N_r \), then the \( M \) matrix for a broker node can be represented by equation 4.

\[
M(s,d) = R(s) + \mu \{ \alpha [R(n_1) + M(n_1,d)] + (1-\alpha) [R(n_2) + M(n_2,d)] \}
\]

Next, we express a set of constraints for the adaptive nature of the algorithm, as it refines the routes over a period of time. If the decision to adapt, were unconstrained, then there is a possibility that a cyclic path may update the matrix at a broker. To avoid this, we express rules given below, which ensure that a broker estimates only non-cyclic paths. Assume \( P\{n_i\} \) represents the set of nodes in a route being established so far (i.e., the broker nodes occurring along the route chosen by the AR-algorithm to a subscriber) and \( N_r \) is the set of neighbors for source node \( s \).

\[
M(s,d) = \begin{cases} 
R(s) + \mu \{ \alpha [R(n_1) + M(n_1,d)] + (1-\alpha) [R(n_2) + M(n_2,d)] \} & \text{if } n_1, n_2 \notin P\{n_i\} \\
M(s,d) & \text{if } n_1, n_2 \in P\{n_i\} \\
R(s) + \mu \{ \alpha [R(d)] + (1-\alpha) [R(n_2) + M(n_2,d)] \} & \text{if } d \in N_r
\end{cases}
\]

- **Non Cyclic Route**: None of the neighbors occur in the route established so far, hence the \( M \) matrix is updated as per equation 4.

- **Cyclic Route**: This constraint, states that the broker matrix \( M(s,d) \) will not be updated, if the neighbor nodes occur in the route established so far. This is done in order to avoid cycles in routes being established.

- **Route Termination**: If the destination node appears in the next hop, then we assign the biasing factor \( \alpha \) to destination node, regardless of whether it gives maximum reward or not and we randomly choose another neighbor to route to, with a biasing factor of \( 1-\alpha \). This ensures that the algorithm converges and we establish a route which is shorter than other routes, which are still being traced.

In the next section, we present the Adaptive-Reliability(AR) algorithm, based on the route refining model we have discussed so far, to establish reliable routes between publishers and subscribers.

5. THE AR-ALGORITHM

The AR (Adaptive Reliability) algorithm is discussed at the level of a single event broker, but will be applicable to all
brokers in the event broker network. The input to the algorithm is the source broker node, the destination broker node, refining rate $\mu$ and biasing factor $\alpha$. The algorithm consists of three phases. The first phase is the querying phase in which the broker queries its neighbor nodes for their reliability estimates. The second phase, is the refining phase, in which each broker refines its reliability-estimate. The final phase, or notification phase, is one in which the broker network, sends event notifications along previously refined routes. We now discuss the three phases of Algorithm 1 in detail.

Algorithm 1 ARAlgorithm

Require: $R_{path}$, currentNode, $\mu$, $\alpha$, $s$, $d$
Ensure: The value of $R_{path}$

1: $N_r \leftarrow$ Get Neighbors of currentNode
2: while there are more neighbors $\in N_r$ do
3: $M(n, d) \leftarrow$ Query Neighbor Reliability-Estimate
4: end while
5: $n_1 \leftarrow \arg \max \{M(n, d)\}$ s.t. $n_1 \in N_r$
6: $n_2 \leftarrow$ neighbor $\in N_r$ with random reward $\in \{M(n, d) - M(n_1, d)\}$
7: if $n_1, n_2 \in$ partial path then
8: Drop path
9: else if $d \in N_r$, then
10: $M(s, d) \leftarrow R(s) + \mu \{\alpha[R(d)] + (1 - \alpha)[R(n_2) + M(n_2, d)]\}$
11: $R_{path} \leftarrow$ Calculate reliability of partial path
12: else
13: $M(s, d) \leftarrow R(s) + \mu \{\alpha[R(n_1) + M(n_1, d)] + (1 - \alpha)[R(n_2) + M(n_2, d)]\}$
14: $R_{path} \leftarrow$ Calculate reliability of partial path
15: ARAlgorithm
16: end if
17: Send reliabilities and paths to source
18: Send notification to subscriber

1. **Querying Phase:** The current node, sends a query message (step 3) to each of its neighbors, in order to determine the reliability estimate of each of its neighbors to the destination. Each neighbor in turn queries its neighbors, for their reliability-estimate, till it reaches the destination.

