2014 IEEE 17th International Conference on Computational Science and Engineering

Fast and Reliable IP Recovery for Overlay Routing in Mission Critical Message Oriented Middleware

Yue Jia
School of Electronic Engineer and Computer Science
Queen Mary, University of London
London, United Kingdom
E-mail: y.jia@ qmul.ac.uk

Chris Phillips
School of Electronic Engineer and Computer Science
Queen Mary, University of London
London, United Kingdom
E-mail: y.jia@ qmul.ac.uk

***Abstract*—For a mission critical message oriented middleware application, such as a financial data delivery bank system or an online stock market, both packet loss rate and latency should be guaranteed within a tolerance range. Fast and reliable recovery from link or node failures is essential. This paper presents a novel algorithm for pre-calculating routing tables and condensing them to save space. This algorithm guarantees 100% fast recovery from single link failures or single node failures. In addition, shortest path first (SPF) routing is maintained even after a failure has happened. The algorithm is compared with two other popular fast recovery schemes (i.e. IP Fast Recovery and the Resilient Routing Layers algorithm) and provides favourable results.**

Keywords- Message Oriented Middleware; Resilience; Overlay Routing

I. INTRODUCTION

Message-oriented middleware (MOM) provides messaging services between the transport layer and application layer [1]. Processes and/or applications do not need to know the existence of each other, so it allows communication in an asynchronous, decoupled manner. Because of its nature, a MOM system is widely recognized as a promising solution for communications between heterogeneous systems. Figure 1 shows an example of a MOM system based on a publish/subscribe structure. Through an overlay network, brokers are inter-connected. Application components, like sensors or processing elements, attach to a local broker [2] and they do not interact directly. In a topic-based publish/subscribe mission critical MOM, each message is classified as belonging to one of a fixed set of topics. There can be an arbitrary number of topics in the system. A publisher labels each message it produces with a particular topic. Similarly, the subscriber has the ability to express their interest in a topic or a pattern of topics to a broker in a suitable data structure. When a component publishes a message, the broker matches this against existing subscriptions, and delivers the message to all those application components that issue matching subscriptions that are local or at neighbouring brokers, and optionally performs message mediation. Each component can publish and subscribe to one or many topics. During the communication phase, brokers located at different geographical areas may suffer from unexpected failure of a broker or overlay link.

A mission critical message oriented middleware application requires a reliable and efficient message delivery mechanism [3, 4]. Messages are deemed to be time sensitive so it is necessary to employ an overlay network fast recovery scheme. Given the growing size and complexity of mission critical MOM overlay networks, the presence of component failures can be an everyday occurrence. Hence, considerable attention has been paid to the problem of fast recovery from link failures [10, 21].

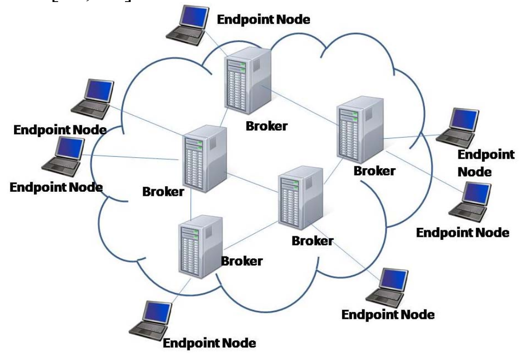


Figure 1. Example MOM Overlay Network

In order to recover quickly from a network failure, some researchers have proposed to reduce the failure detection time [5] whilst others have focused on saving time associated with calculating new routing tables and forwarding and processing state update packets[20].

For one of the most wildly used network routing protocols, Open Shorted Path First (OSPF), if a router does not receive a Hello message from its neighbour within the RouterDeadInterval (typically 40 seconds or 4 HelloIntervals), it assumes the link between itself and the neighbouring node to be down. Therefore, Goyal [5] propose optimizing the HelloInterval such that fast failure detection in the network is possible whilst keeping the false alarm occurrences within acceptable limits. However, the HelloInterval is expressed in 'seconds', so the minimum detection time will be 4 seconds. For a mission critical MOM application, a delay of 4 seconds will still be intolerable.

