A Distributed Algorithm for Underlay Aware and Available Overlay Formation in Event Broker Networks for Publish/Subscribe Systems

Madhu Kumar S D, Umesh Bellur
Kanwal Rekhi School of Information Technology, IIT Bombay
Powai, India -400076
E-mail {madhu, umesh}@it.iitb.ac.in

Abstract
Event broker networks are basically overlay networks formed over the underlying physical network. In modern distributed applications, ensuring high availability in the face of the runtime failures is a major issue. This paper presents an asynchronous distributed algorithm for constructing and maintaining an underlay aware overlay which ensures high availability in the presence of node and link failures in the underlying physical network. We prove theoretically that our algorithm is correct. The time complexity of the algorithm is estimated to be \(O(\text{diameter} \times \text{degree})^2\) of the network and the message complexity is \(O(\text{diameter} \times \text{degree})\). A model for availability of an underlay aware overlay network and a classification of available overlays are the other important contributions of this paper.

1. Introduction
In a distributed event processing system, event publishers and event subscribers are clients which are connected through a set of event brokers. In a multi-broker publish-subscribe system these event brokers are connected in a peer-to-peer fashion[1] to form an overlay network over the underlying physical network (underlay)[4].

An overlay network is a logical abstraction of the underlying physical network. An overlay network can be represented as a graph \(G=(B,L)\), where the vertex set \(B\) is the collection of overlay nodes, and the edge set \(L\) is the collection of paths between overlay node pairs that are determined by the application or the event broker network.

The performance of an overlay network is tightly linked to that of the underlay. In order to maintain a specific quality of service and to be robust, the event based middleware must adapt to changing conditions both in the application requirements and changes in the computation and communication infrastructure provided by the underlying physical network. Physical node failure and link failure are crucial environmental changes, as they may block the communication to clients, if there are no alternate paths readily available or efficiently computable in the network. The overlay needs to be underlay aware so that it adapts itself to the faults or failures that occur in the underlay. The main focus of this paper is on improving availability in event based systems by providing an overlay network that is adaptable to node and link failure triggers[3]. In this paper we propose a distributed algorithm for formation and maintenance of an underlay aware, available overlay for the event broker network.

The paper is outlined as follows. In Section 2 we present our model of availability of event broker networks. Section 3 discusses the concept of underlay aware overlay networks and integrates the availability and underlay awareness requirements of overlay construction method. The distributed algorithm for overlay formation is described in Section 4. Section 4 also describes the algorithms for overlay maintenance, the complexity analysis of the algorithms and the proof of correctness for our algorithm. The related work in this area is summarized in Section 5. Section 6 outlines our future research plans and concludes the paper.

2. Modeling Availability
Availability of a system is defined in standard literature as the fraction of time for which a system is up ( usable). In [2] the availability of a general network is discussed and the degree of availability of a network is defined and related to the node connectivity of a network. According to [2] a network with an availability of degree \(k\) has probability of non-availability of a path between any two nodes as \((1-(1-p)^n)^k\), where \(n\) is the average path length, and \(p\) is the probability of failure of a link. Such a network should have a topology that guarantees the existence of \(k\) independent paths between any pair of nodes.

![Figure 1. A physical network](image)

In this paper, we extend the definition of availability and model it for overlay networks. In the context of event broker networks, the availability of the overlay network is the primary concern and the availability of the physical network is of consequence only to the extent of the influence it has on the availability of the overlay network. To illustrate our point, consider Figure 1 and Figure 2. The availability of the physical network in Figure 1 is constrained to 1 as there is only one physical path.
between nodes p and q in the physical network, but the broker network (shown in bold) illustrated in Figure 2, based on the same physical network, has an availability degree of 2 as there are 2 node disjoint paths between every pair of overlay nodes. Whitney’s theorem [13] states that the node connectivity of a graph cannot be more than the minimum degree among all the nodes. Hence a physical network cannot have a connectivity and thereby a degree of availability more than the degree of its lowest degree node, which is 1 in all probability.

