ADDING UNDERLAY AWARE FAULT TOLERANCE TO HIERARCHICAL EVENT BROKER NETWORKS

Madhu Kumar S. D., Umesh Bellur
Indian Institute of Technology Bombay, Mumbai, India

Erusu Kranthi Kiran
National Institute of Technology Calicut, Calicut, Kerala, India

Keywords: Overlay, Underlay Awareness, Event Broker Networks, Fault Tolerance.

Abstract: Recent studies have shown that the quality of service of overlay topologies and routing algorithms for event broker networks can be improved by the use of underlying network information. Hierarchical topologies are widely used in recent event-based publish-subscribe systems for reduced message traffic. We hypothesize that the performance and fault tolerance of existing hierarchical topology based event broker networks can be improved by augmenting the construction of the overlay and subsequent routing with the underlay information. In this paper we present a linear time algorithm for constructing a fault tolerant overlay topology for event broker networks that can tolerate single node and link failures and improve the routing performance by balancing network load. We test the algorithm on the SIENA event based middleware which follows the hierarchical model for event brokers. We present simulation results that support the claim that the use of underlay information can significantly increase the robustness of the overlay topology and performance of the routing algorithm for hierarchical event broker networks.

1 INTRODUCTION

The recent popularity of the publish/subscribe systems can be attributed to the flexibility and ease of deployment of overlay topologies and their routing algorithms. Extensive research has been done on creating scalable, highly efficient overlay topologies and routing algorithms (A. Carzaniga, 2001; L. Fiege, 2002; Pietzuch, 2004), but the importance of using the underlying physical topology information in their development has not been considered in most of the research works. Recent studies (Tang and McKinley, 2004) have shown that the incorporation of underlay information(i.e information about the first four layers in the network protocol stack) in the overlay topology and routing algorithms can significantly improve their performance.

We propose that static hierarchical overlay topologies can be improved by including underlay information to make them adaptive to faults. We present a fault tolerant underlay aware overlay construction and maintenance algorithm for a hierarchical overlay network. The constructed overlay network is capable of handling single node and link failures and also provides the flexibility to balance network load, thereby improving the routing performance of the event broker network over the simple hierarchical network that does not incorporate underlay information. The rest of this paper is organized as follows: In Section 2 we present the related work in this area. In Section 3, the importance and feasibility of use of underlay information in overlay construction is discussed and our procedure for underlay aware overlay topology generation is outlined. In Section 4, algorithms for overlay topology creation and maintenance are described in detail, the mechanisms for handling failures are described and the proof of correctness for the concept used in the algorithm is given. Section 5 describes the experimental setup and presents the implementation results that demonstrate the improvement in performance achieved. Section 6 concludes the paper.

Kumar S. D. M., Bellur U. and Kranthi Kiran E. (2007).
ADDING UNDERLAY AWARE FAULT TOLERANCE TO HIERARCHICAL EVENT BROKER NETWORKS.
In *Proceedings of the Second International Conference on Software and Data Technologies - PL/DPS/KE/WsMUSE*, pages 99-105
DOI: 10.5220/0001338000990105

2 BACKGROUND

Publish/subscribe paradigm is a new communication paradigm where a set of clients, in a distributed environment, communicate asynchronously through a notification service. The clients are connected through an overlay network of broker nodes, and are classified as publishers, which publish the notification events, and subscribers, which subscribe those notifications. The overlay network of broker nodes and the routing algorithm used ensure the delivery of notifications or publications to appropriate subscribers. There are different kinds of overlay topologies defined in literature (Singh and Cao, 2005; A. Carzaniga, 2001; Pietzuch, 2004; L. Fiege, 2002; Rowstron and Druschel, 2001). In a static hierarchical topology each event broker is connected to a single parent event broker, and every broker receives event notifications from the parent broker. This topology is simple but it is not fault tolerant. We have conducted a survey of the existing publish/subscribe systems and studied their topologies and adaptability to failures of nodes and links and summarized it in Table 1.

Table 1: Adaptation to Failures in Existing EBMs.