2. **Reliability Phase:** Once the node has the reliability-estimates, it finds the neighbor having the highest reliability-estimate (step 5) and randomly picks (step 6) one neighbor from the rest. For both the chosen neighbors, we check if they have already occurred previously on the path established so far (step 7). This check is introduced in order to avoid cycles. If the node previously occurs on the path, then the path is dropped (step 8). If the destination node occurs as one of the neighbors, then the destination node is set as the maximum reliability-estimate neighbor by default. This is because, we follow the multiplicative model when calculating reliability along a path. Hence, shorter the path, higher is the expected reliability of the overall path. The matrix is updated as shown in step 10. If not, then we update the matrix of the current broker node as shown in step 13, with the reliability-estimates obtained. We then send a message to both the chosen neighbors and also update the reliability of the partial path (step 14). The AR-Algorithm is recursively called (step 15) for each of the neighbors until destination node is reached. Once the routes converge to the destination, the reliability values, along with the routes, are sent to the source (step 17).

3. **Notification Phase:** At the end of the refining phase, the source node will have a set of paths, with their reliability values. Each broker also has a reliability-estimate of the $M$ matrix, to the destination via its neighbors which has been refined. The source node, then takes a decision to send a notification either along a single route or multiple routes as discussed in Section 3.2. A single path or a parallel path notification, meeting the reliability requirement will be sent to the subscriber (step 18).

5.1 Message Complexity of AR

As a part of the complexity analysis, we study the message complexity of the AR-algorithm which is in terms of the number of messages generated in the system during route establishment. AR is a heuristic based algorithm. Also owing to the dynamic nature of the broker network, we define a new metric to determine the time complexity of our algorithm in Section 6, and illustrate the time complexity of AR, through experimental results.

- As described in Section 5, there are 2 messages sent from each broker in the network in AR. Initially, a broker, will query all its neighbors, in order to find the neighbor’s estimate of the path-reliability to the destination. Each neighbor in turn, will respond with the reliability it has estimated so far. If each broker node has $n$ neighbors and there are $N$ nodes in the broker network, then the total number of messages introduced in the network, when querying the reliability estimates of the neighbors is $N(2n)$. Once the broker receives estimates of reliability from all its neighbors, it chooses two of its neighbors, as described in Section 5 and sends messages to those brokers. If $\text{diam}$ is the diameter of the broker network, then the maximum number of messages introduced when establishing reliable routes is $\sum_{i=1}^{\text{diam}} 2^i$.

- Once the paths converge at the destination, an additional $(\log N)$ messages are introduced in the system, in order to route the reliability values of the paths established back to the source. The messages introduced here are $(\log N)$ since we use the routing algorithm provided by the Pastry overlay routing substrate.

- Once the path establishment phase is complete, notification can be sent along maximum of $2^\text{diam}$ paths, if multipath routing is required, in order to meet the reliability requirement of the subscriber.

The maximum number of messages $M$ introduced in the broker network, during route establishment of the AR-Algorithm is given by the following expression.

$$M = N(2n) + \sum_{i=1}^{\text{diam}} 2^i + (\log N)$$
From the above expression we infer, that the message complexity of the AR Algorithm, once stable reliability is attained, is of $O(2^{\text{diam}})$

### 6. Performance Metrics

In this section, we discuss the performance metrics for our experimental evaluation. Each experiment calculates the system reliability (reliability attained across all subscribers in the broker network) and the time required to refine routes to attain stable reliability. The simulation is done against a backdrop of a network of broker nodes having random topology, with a fixed number of publishers and subscribers assigned to the broker nodes. The publishers and subscribers are distributed throughout the broker network with a uniform random distribution. The outcome of each experiment is to calculate the system reliability, i.e., the reliability attained across all subscriptions. Every experiment is repeated multiple times and results averaged. Each experiment consists of multiple rounds of refinement, the number of which is driven by reliability converging to a stable value. During each of these rounds, AR establishes reliable routes between the publishers and subscribers that is a refinement of the routes established during the previous round and sends notifications to each of the subscribers at the end of route establishment phase. This refinement of routes continues, until stable reliability (defined below) is attained. Successful repetitions of an experiment start afresh in terms of events (essentially on an empty network) but with a different distribution of clients on the broker network. Initially, the buffers at the broker nodes are empty and the system reliability attained at the end of the run as expected is 100%. As more events continue to be published, the buffers get full and start dropping events and the reliability of the broker nodes starts falling, thus leading to a variation in the system reliability. We ignore these initial variations in the reliability in our experiments, by identifying the initial transient and smoothing it using the Welch Technique [7].