Figure 2. Overlay network

Our assertion is that the broker network can be given a higher availability i.e., a higher connectivity in the physical network, by careful selection of broker nodes and a systematic overlay construction method. In this paper a distributed algorithm for the construction of high availability overlay networks is outlined.

We start by modeling the availability of an overlay network. We define availability of three entities (i) the physical link (ii) the link between two overlay nodes, and (iii) the overlay network

The availability of the physical link (A\textsubscript{link}) between two nodes is defined as the probability that the link is up. A\textsubscript{link} = 1- P\textsubscript{link}, where P\textsubscript{link} is the probability that the link is down. The availability of the physical path (A\textsubscript{path}) between two overlay nodes is defined as the probability that the path is up, which is the product of the availability of all the physical links (A\textsubscript{i}) in the path.

A\textsubscript{path} = \prod_{i=1}^{n} A\textsubscript{i} = \prod_{i=1}^{n} (1 - P\textsubscript{i}), where P\textsubscript{i} is the probability of failure of the i\textsuperscript{th} link in the path and n is the length of the path in terms of the number of physical links in the path.

The availability of the overlay network (A\textsubscript{nw}) is defined as the least availability of a path among the paths between every pair of overlay nodes in the network.

A\textsubscript{nw} = \min ( A\textsubscript{path(i,j) } ) over all overlay node pairs i and j, where path(i,j) is the path between the overlay nodes i & j such that it has the highest availability among all physical paths between i & j.

We define the extended diameter ‘d’ as the maximum physical path length between any pair of nodes in the network. d = \max \{d\textsubscript{i,j} | i and j are overlay nodes & d\textsubscript{i,j} is the length of the shortest physical path from node i to j\}.

Let “w” be the weakest physical link in the overlay network. w = \{i,j\} i and j are physical nodes and (i,j) is a part of a physical path from a broker node to another broker node). Let A\textsubscript{w} be the availability of w. A\textsubscript{w} = 1- P\textsubscript{w} where P\textsubscript{w} is the probability of failure of w. Consider any pair of overlay nodes A and B.

A\textsubscript{AB} => (1- P\textsubscript{w})\textsuperscript{d} => (A\textsubscript{w})\textsuperscript{d}

If there are k node disjoint paths in the network, between A and B then, availability of the path between A and B is

A\textsubscript{AB} = 1- Probability that k paths are down

= 1- \prod_{i=1}^{k} (1- A\textsubscript{path,i}), where path \textsubscript{i} is the i\textsuperscript{th} path.

=> 1- \prod_{i=1}^{k} (1- (A\textsubscript{w})\textsuperscript{d}) => 1- (1- (A\textsubscript{w})\textsuperscript{d})\textsuperscript{k} = k (A\textsubscript{w})\textsuperscript{d} (neglecting the higher order terms)

≈ Connectivity * Availability of a single path.

Hence, the scaling of availability due to higher connectivity is connectivity of node disjoint paths. On the basis of the above, we define a fixed availability network of degree of availability k as one that guarantees k physically node disjoint paths between all pairs of broker nodes.

In order to construct and maintain such an overlay network, the nodes of the broker network require information about the underlying physical network, called underlay awareness. In the next section we discuss the underlay awareness called for by our available overlay network construction methods and present two different perspectives of available overlays.

3. Underlay Awareness

Clearly, all aspects of the physical network cannot be made known to the overlay network due to the distributed and dynamically changing nature of the underlying network. Hence an overlay construction methodology has to be outlined with details of the physical network it needs to know and has to be analyzed taking into account the overheads associated with the acquisition and storage of this knowledge. [3] provides a taxonomy of overlay networks and classifies underlay aware networks as proximity aware and quality aware. Proximity aware overlays are aware only of neighborhood information. Quality aware networks are aware of other aspects of the links from the node, such as disjointedness and performance parameters. We describe algorithms for an underlay quality aware overlay. The overlay nodes gather and store information about the underlying path for the overlay links originating from them including information about the nodes in the path and the node overlap in the overlay links from the node. But, this information is required only about nodes that ‘matter’ in the path, not all.