EBM	Overlay Topology	Adaptability to Failures
SIENA	Acyclic Graph	Not Adaptable
REBECA	Acyclic Graph	Not adaptable
HERMES	General Graph	Adaptable to Rendezvous Node Failures
MEDYM	General Graph	Detects Link Failures
Pastry	General Graph	Adaptable

3 UNDERLAY AWARENESS

The Need for Underlay Awareness The event based middleware systems like SIENA (A. Carzaniga, 2001), REBECA (L. Fiege, 2002), HERMES (Pietzuch, 2004) are not underlay aware. BICON (Madhu and Bellur, 2006) overlay provides an algorithm for the construction of underlay aware availability manifest overlays. Correlation between the underlying physical topology and the overlay topology is required for guaranteeing the performance of the routing algorithm. In an overlay network each node can be a multi degree node, but in the actual physical network there may be one physical link corresponding to multiple overlay links. Figure 1 shows an overlay where each of the broker nodes A,B,C,D have a degree of 3 whereas the corresponding physical nodes do not. Also if the link between the physical nodes 1 and 2 fails then all the overlay links connecting the broker node A fail. Without underlay awareness the routing algorithm would be ignorant of these failures and hence perform incorrectly. A fundamental challenge in using large-scale overlay networks is to incorporate physical level (IP level) topological information in the construction and design of the overlay to adapt to the node and link failures.

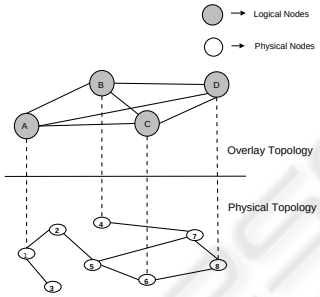

Figure 1: Overlay topology.

Feasibility The environmental information like node quality and path quality may change frequently due to work load and traffic variation, hardware unavailability and software and hardware errors. The overlay topology should periodically monitor the node capabilities and link qualities. A node can obtain the local system parameters with the support of the operating system and the quality of the incident links can be inferred using network monitoring mechanisms such as pathload, traceroute, Sprobe (Caida, 2007) which may be classified as passive observation or active probing (Tang and McKinley, 2004). The routing algorithm for the overlay network can be designed to use the periodically obtained underlay information to make changes to the overlay topology.

Construction of an Underlay Aware Overlay topology We consider the hierarchical or tree structured overlay topology and incorporate underlay awareness features to make it capable of handling a single node and link failure. The resultant topology also provides the flexibility of balancing the network load. In a tree structured overlay topology when a broker node joins the overlay network, it establishes a link with one of the existing brokers. We enhance this standard tree topology by making each new broker node establish two disjoint overlay paths with one of the existing brokers. Each of these overlay paths are ensured to be node disjoint in the underlying physical network. Each overlay link has the information about the physical path that connects the corresponding nodes in the physical network, stored in its incident nodes. This guarantees the existence of a real alternate physical path even in the context of one overlay link failure. Each new broker N chooses an existing broker C as its parent to establish a link. The two disjoint paths con-

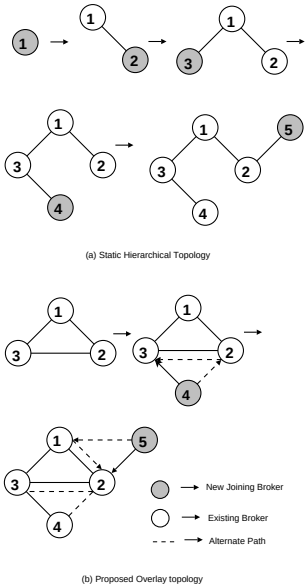

Figure 2: Topology formation.

sist of a direct link from the new broker to the chosen broker C and an alternate path to the chosen broker through a neighbor of the chosen broker A. Figure 2 shows a sample topology. These two paths are ensured to be node disjoint in the physical network. The new joining broker N also establishes a link with the parent broker P of the chosen broker C. The physical path corresponding to this link does not contain the chosen broker C. This will allow the the broker N to reach P when broker C fails. Each broker stores the replica of the routing table and other information of all the children. This is used for handling the failure of child. Also the alternate path from each broker to its parent broker can be used to distribute the network load and thereby obtain a better routing performance.