The Welch technique uses a moving window averaging mechanism to remove the initial transients from data sets. The window size $w$ is defined s.t., $w < \frac{k}{4}$, where $k$ is the total number of readings in the data set. If $l$ is the point of stability, (total $m$ reliability values) for each simulation, then we determine the confidence intervals for the guaranteed reliability, based on the reliabilities attained in the interval $[l, m]$. We now introduce some metrics for our experimental calculations

1. **Stable Reliability** - The time from the beginning of the first run to the beginning of the time interval during which the deviation of the system reliability lies within $\varepsilon \%$ ($\varepsilon=0.75$) of the mean reliability attained is called Time To Stabilize (TTS). If $\sigma$ is the standard deviation about the mean, and $\bar{R}$ is the mean reliability attained, and $t_{\text{begin}}$ and $t_{\text{end}}$ are the beginning and end of the time interval, then, TTS represents the time from the beginning of the first run to the beginning of the time interval, $t_{\text{begin}}$, s.t. $\sigma < (\bar{R})$, for all values of reliability lying within the time interval $[t_{\text{begin}}, t_{\text{end}}]$. The mean reliability attained in this time interval, is the stable reliability attained by the AR algorithm. The reliability attained is driven by the dynamics of the overlay network such as variations in the traffic load, which directly affects the reliability of a broker node as events start getting dropped at the buffer. We use sliding window technique, to verify the stability of the mean reliability over all intervals, once the TTS is determined.

2. **Route Establishment Latency** - If a subscriber subscribes for an event at time $t$, and the time at which the refinement and establishment of reliable routes in the overlay network for this subscription is complete is $t'$, then the Route Establishment Latency is $(t'-t)$. Route Establishment Latency depends primarily on the size of the broker network and the connectivity of each broker. The diameter of the network plays a role in the determining the upper bound for latency, when establishing routes in overlays and we verify this claim through experiments conducted on increasing sizes of broker networks.

3. **Cost of Route Establishment** - The cost for establishing reliable routes is measured in terms of the number of messages generated in the system, during the route establishment phase. This reflects the message complexity of the algorithm. The cost is dependent on the degree of each broker node, during the Querying-phase of the AR-Algorithm, since each neighbor is queried for its reliability-estimate matrix, till the destination node is reached. Once the routes are refined and a stable reliability is attained, the cost decreases since each broker no longer queries its neighbors for the reliability-estimate. This leads to a decrease in the overall cost during reliable route establishment.

### 7. Simulation Setup

The parameters for our simulations are as shown in the table. All experiments were performed using an underlying transit-stub topology [11] with 10 autonomous systems, each having 100 nodes. The physical topology of the network comprised of 1000 nodes. Event arrivals form a Poisson process with $\lambda=100$. Each time the reliability was calculated for the entire system, i.e., averaged across all subscribers and event types. We conducted 300 runs for each simulation and repeated each experiment 3 times.

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transit Stub Topology</td>
</tr>
<tr>
<td>Number of Event Brokers</td>
</tr>
<tr>
<td>Neighborhood Set Size</td>
</tr>
<tr>
<td>$\mu$</td>
</tr>
<tr>
<td>$\alpha$</td>
</tr>
<tr>
<td>Number of Publishers</td>
</tr>
<tr>
<td>Number of Subscribers</td>
</tr>
<tr>
<td>Number of Event Types</td>
</tr>
<tr>
<td>Threshold Reliability $\phi$</td>
</tr>
</tbody>
</table>

#### 7.1 Dynamic-Reliability Simulator

Our aim in building a Dynamic-Reliability-Simulator (DR-Sim) was to be able to measure dynamically changing values of reliabilities of broker nodes. This was done in order to simulate a real world scenario in which, publishers and subscribers publish and subscribe to events at different times, with continuously varying reliability values of the overlay network.
network nodes. The equation for calculating the reliability of a node is as described in Section 3.1. To measure the reliability of overlay nodes, we introduced a configurable finite-sized buffer at each event broker in the Hermes middleware simulator. The Hermes simulator, establishes the initial topology of the overlay network and creates the event brokers. We use uniform random distribution to select brokers in the event broker network, to which the publishers and subscribers will get attached. We assume that every publisher, publishes a single event type in the system, and event publication arrivals form a Poisson process. We then determine the times at which a publisher will advertise its events and the times at which a subscriber will subscribe to those events. This event instance is then routed by Hermes to subscribers (if any) which have subscribed to this event type at that point in time.