We present a classification of the nodes in the physical graph to illustrate our point.

3.1 Node Classification

All the nodes in the graph may be classified as follows – overlay nodes, expander nodes, connector nodes and trivial nodes as illustrated in Figure 3.

1. Overlay nodes: Overlay nodes are the nodes selected to be brokers. They should satisfy the criterion that their physical degrees are greater than or equal to k, where k is the degree of connectivity required.
2. Expander nodes: Expander nodes are nodes (non brokers) with degree more than 2. They have the potential to exist in more than one physical path. Hence any overlay link that contains such nodes can overlap with other overlay links, which contain the same expander node. Overlay nodes should remember such nodes that exist in their overlay links.

3. Connector nodes: These nodes have degree equal to 2. They cannot be a part of more than one path between two higher degree nodes and are hence "collapsible" as far as disjointedness studies are considered, unless their degrees get changed.

4. Trivial or client nodes: These correspond to "pendant" nodes in graph theory terminology and have a degree of just 1. They cannot be guaranteed a connectivity more than 1, and can serve only as client nodes, if at all, in an overlay of connectivity k higher than 1. Hence these nodes can be ignored for overlay construction.

3.2 Link numbering

The links associated with every node in the physical network are ordered and specified by their link number for the node. For example, a node with degree 5 has a fixed ordering 1, 2, 3, 4, and 5 for its outgoing links. The ordering is fixed based on a function of link type and bandwidth, and uniquely determinable for every node. Figure 3 illustrates this.

3.3 Underlay information

The overlay node stores a sequence of three tuples for each of its overlay links. The three elements of the tuple are (node id, link1, link2), where node id identifies the physical nodes in the sequence, and link1 and link2 are the link numbers of the links of the node. The end nodes in this sequence correspond to broker nodes and the intermediate nodes are expander nodes. Connector nodes are not included as they cannot possibly overlap. Pendant nodes cannot be a part of such a path. This information is used by the broker nodes for overlay construction.

3.4 Classification of Available Overlays

A given physical network can form the basis for many overlays networks. The availability guaranteed by an overlay can be manifested in two ways. It may be explicit or manifest, which means that different paths that are node disjoint in the overlay also map to physically node disjoint paths. On the other hand, an overlay’s availability may be implicit or latent, which means that node disjoint overlay paths are not necessarily node disjoint at the physical level, but the existence of different disjoint paths at the underlay level for the overlay node pairs is guaranteed by the overlay. On this basis we classify available overlays as Manifest and Latent.

Availability Manifest Overlay An availability manifest overlay can be defined as follows. Let G=<V,E> represent a physical network, and A=<B,L> be an overlay network based on it, where B ⊆ V and L consists of overlay links which are paths in G between two vertices that also belong to B. A represents an availability manifest overlay network if for any two nodes b1 and b2 belonging to B, if (b1, x1... xk, b2) and (b1, y1... yk, b2), where x1, xk, and y1... yk are brokers, be two overlay paths between b1 and b2 such that {x1...xk} and {y1...yk} are disjoint sets, then the two paths are also disjoint at the underlay level. In other words, the set {p1,x1,p2,x2... pk,xk} and the set {q1,y1,q2,y2... qk,yk} where p’s and q’s represent the physical nodes along the path, are also disjoint sets. This is illustrated in Figure 4.