4 ALGORITHMS

The algorithms for overlay network creation and maintenance are described below. The algorithm ensures that each overlay node has an alternate path to its parent broker node which is node disjoint with respect to the underlying physical network.

Joining of a Broker Node When a new broker node wants to join the overlay network, it executes the *Node_Join()* algorithm. The routines used in this *Node_Join(Node N)* algorithm are described below:

Find_Broker(): Finds a broker node which is nearest to the calling broker node measured in terms of number of IP hops.

Parent(): Returns the parent of the broker.

Neighbors(): Gives the list of neighbors of the broker.

Paths_To_Neighbors(): Gives the physical paths associated with each of the overlay links connecting the neighbors.

Find_Alternate(Neighbors,paths): Finds the neighbor of the chosen parent broker C through which the alternate path is established. This broker is termed as Alternate broker.

Find_Node_disjointpath(grandparent P,parent C): Finds a path to the grandparent P which does not contain C.

Set_Child(child N, Alternate broker A): Sets N as a child and sets A is the neighboring broker to reach N.

Set_GrandChild(Child N,Parent C, Path P): Sets N as the Grandchild and P as the path to reach N bypassing C.

Set_Alternate(child N, Parent C): Sets the broker as the alternate broker for reaching C form N and vice-versa.

The new broker node N joining the overlay network

Algorithm 1 Joining of a broker node

```
Node_Join(Node N)
1. trans_flag=true;
2. Node C=Find_Broker();
3. Node P=C.Parent();
4. Node[] Neighbors=C.Neighbors();
5. Path[] paths=C.Paths_To_Neighbors();
6. Node A=Find_Alternate(Neighbors,paths);
7. Path path2=Find_Node_disjointPath(P,C);
8. C.Set_Child(N,A);
9. P.Set_GrandChild(N,C,path2);
10. A.Set_Alternate(N,C);
11. trans_flag=false;
```

chooses a broker node C, already in the network. The choice is based on the proximity with the new broker node in terms of number of IP hops. The broker node C replies the new node N with the address of is parent, set of its neighbors and the IP paths to all its neighbors. The new node N then establishes two paths with the chosen broker node C which are node disjoint in the underlying IP physical network. One path is the direct path and the other path is through one of the neighbors, A, of the chosen broker node C. The choice of the alternate broker is also based on the proximity in terms of number of IP hops. Finally N finds a path to a node P, the parent of C, which does not contain C. The arrival of new node N is registered at C,A and P. The parent broker marks N as its child and A as the alternate broker to reach C if the direct path from C to N fails. Figure 3 shows the sequence of the operations that occur when a new node joins the overlay network.

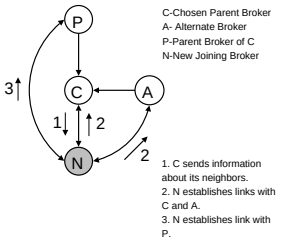

Figure 3: Joining of Broker node.

Leaving of a Broker Node *Node_Leave(Node B)* is performed by a broker when it leaves the overlay topology . The routines used in this algorithm are:
*Connect_To(Node P):*Performs node joining of the calling node with Node P.
*Alternate_To_Child(Child C):*Gives the broker node which forms the alternate broker node to reach C.
*Un_Link(Parent B, Child C):*Unlinks the alternate path from B to C.
*Reset_GrandParent():*Resets the grand parent of the node to the parent node of its parent node.
*Choose_Alternate():*Chooses a new alternate Broker node to reach the parent node.