Every broker node is designed to have a finite sized buffer, so that it will drop events once the buffer is full. Any event delivered to the node, is buffered, if the node is busy processing some other event at that time.

8. EXPERIMENTAL RESULTS

Our goal is to study the performance of the AR algorithm vis-a-vis the Pruning algorithm [9], which we have implemented on the Hermes Event Based Middleware simulator. We study the performance of both these algorithms in terms of the reliability attained and the message complexity incurred during the route establishment phase. Besides, we also conducted a detailed experimental evaluation of AR to study how the refining rate affects the performance metrics and how the biasing factor influences our decision in choosing next hop neighbors when establishing routes. We determine the time to stabilize (TTS) for adapting to reliable routes during route establishment and the variation in the message complexity of the AR-algorithm once a stable route is determined. We compare the AR-algorithm with the Pruning algorithm in terms of message complexity, and compare the reliability attained with a non refining random (NRR) approach. The NRR algorithm, picks two neighboring broker nodes at random at every hop during route establishment. It does not take into account the dynamics of the broker network or make use of any historical data stored at broker nodes in decision making when establishing routes.

8.1 Effect of Refining Rate and Bias Factor on Reliability Guarantee

In these set of experiments, we study the effect of the refining rate in terms of the reliability attained and time to stabilize (TTS), for varying sizes of the broker network. As shown in graphs 3(a), 3(b), 3(c) and 3(d), we keep the bias factor constant (=0.9) or (=0.1) and observe the reliability attained and TTS with increasing number of brokers in the overlay network. The number of clients (i.e., the publishers and subscribers) are constant and a QoS guarantee of 70% reliability system-wide, is required for event notifications. We observe that with increase in the size of the broker network, the time taken by the algorithm to stabilize decreases. This is because the algorithm is able to find routes quickly, as there are more broker nodes to choose from. The reliability of the routes established increases for the same reason. We also observe that a higher refining rate establishes a route with higher reliability as compared to one with a lower refining rate, however a higher refining rate takes longer time to stabilize as compared to a lower one. Reason being, a higher refining rate takes into account the future rewards. As a result the estimated value of the reliability matrix varies, and the choice of a neighbor (depending on the reliability-estimate matrix) also changes, which increases the TTS. However in the process, a higher refining rate attains a better route in terms of reliability attained.

Another observation we make here is that the behavior of the refining rate overrides the degree of randomness in the system, with a higher refining rate always determining a route with a higher reliability as compared to a lower one. However for both the refining rates, a higher bias factor (=0.9) always determines a route with a better reliability than the lower one. This is because with a lower degree of randomness (high bias factor), we give weightage to the refined route as against the random one. This is clearly seen in the graphs 3(e), 3(f), 3(g) and 3(h).

Next we increase the number of clients (subscribers) in the system by keeping the size of the broker network constant. We observe (Figures 3(i), 3(j)) how the refining rate affects the reliability and TTS keeping a high bias factor (0.9) towards the refining rate. It is expected that as the number of subscribers in the system increase, the time to stabilize also increases, since the number of destination nodes (subscribers) for which to establish reliable routes increases. However the reliability attained either remains constant or shows a slight, though not significant decrease. This is mainly because with increase in the number of subscribers, the traffic in the overlay network increases, which leads to a drop in the route reliability. However, we have observed (with increase in subscribers upto 100) that this drop is not very significant as seen from the graphs and the reliability requirement of the subscribers is guaranteed. The refining rate parameter shows similar behavior as in the case of increase in size of overlay.

Finally, we increase the size of the broker network, with a bias factor of 0.9 and observe the behavior of the refining rates 0.1 and 0.9 as shown in Figures 3(k) and 3(l). Our simulations show that the reliability of the routes established goes on increasing with increase in the size of the overlay network, and almost tends to 100%. On the other hand, as expected the TTS goes on decreasing. We can conclude that, for large network sizes our algorithm, shows the best performance in terms of reliability of routes established for a low degree of randomness and a high refining rate. However the TTS for a high refining rate is greater than the TTS required for a low refining rate. The difference in the increase in TTS is observed to be ≤ 10%, hence the refining rate chosen should depend on the latency constraint (if any) specified by the subscriber. If there is no latency constraint, then a high refining rate with a low degree of randomness, is the ideal choice for large sizes of broker networks.