Figure 4. Availability Manifest Overlay

Given a physical network and an overlay graph, constructing an availability manifest overlay on the physical graph can be directly mapped to the fixed, node disjoint subgraph homeomorphism problem [12], which is NP-complete for general subgraphs (overlays). However for simple subgraphs, such as triangles and two node disjoint paths between 2 nodes [12], [13], it has linear time solutions. Manifest overlay creation requires a complete knowledge of the underlay, which is a practically difficult proposition in the context of event broker systems. Moreover, availability manifest overlays are not absolutely necessary if availability is the only requirement. Hence, we focus our study to availability latent overlays.
two broker nodes, k node disjoint paths are guaranteed. Overlay links can have node overlaps, but between any node disjoint paths in the physical network between the nodes corresponding to b₁ and b₂. Figure 5 illustrates this. Hence in an availability latent overlay, two distinct overlay links can have node overlaps, but between any two broker nodes, k node disjoint paths are guaranteed.

4. Overlay Formation Algorithm

Our focus is on providing algorithms for the construction of fixed availability, latent overlay networks of degree k, on a physical network of computers, for providing an event based broker network to running applications working on the publish subscribe paradigm. This is typical distributed computing environment. The distributed algorithms we describe are asynchronous algorithms.

The model of the underlying network for our algorithm assumes the usual characteristics of the distributed environment [14], such as unknown network topology. For such an overlay the broker software can be run only on machines having a physical degree of k, i.e., k independent network connections to other computers. As high values of k are rare, we concentrate on providing algorithms for k=2 and k=3 and present an algorithm for k=2 in this paper. The concept of the algorithm is also applicable to overlay networks with higher degrees of availability. The algorithms are based on a level of underlay awareness, initially statically obtained and later dynamically gathered and updated.

Initially two broker nodes are selected as stellar nodes. These are selected in such a manner that there exist two node disjoint physical paths between them, ascertained manually using global information. For higher degrees of availability, bigger stellar broker networks would be required. Broker software is installed and run on these nodes. The underlay information consisting of the path information in the form of three tuple sequences of expander nodes in the paths is stored at these nodes. The path information is gathered by the messages in the algorithm as they proceed from node to node in the physical network. The stellar nodes remain a part of the broker network as long as the broker network exists. The id of these nodes is later made known to every aspiring broker node, and the initiation process of new brokers involves trying to establish an overlay link to these stellar nodes. However, these requests may be intercepted and processed by other broker nodes in the path to reduce the load on stellar brokers and also to preserve locality and underlay awareness.

Figure 6 illustrates the stellar broker node network. It may be noted that every expander node in the initial stellar overlay has a connectivity of two to every broker node, and other expander nodes as well.

---

**Figure 5. Availability Latent Overlay**

_A availability latent overlay of degree of availability k on a graph <V, E> is defined as A = <B, L>, where B is a subset of V, and L is a set of links which represent paths in the physical network, and for any (b₁, b₂) belonging to V, there exist k node disjoint paths in the physical network between the nodes corresponding to b₁ and b₂. Figure 5 illustrates this. Hence in an availability latent overlay, two distinct overlay links can have node overlaps, but between any two broker nodes, k node disjoint paths are guaranteed._