When a node leaves the overlay network following

Algorithm 2 Leaving of a broker node

Node_Leave(Node B)
1. trans_flag=true;
2. For each child C of B
3. C.Connect_To(B.Parent);
4. NodeA=B.Alternate_To_Child(C);
5. A.Un_Link(B,C);
6. For each Grand Child C of B
7. C.Reset_GrandParent();
8. For each child C for which B is alternate Node
9. Node P=C.Parent();
10. P.Un_Link(B,C);
11. C.Choose_Alternate();
12. trans_flag=false;

steps are to be taken. The overlay topology needs to be maintained and the responsibilities of the leaving broker node needs to be transferred to other nodes in the overlay network. Each node can be a parent, a grand parent and also an alternate broker. Each of the children of the leaving broker node B has to choose the parent of B as their parent and perform *Node_Join()* to the parent of B. Node B notifies all the alternate brokers of each of its children to unlink with B and the corresponding child. All the grand children of B have to choose new grand parents based on the new parent nodes of their parent nodes. Each node C to which B is an alternate broker has to choose a different alternate broker node. Figure 4 represents the sequence of actions that take place when a broker node leaves the overlay network. While the *trans_flag*

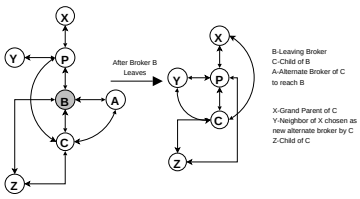

Figure 4: Leaving of Broker Node.

is true the broker does not perform any other operation like accepting a new broker or any other routing related operation like forwarding subscriptions etc. This is to ensure the consistency in a distributed and asynchronous environment. Any new broker trying to contact this node, will find another alternate path as this node does not reply. *Node_Leave()* also includes a priority based deadlock avoidance mechanism which allows nodes lower in the hierarchy to leave first.

Complexity Analysis The complexity of the *Node_Join()* algorithm is $O(degree)$ where degree is the maximum degree of any node in the overlay network. This is because the node joining involves probing all the neighbors of the parent Broker. The complexity of the *Node_leave()* algorithm is $O(degree^2)$. Since this involves all the children of the leaving node also performing *Node_Join()*.

The Data Structures Every broker node in the overlay network stores the path to its parent node, alternate node and grand parent. In addition, the *routing table* which can be a hash table containing entries like (event type, children to be forwarded to) is required. The *topology data table* which contains the table containing the list of children and their associated alternate brokers and the paths to each of them is also stored. The *replicas* of the routing table and the topology data of all its children are also maintained and used to handle the node failure of its children. These replicas are updated periodically by the children of the node.

4.1 Failure Handling

Handling an Overlay Link Failure The overlay link failure can be detected by a child while it tries to send subscription to its parent or by a parent while it tries to send a notification to its child. In case of failure of the direct link, the alternate link is used and the parent node monitors the failed link for a time interval τ and if the link is not up in that interval then the child node has to join its grand parent or the alternate broker.

Handling an Overlay Node Failure When a node is unreachable through both the paths i.e the direct and the alternate path then it is assumed to be failed. When a child C detects that it's parent P has failed then it sends its grand parent the node failure event of P. The failure of node P can also be detected by the parent of P. In both the cases the parent of the failed node performs the *Node_Leave()* operation of P on behalf of P.