8.2 AR Performance in comparison to Pruning and NRR

We compare the performance of AR Algorithm, with Pruning [9] and a base case - Non Refining Random (NRR). As
Figure 3: Effect of Refining Rate and Bias Factor on Reliability Guarantee
In the case of NRR, we randomly pick two neighbors at every step and route the event. There is no refining of the routes based on reliability estimates. As seen from Figure 4(b), the reliability attained by AR is consistently higher than NRR, which uses a random approach for route establishment. As against this, the AR-Algorithm refines routes based on future rewards obtained, and hence attains a much higher system reliability as compared to NRR.

Figure 4: AR Performance and Adaptive Message Complexity

Figure 5: AR-Pruning Performance

seen in Figure 4(a), the number of messages in the system increase, with increase in overlay connectivity for the Pruning algorithm as the flooding levels increase. In comparison the AR-algorithm has a lower message complexity. Reason being, once a stable route is established, the AR-algorithm does not query every neighbor for its reliability-estimate. From this point onwards the message complexity (and also the route establishment latency) of the AR-algorithm drops further (Figures 4(c) and 4(d)) as the AR-Algorithm adapts to routes which have been previously refined. Graphs 4(c) and 4(d) show the drop in message complexity and latency for route establishment, for increasing sizes of broker networks, over a period of time (shown in terms of number of runs). Hence, we conclude that for highly connected networks the AR-algorithm shows a better performance as compared to the Pruning algorithm in terms of message overhead.

In the case of NRR, we randomly pick two neighbors at every step and route the event. There is no refining of the routes based on reliability estimates. As seen from Figure 4(b), the reliability attained by AR is consistently higher than NRR, which uses a random approach for route establishment. As against this, the AR-Algorithm refines routes based on future rewards obtained, and hence attains a much higher system reliability as compared to NRR.

Figure 6: Refining Rate and Message Complexity

In Figure 5 we compare the performance of the Pruning algorithm and AR, in terms of the reliability attained, over a period of time (shown in terms of number of time intervals, where each time interval is a run of the experiment). We observe that the reliability attained by AR, stabilizes after a point where as the reliability attained by the Pruning algorithm may also drop below the threshold (less than 0.7). This is because, the Pruning algorithm blindly floods, until a predefined level of flooding and hence though it may attain reliable routes initially, with increase in traffic in the network, the Pruning algorithm is not able to refine routes and adapt and is totally dependent on the route taken by the routing algorithm of the overlay network.

8.3 Effect of Refining Rate on Message Complexity

In this experiment (Figure: 6(a)) we study the effect the refining rate has on the message complexity by varying the connectivity of each broker node in the overlay network. As expected the message complexity increases, with increase in the overlay connectivity. This is because every broker node queries its neighbors for the reliability estimate, when making a choice of the next hop neighbor, till the route is refined. An interesting observation we make here is that, a higher refining rate has greater message complexity than a lower refining rate. A higher refining rate implies, we are giving weightage to future rewards. With increase in the overlay node connectivity, the number of neighbors to choose from based on their reliability-estimate matrix increases. As a result, with each route establishment, the choice of neighbor may change, leading to different routes being established. This behavior is obvious from the standard deviation obtained for the message complexity as depicted in Figure 6(b).

8.4 Space Efficiency

We studied the distribution of the additional routing table entries at each broker, introduced by the AR-algorithm, in comparison to the original Hermes routing algorithm, with increasing sizes of overlay networks and increasing number of clients connected to the overlay. In Figure 7(a), we keep the size of the overlay network constant and increase the number of subscribers. It is observed that the number of routing table entries at each broker node cannot exceed the number of subscribers in the broker network. In every case it is observed that a majority of the brokers were not involved in the routing. Less than 30% of the brokers involved in the routing had the maximum number of entries in their routing tables, while the rest of the entries were distributed across the remaining 70% brokers in the network. Similar behavior
was observed with increasing the size of the broker network as shown in Figure 7(b). We see that the AR-algorithm, introduces negligible number of additional routing table entries and hence is space efficient.