**Figure 6. Stellar Broker Network**

---

**Table 1. List of Messages in the Distributed Algorithm**

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Join_Stellar_Broker(B)</strong></td>
<td>This is sent by a node B wishing to join the broker network</td>
</tr>
<tr>
<td><strong>New_Contact(R,B,path, list)</strong></td>
<td>This is a message sent by a broker R which intercepts the Join_Stellar_Broker message, or by the stellar broker (in case the Join_Stellar_Broker was not intercepted by any other broker) to the broker B which wishes to join. It contains the knowledge of the path by which the message reached R from B, and also a list of other brokers and expander nodes, that B should try to contact, to get two node disjoint paths to the broker network.</td>
</tr>
<tr>
<td><strong>Establish_Broker_Link(B, A, path)</strong></td>
<td>This is a message sent by a broker B to broker A to establish an overlay link between them with the underlay information in path.</td>
</tr>
<tr>
<td><strong>Confirm_Broker_Link(A, B, path)</strong></td>
<td>This is a message sent by a broker A which receives the establish broker link message from B, meant for A itself or some other broker X to which B had sent the Establish_Broker_Link message, if A was an expander node on the path from B to X, and subsequently became a broker. Then the broker link is established between A and B, instead of X and B.</td>
</tr>
<tr>
<td><strong>Check_Path(A, B, nodelist)</strong></td>
<td>This is sent by a node A to a node B to check whether the path by which the Check_Path message reaches node B has more/less expander nodes than listed in nodelist (nodes in path). This is a periodic refresher message sent between overlay nodes, for the purpose of dynamically monitoring changes in the physical network.</td>
</tr>
<tr>
<td><strong>Leave_Stellar_Broker(A, nodelist)</strong></td>
<td>This is sent by a node B to inform all the overlay neighbours that B has ceased to be a broker. nodelist contains the list of neighbours of B.</td>
</tr>
<tr>
<td><strong>Bye (A, B)</strong></td>
<td>This message is sent by a neighbouring broker A or stellar node to allow B to stop its routing activities in the overlay.</td>
</tr>
</tbody>
</table>

The routines described here are:

1. **Make_Broker(B)**  This is executed when a node B wants to join the broker network.
2. **Leave_Broker(B)**  This is executed by a broker B to leave a broker network.
3. **Execute_Broker(B)**  This is a routine executed throughout by a broker which is a part of the network.
Messages sent/received by the broker nodes for the construction and maintenance of the overlay are described in Table 1. This algorithm is based on the expansion lemma of Whitney’s theorem[13] which states that if G is a k connected undirected graph then a graph G’ obtained by adding a new node x and adding k distinct edges from x to k distinct nodes of G is also k-connected, as illustrated in the Figure 7. The stellar node overlay network is a known network of connectivity 2. Any nodes that join this network through node disjoint links would inductively result in networks of connectivity two.

Figure 7. A new node joining a k-connected graph

![Diagram](image)

Algorithm 1 Make_Broker(B)

1. if (degree(B)==1) then exit. /* does not qualify*/
2. send Join_Stellar_Broker(B) from all the links of B to the stellar brokers.
3. while (not timeout)
   4. if receive(New_Contact(A, B, path, list)) then store (path, list) in pathset
5. if pathset is empty
6. then exit /*brokers unreachable*/
7. else select two node disjoint paths from pathset
8. if found
9. then send(Establish_Broker _Link(B, A, path))
10. else if (suggested_paths present in the pathset)
11. then send (Join_Stellar_Broker(B)) through suggested path in list of one New_Contact message
12. delete suggested path
goto step 3
13. else exit */ fails to find node disjoint paths*/
14. while (not timeout)
   15. receive(Confirm_Broker_Link(A,B,pathlist))
16. start Execute_Broker(B)
17. start routing activity

The Make_Broker() routine is executed by an aspiring broker B to connect to a broker network which is already guaranteed to have a connectivity of two. Make_Broker(B) executed by an aspiring broker B enables it to find two node disjoint paths to different nodes in the existing broker network, and it starts the process of establishing the links. Different cases in the algorithm are illustrated below. As shown in Figure 8, B’s Join_Stellar_Broker message sent via its two links may contact the brokers p and q on node disjoint paths through expander nodes c and d respectively. Nodes p and q respond with path and suggested brokers. Node B on getting these finds the two disjoint paths and decides to establish these links. On the other hand, suppose from both links of B, node p is contacted through expander node c, as shown in Figure 9.

Figure 8. Broker Joining

![Diagram](image)

Algorithm 2 Leave_Broker(B)

/* this is executed by a broker B to leave a broker network.*/
1. leaving = true
2. safe = false
3. over = false
4. while (not safe) wait;
5. stop(Execute_Broker(B))
6. send(Leave_Stellar_Broker (B, brokerlist)) through all the links of B
7. while (not timeout or not over)
   8. for each node in neighbourlist
   9. if not receive(Bye(A, B)) wait

Figure 9. Broker joining using suggested path

As p returns path as well as suggested nodes, b, d and a to contact q, B again sends a Join_Stellar_Node() through the suggested paths and finally discovers the path to q via b. If such a path had not been present, ultimately the suggestions list would become empty and Make_Broker would terminate with a failure because it is not possible to provide a connectivity of 2 to the broker network if B is also made a broker.