4.2 Proof of Correctness

Lemma: (i) In a hierarchical network in which every broker node has two node disjoint paths *x* and *y* to its parent node as well as a path *z* to its parent's parent, such that it is node disjoint from *x* and *y* and does not contain the parent node, every node remains connected to the root node in the event of failure of a single physical node or link.
(ii) Moreover, if a new broker is added to the hierarchical network by forming three physical paths, *x'*, *y'* to its parent and *z'* to its parent's parent such that *x'*, *y'* and *z'* are pairwise node disjoint and *z'* does not contain the parent of the new node, then the resultant network is also tolerant to single node and link failures.
Proof: Consider any non root broker node *b* in the hierarchical network. We show that it remains connected to the root node *r* in the event of failure of a single node *s (s≠b)* or single link *l*.
I. If Node *s* fails
Case 1: *s* is *b*'s parent. Node *b* still has a path *z* to node *s*'s parent, which does not contain node *s*, and the network being hierarchical, the path from parent of *s* to *r* does not contain *s*.
Case 2: Node *s* is *b*'s parent's parent. Node *b* has paths *x* and *y* to its parent, which is connected to its own parent's parent via a path independent of *s*, and as the network is hierarchical, all the way to the root.
Case 3: Node *s* is not a broker node. If node *s* does not occur in the physical path from *b* to *r* then its failure cannot affect the connectivity of *b* to *r*. If it exists in the path, then let (b,$p_1, p_2 \ldots p_n$, r), be the path along parent nodes from *b* to *r*. For any overlay edge p_i, p_{i+1} along this path, the failure of *s* does not affect the connectivity of p_i to p_{i+1}, as there is an alternate physical path which does not contain *s*.
II. If Link *l* fails
If link *l* does not occur in the physical path from *b* to *r* then its failure cannot affect the connectivity of *b* to *r*. If it exists in the path, then let (b, $p_1, p_2 \ldots p_n$, r), be the path along parent nodes from *b* to *r*. For any overlay edge p_i, p_{i+1} along this path, the failure of *l* does not affect the connectivity of p_i to p_{i+1}, as there is an alternate physical path which does not contain link *l*.

5 SIMULATIONS

Experimental Setup The experimental setup consists of a simple network simulator for event based middleware. The Simulator, developed in java, models all the basic network features like delay, bandwidth and loss of data. The application data is converted into simulation events and kept in a simulation event queue and then processed according to their attached time stamps. The time stamp is assigned according to the delay for data to get transferred from the source to destination in a real network. The Simulator can generate performance data like the data traffic, control traffic and the processing load. The Simple Event Based System network Simulator uses BRITE (Boston university Representative Internet Topology gEnerator) (Alberto Medina and Byers, 2001) to generate Internet topology. The overlay topology is formed from the BRITE generated physical network topology by choosing the overlay nodes from the physical nodes. The delay and bandwidth are calculated over the physical path that represents the overlay link. An AS-level physical topology of 10000 nodes, generated by BRITE using Waxman generation model is used for overlay topology construction. The bandwidths for the links are uniformly distributed.

Table 2: Simulation Parameters.

Number of events	10000
Number of Event Clients	1000
Number of Event Brokers	100
Distribution of Clients	Uniform
Average Message size	50 bytes
Failure distribution	random

For experimentally verifying the advantages of an underlay aware overlay topology we implemented SIENA(Scalable Internet Event Notification Architecture)(A. Carzaniga, 2001) which has hierarchical topology and extended it with underlay awareness using the above discussed algorithms. The percentage of messages delivered to subscribers in the face of increasing number of link and node failures is monitored for underlay aware SIENA and unmodified SIENA. The redundant paths between parent and child node in underlay aware Siena are used to reduce link stress on heavily loaded links. The experimental results of this load balancing are plotted for underlay aware SIENA from the simulated results. Table 2 shows the simulation parameters.

Implementation Results The results shown here represent the effect of underlay awareness on fault tolerance and the effect of balancing network load in underlay aware SIENA over underlay unaware SIENA. In SIENA the load on each overlay link increases when the amount of data traffic thus increasing the average delay. In Underlay aware SIENA the load is distributed among the two different paths between a child node and its corresponding parent node. The data load considered here is generated only by publications. The results are plotted by taking the average in twenty simulation runs. Figure 5 shows the

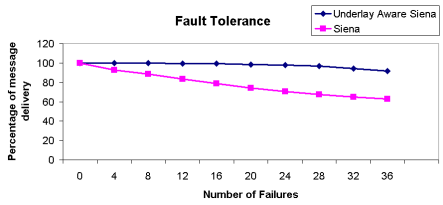

Figure 5: Message delivery in the presence of failures.

percentage of notifications delivered in the presence of varying number of faults, for underlay aware and unmodified SIENA, and shows that the message delivery percentage is higher in underlay aware SIENA in comparison with unmodified SIENA, and the difference between them increases when the number of failures increase. Figure 6 shows the average delivery