8.5 Confidence Intervals for Route Reliability

<table>
<thead>
<tr>
<th>φ</th>
<th>95% confidence</th>
<th>99% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.7081 [0.7052, 0.711]</td>
<td>0.7081 [0.7046, 0.7110]</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7537 [0.7505, 0.7569]</td>
<td>0.7537 [0.7499, 0.7575]</td>
</tr>
<tr>
<td>0.8</td>
<td>0.8281 [0.8248, 0.8314]</td>
<td>0.8281 [0.8241, 0.8321]</td>
</tr>
<tr>
<td>0.9</td>
<td>0.9036 [0.8996, 0.9076]</td>
<td>0.9036 [0.8988, 0.9084]</td>
</tr>
</tbody>
</table>

We conducted this experiment in order to determine confidence intervals for the reliability value attained by the AR-Algorithm. We calculated the overall system reliability over a period of time for ten different event types. Based on the threshold reliability requirement of the subscriber \( \phi \), we obtained intervals with 95% and 99% confidence. From the table we infer that the stable reliability attained is always greater than or equal to the threshold reliability requirement. Also in most cases (except for the one with \( \phi = 0.9 \)), the confidence interval for the route-reliability attained by the AR-Algorithm always has a lower bound greater than the threshold reliability. The only exception is the case with a threshold reliability requirement of 0.9, where we see that the lower bound of the confidence intervals of both (95% and 99% confidence) drops below the threshold reliability of 0.9.

9. CONCLUSION

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Message Complexity</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>Low</td>
<td>Best reliability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>above threshold</td>
</tr>
<tr>
<td>Pruning</td>
<td>High</td>
<td>Meets threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reliability</td>
</tr>
</tbody>
</table>

As summarized in Table 9, the AR-Algorithm performs better than the Pruning algorithm in terms of message complexity once it attains a stable reliability and is also consistent in terms of the reliability attained as compared to Pruning. In the case of NRR, even though the AR-Algorithm has a higher message complexity during the Querying Phase, its messages complexity reduces, once a stable route has been established and is comparable to the message complexity.

Based on our experimental results, we arrive at the following conclusions regarding refining rate and bias factor:

- A high refining rate and a low degree of randomness, establishes routes with the the highest possible reliability (above the threshold), at the cost of a slight increase (\( \leq 10\% \)) in the time to stabilize. This result holds good for varying sizes of overlays, and varying number of clients within a fixed overlay.
- Keeping the refining rate constant, a low degree of randomness in the choice of a neighbor node, again shows better performance (in terms of reliable routes established), since we give weightage to the refined route as against the random route. Similar results are attained with varying sizes of overlays and varying number of clients within a fixed sized overlay network.
- The algorithm also shows optimal performance in terms of space efficiency, since the maximum number of routing table entries at each broker do not exceed the number of subscribers in the broker network. Also the total number of brokers having maximum number of routing table entries is low (<30%) as seen from the simulations.
- The AR-algorithm, initially shows a high message complexity, when refining the route, but once the route is stable, there is a gradual decrease in the message complexity and the route-establishment latency.
of NRR. But the reliability attained by AR is consistently greater than that attained by NRR.

Overall the AR-algorithm performs better than the Pruning algorithm, by refining reliable routes with a lower message complexity as compared to Pruning. The concept of using redundancy to achieve reliability is similar to that used in the Pruning algorithm - but additionally AR, also refines routes along the way and takes decisions for sending event notifications along previously refined and stable routes. This also leads to a decrease in the overall latency for notification of events to subscribers. AR has also proved to be scalable as compared to the Pruning algorithm, in terms of number of clients in the broker network and size of the overlay.

In comparison to Hermes, the additional space required at every broker by AR is negligible, and AR also sends notifications along routes that meet the reliability requirement of the subscriber.

In this paper we presented an adaptive algorithm that refines routes based on dynamically changing network conditions. We focussed on measuring the efficacy of AR in terms of current utility of a system. Current utility grants the best possible reliability above the subscriber specified threshold, with available resources as opposed to the notion of future utility, which conserves capacity for future requests. We now plan to study the efficacy of AR in terms of future utility, which answers questions pertaining to the capacity of the network, such that, future QoS requests are guaranteed, given the available resources after guaranteeing QoS in the current scenario. We also plan to refine the AR-Algorithm further, by removing the bias-factor towards randomness and incorporating other metrics for making a choice of next-hop neighbor, when establishing reliable routes.

10. ACKNOWLEDGMENTS

We would like to thank Ramdas Rao for his contribution in building the Dynamic-Reliability Simulator.

11. REFERENCES