Now let us consider a third case, arising when there are more broker nodes in the broker network. Figure 10 illustrates the case in which the Join_Stellar_Broker message is intercepted by a broker on the path and subsequently the overlay link is established between the new node and the intercepting broker.

Figure 10. Broker joining non stellar broker

![Diagram](image)
A broker that wishes to stop being a broker executes a
\textit{Leave\_Broker}(B). It still remains a part of the physical
network. This message is also intercepted and replied by
the neighbouring broker which has B as its overlay
neighbour with a \textit{Bye} message. If this node had already
ceased to be a broker the \textit{Leave\_Broker} message travels
up all the way to the stellar node which replies with a \textit{Bye}
message.

We can visualize the process of a new broker node trying
to join the broker network as an ancient ship trying to
reach land by following a star. The ship stops when it
finds land. The new broker node tries to reach the stellar
brokers in the sea of IP addresses in the physical network,
in the process stops its search when it finds another
broker which is already a part of the network.

4.1 Maintenance of the Overlay

Every overlay node executes the \textit{Execute\_Broker()} for
overlay maintenance. Broker addition was discussed
in the previous section. It also holds for a node already
existing in the broker network as an overlay node,
turning a broker also.

\begin{algorithm}
\caption{Execute\_Broker(B)}
\begin{algorithmic}
\STATE 1. \textbf{while} (true) \textbf{do} /* execute until terminated*/
\STATE 2. \textbf{if} receive(\textit{Join\_Stellar\_Broker}(A) and leaving==false) \textbf{then}
\STATE 3. \textbf{intercept\_message}
\STATE 4. \textbf{send} \textit{New\_Contact}(B, A, path, list)
\STATE 5. \textbf{enqueue}(B,A,path,list)
\STATE 6. \textbf{if} \textbf{receive}(\textit{Establish\_Broker\_Link}(A, B, path)) \textbf{then}
\STATE 7. \textbf{send} (\textit{Confirm\_Broker}(A, B, path))
\STATE 8. \textbf{dequeue}(A, B, path, list)
\STATE 9. \textbf{add} overlay\_path (A,B,path)
\STATE 10. \textbf{if} (queue is empty) \textbf{safe=true}
\STATE 11. \textbf{if} receive(\textit{Leave\_Stellar\_Broker}(A, brokerlist)) \textbf{then}
\STATE 12. \textbf{add} new overlay\_links to the nodes in
\textbf{brokerlist}
\STATE 13. \textbf{send} \textit{Bye}(A)
\STATE 14. \textbf{for} all brokers x in neighbourlist of B
\STATE 15. \textbf{send} (\textit{Check\_Path} (B, x, nodelist1))
\STATE 16. \textbf{if} \textbf{receive}(\textit{Check\_Path}(x, B, nodelist2) and
\textbf{nodelist2 \neq nodelist1}) \textbf{then}
\STATE 17. \textbf{update\_path}( )
\end{algorithmic}
\end{algorithm}

When the \textit{Join\_Stellar\_Broker} message is received by the
brokers on the links from that node, they respond with
\textit{New\_Contact} message and new broker links get
established.

If in the time interval between sending \textit{New\_Contact} and
receiving an \textit{Establish\_Broker}, a new broker joins on the
path, then the \textit{Confirm\_Broker} is sent by the new broker
node, and the overlay link is established only to that path.

When the broker gets a \textit{Leave\_Broker} message meant for itself, it replies with a \textit{Bye} and establishes
overlay links to the neighbour of the neighbours, which is
present in the list sent by the leaving broker. Otherwise, it
ignores the message and lets it proceed further up the path
to the stellar node.