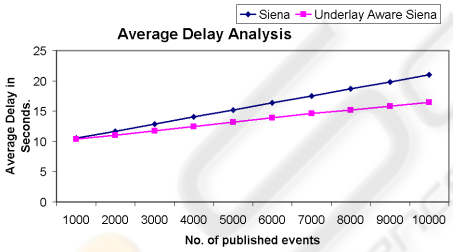

Figure 6: Delivery latencies with increasing events.

latency per event. In underlay aware Siena, the alternate routing paths available are used to reduce the waiting time for event forwarding. This causes a reduction in the message delivery latencies over unmodified Siena. The average stress (load) on the links in Kilobytes and the standard deviation of the link stress (load) were studied with increasing number of events from 1000 to 10000. Figure 7 shows that the average link loads are less in underlay aware Siena, due to the load balancing effect of alternate routing paths, and the increase in the number of overlay links. Figure 8 shows that in underlay aware Siena, the standard deviation of the loads on different links is comparable to that of unmodified Siena, indicating that the added

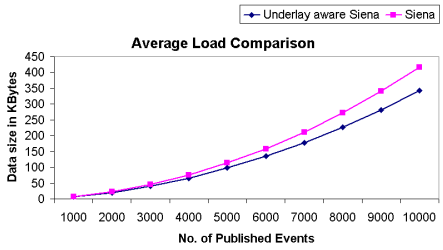

Figure 7: Average link stress comparison.

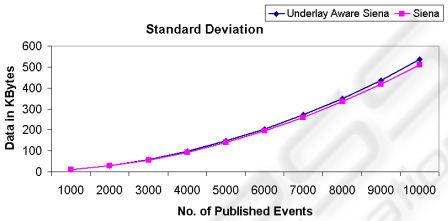

Figure 8: Comparison of standard deviation of Link Stress.

links are also loaded to a degree comparable to existing links.

6 CONCLUSIONS

In this paper, we demonstrate that hierarchical overlay networks can be improved by enhancing them with underlay awareness information. We outlined algorithms to enhance the Siena overlay by adding node disjoint paths from the newly joined broker to its parent and grandparent, making the overlay tolerant to single node and link failures and proved our algorithm to be theoretically correct. We also proposed that the redundancy in paths so achieved, could be used for reducing message delivery latencies by using alternate routing paths. The correctness of our hypothesis is corroborated by our simulation results which show that with increasing number of faults, underlay aware Siena shows much better event delivery performance than unmodified Siena. The results also show that with increasing number of messages, underlay aware Siena gives less message delivery latencies and better load balancing than unmodified Siena. Our future work includes construction of underlay aware overlay networks which can tolerate a larger number of node and link failures for general topologies.

REFERENCES

A. Carzaniga, D. S. Rosenblum, A. L. W. (2001). Design and evaluation of a wide-area event notification ser-

vice. *ACM Trans. on Computer Systems*, 19(3):332–383.

Alberto Medina, Anukool Lakhina, I. M. and Byers, J. (2001). BRITE: Universal topology generation from a user's perspective. Technical Report BUCS-TR-2001-003, Boston University.

Caida (2007). Performance measurement tools taxonomy. *http://www.caida.org/tools/taxonomy/performance.xml*.

L. Fiege, G. M. (2002). *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, TU Darmstadt Germany.

Madhu, K. and Bellur, U. (November 2006). An Underlay Aware, Adaptive Overlay for Event Broker Networks. In *Proceedings of the 5th International workshop on Adaptive and Reflective Middleware (ARM '06), Melbourne*.

Pietzuch, P. R. (2004). Hermes: A scalable event-based middleware. Technical Report UCAM-CL-TR-590, University of Cambridge.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 3rd International Conference on Middleware, Middleware'01*, pages 329–350, Heidelberg.

Singh, J. P. and Cao, F. (2005). MEDYM: Match-early and dynamic multicast for contentbased publish-subscribe service networks. *Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW 05)*, 4(3):370–376.

Tang, C. and McKinley, P. K. (2004). Underlay-aware design of overlay topologies and routing algorithms. Technical Report MSU-CSE-04-09, Department of Computer Science and Engineering,Michigan State University, East Lansing, Michigan 48824.