In the overlay network every node checks its links
information periodically to confirm the underlay
knowledge. If new expander nodes are added in the path
(say, by adding a new link to a physical node), then that
knowledge is updated in the overlay node, so that further
node sharing information among links can be correctly
assessed. The \textit{Check\_Path} (A, B, brokerlist ) message
does this. The receiving node compares nodelist with the
path by which the message actually comes and updates it.
While checking the overlay information the old broker
links may be modified by the old broker nodes, and new
links to the new broker added.

4.2 Proof of correctness

We start by giving an inductive proof for the correctness
for the algorithm.

\textbf{Lemma:} An overlay graph $N'$ obtained by the concurrent
and sequential executions of the \textit{Make\_Broker()} and
\textit{Leave\_Broker()} procedures by different physical nodes in
a given physical network $G$ is a fixed availability latent
overlay of availability degree 2, i.e., any broker node in $N'$
has two node disjoint paths to every other broker node in $N$.

\textbf{Base:} The initial stellar broker network has two node
disjoint paths between pairs of broker or expander nodes
belonging to the paths between stellar brokers.

\textbf{Inductive step:} If the broker network $N$ has two node
disjoint paths between every broker or expander nodes,
then a broker network $N'$ obtained by the execution of
\textit{Make\_Broker}, or concurrent execution of \textit{Make\_Broker()} by
two different nodes, or by \textit{Leave\_Broker()} executed by
a non stellar node, or concurrent \textit{Leave\_Broker()} executions
by non stellar nodes, also has two node disjoint paths
between every two broker or expander nodes which form a part
of $N'$.

\textbf{Proof:} We prove the inductive step by considering the
execution of the routines.

\textbf{Case 1:} Execution of a \textit{Make\_Broker()} by a node $b$.

(i) If $b$ has degree 1 or it is not connected or too far off
(time-out) from $N$, then no change is made to $N$, hence
$N'=N$.

(ii) If $b$ gets at least two replies that include node disjoint
paths to two different expander or broker nodes in $N$, $x$
and $y$, then it joins with those links. Hence, as $x$ and $y$
are a part of a network which has two node disjoint paths
to every other expander/broker node, and $b$ joins to
two different such nodes, $b$ also has two node disjoint
paths to every node in $N$, by the expansion lemma of
Whitney's theorem [13]. Hence $N'$, which is $N \cup \{b\}$
satisfies the availability requirement.

(iii) If it does not get two disjoint replies, it uses the
information given by one of the replying brokers, which
contains information about other expander nodes and
broker nodes to contact. Hence if the node is able to
establish a node disjoint link to that expander node, then
also $N'$ satisfies the availability criteria.

(iv) If it is not possible, the node $b$ does not join, so $N'=N$.

\textbf{Case 2:} Concurrent execution of \textit{Make\_Broker()}: Suppose
nodes $b_1$ and $b_2$ execute \textit{Make\_Broker()}.
(i) If one or both have lesser degree or get timeout then the proof enumerated in Case 1 holds.
(ii) If nodes $b_1$ and $b_2$ get two node disjoint paths each and these are to different broker/expander nodes then they can simultaneously join $N$, and the proof for case 1 holds.
(iii) If two nodes $b_1$ and $b_2$ select paths that are mutually disjoint but overlap with each other’s paths, it could happen in different ways as we enumerate.

1. Both join to same expander nodes/node. Both still get node disjoint paths to each other and to the rest of the network by Whitney's theorem expansion lemma [13]. This is illustrated in Figure 11.
2. Both join same nodes, but with overlapping paths we can observe from Figure 12 that they still have two node disjoint paths.

(i) If the brokers contacted by $\text{Make_Broker}()$ are different from the node executing $\text{Leave_Broker}()$ then they would each execute correctly individually as proved in cases 1 and 3.
(ii) If the broker executing $\text{Leave_Broker}()$ is one of the nodes that is contacted by the node executing $\text{Make_Broker}()$ then if the contact is made after it starts executing $\text{Leave_Broker}()$ then the $\text{Join_Stellar}$ message is simply forwarded, as it would have executed $\text{leaving} = \text{true}$.

**Time Complexity analysis:** As the algorithm is based on routing to the stellar node in the worst case, the maximum diameter of the network would determine the upper bound on the time for sending and receiving messages. Comparing node disjointedness in the obtained paths is $O(\text{number of paths}^2 \cdot \text{path length}^2)$. As the number of paths is determined by the degree, and path length by the diameter, node disjointedness can be found in $O(\text{degree} \cdot \text{diameter}^2)$.

---

**Figure 11. Brokers joining same nodes**

**Figure 12. Brokers joining with overlapping paths**

3. One of the nodes lies in the path selected by the other. Figure 13 illustrates and shows that there are still 2 node disjoint paths.
4. Both nodes lie in paths selected by each other. Figure 14 shows that still 2-connectivity exist in broker network.

**Case 3:** $\text{Leave_Broker: Leave_Broker()}$ calls, whether executed simultaneously or individually, do not affect the underlying network connectivity, as it is just a change in status.

**Case 4:** $\text{Leave_Broker()}$ and $\text{Make_Broker()}$ execute concurrently.
Message complexity analysis: Message Complexity is limited by (Degree * Maximum diameter).

5. Related Work
Event based Middleware is a well studied area and a number of research projects have been carried out in this field. Our survey on the EBM research projects and study of the type of overlay networks used and their support for availability is summarized in Table 2.

<table>
<thead>
<tr>
<th>EBM</th>
<th>Overlay topology used</th>
<th>Support for Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siena</td>
<td>Acyclic peer to peer</td>
<td>No support</td>
</tr>
<tr>
<td>Elvin</td>
<td>Star</td>
<td>No support</td>
</tr>
<tr>
<td>JEDI</td>
<td>Tree</td>
<td>No support</td>
</tr>
<tr>
<td>Gryphon</td>
<td>Hierarchical</td>
<td>Not Underlay aware but fault tolerant</td>
</tr>
<tr>
<td>REBECA</td>
<td>Symmetrical, acyclic tree</td>
<td></td>
</tr>
<tr>
<td>Hermes</td>
<td>General Graph</td>
<td>Underlay (proximity) aware and fault tolerant</td>
</tr>
<tr>
<td>MEDYM</td>
<td>General Graph</td>
<td>Underlay (Proximity) aware and detects faults</td>
</tr>
</tbody>
</table>

Table 2: Broker networks used in EBM projects

The research projects studied here are Hermes [1], Siena [7], JEDI [6], REBECA [8], Elvin [5], MEDYM [10] and Gryphon [9]. Pastry [11] uses a distributed hashing technique for the overlay formation and is not underlay quality aware. RON[15] outlines a strategy for recovering from failure by obtaining alternate paths through different overlay paths. Our survey indicates that the current research on event based middleware has not given sufficient attention to available overlay formation. Also distributed algorithms do not exist at present for the creation and maintenance of available overlays for broker networks. [2] Presents a formal definition of availability of overlay networks and presents BICON, an available network of degree 2. BICON is an availability manifest overlay of fixed availability of degree two. However, as the manifest availability criterion is difficult in a practical network, we explore the latent availability solutions.

6. Conclusion and Future work
In this paper we provide an analysis of the availability of broker networks, and study the theoretical formulation of the problem of determining the availability of a network. We provide a strategy for simplifying the analysis of availability and outline the concept of underlay awareness required. We provide a distributed algorithm for the construction of an availability latent broker network of degree 2. We aim to develop algorithms for different degrees of availability and test them on practical network samples. Our study will include algorithms for variable availability networks as well.

7. References