Quite a few proposals attempt to handle link failures as locally as possible, by only undertaking routing updates in a limited number of routers near the point of failure, such as [6]. This proposal announces that if two equal cost paths exist towards a destination, traffic can safely be switched from one to the other in case of a failure. Similarly, Iselt [7] propose using Multiprotocol Label Switching (MPLS) tunnels to ensure that there are always two equal cost paths towards a destination at every node. Other proposals consider the situation from a global perspective, and try to guarantee that there always is a valid routing entry for a given destination even after a failure. With O2-routing [8], the routing tables are set up so that there are always two valid next hops towards a destination. However, they do not guarantee the shortest path. Hansen [9] propose that for a given network topology, there could be several layers with different safe nodes and safe links. Most nodes and links will only appear in one layer. If a failure arises in a particular layer, the cost of all the links within that layer are set to infinity. Each node just needs to store a routing table for normal operations and for each layer-failure situation.

In order to recover rapidly from failure, this paper proposed a Pre-Calculated Routing Tables (PCRT) algorithm. All the routing tables for normal and different failure scenarios are calculated in advance to save the time by avoiding the need to update a Link State Database (LSD) in response to topology changes triggering reactive flooding of Link State Advertisements (LSAs), and the subsequent calculation of new routing tables. Other than sending LSD flooding messages and exchanging Hello messages, another advantage for this PCRT algorithm is that it employs the exchange of regular Heartbeat messages with a "colour" flag, the colour indicating the state of the network (i.e. in terms of which links are operational) as perceived by the sender. In our scheme, all the pre-calculated routing tables have a one-to-one mapping onto a colour with global significance; that is, all brokers know that a particular colour is uniquely associated with a particular combination of operational and failed links for the MOM topology. If a broker receives a Heartbeat message with different colour and larger sequence number from its own, or if a local failure is detected, an update mechanism is triggered and updated Heartbeat messages will be generated.

II. BACKGROUND AND RELATED WORK

A. *IP Fast Recovery Scheme*

In OSPF, two adjacent routers in the same area periodically exchange Hello messages to maintain link adjacency. If a router does not receive a Hello message from its neighbour within the RouterDeadInterval (typically 40 seconds or 4 HelloIntervals), it assumes the link between itself and the neighbour to be down and generates a new Router LSA to reflect the changed topology. All such, LSAs generated by the routers affected by the failure, are then flooded throughout the network and cause the routers across the network to update their LSD and recalculate the Shortest Path Tree (SPT) from which to update the next hop information in the forwarding table. Thus, the time required to recover from the failure consists of: (1) the failure detection time (2) LSA flooding time (3) and the time to complete the new SPF calculations and update the forwarding tables. Goyal *et al*, focus on reducing the failure detection time, which is clearly the main component of the overall failure recovery time in OSPF-based networks. While the availability of link layer notifications can help achieve fast failure detection, such mechanisms are often not available. Hence, the routers use the Hello mechanism to detect the loss of adjacency with a neighbour. However, there is a limit up to which the HelloInterval can be safely reduced. As the HelloInterval becomes smaller, there is an increasing chance that network congestion will lead to loss of several consecutive Hello messages and thereby cause loss of adjacency between routers even though the routers and the link between them are functioning perfectly well.

Goyal examined the network-wide impact of reducing the HelloInterval in terms of number of false alarms under a realistic model of network congestion. They quantify the detrimental effect of these false alarms in terms of unnecessary SPF calculations undertaken by the routers. They also examined how the network topology influences the occurrence of false alarms. Finally, they considered how much does the faster detection of network failures help in achieving faster recovery from these failures in the operation of OSPF networks.

When traffic in the network is heavy, smaller HelloInterval may lead to false declarations of link failure. Conversely, if the traffic is quite light and link utilisation low, the HelloInterval could potentially be in milliseconds, whilst the OSPF scheme only allows a second to be the smallest time measurement. Therefore, we propose an idea of dynamically adjusting the RouterDeadInterval timer (in our scheme), which can increase the robustness of a mission critical MOM system without undue false alarms.

B. *Resilient Routing Layers Algorithm*

Other researchers have proposed an alternative solution for fast recovery from link failures, called Resilient Routing Layers (RRL) [10]. RRL is based on the idea of building spanning sub-topologies over the full network topology, and using these sub-topologies to forward traffic in case of failures. However, routing recovered traffic according to a sub-topology may cause a high concentration of traffic onto certain links, and hence lose some of the recovery gain due to congestion [11].

The sub-topologies in RRL are called routing layers. In each layer there are some nodes or areas that do not carry transit traffic, and these nodes or areas are the safe nodes of the layer. Layers are constructed so that all nodes have a presence in each layer, and there exists a path between all node pairs in each layer. Each node should be safe in at least one layer to guarantee single node fault tolerance. There are numerous ways to construct the layers so that different protection properties are optimized [12, 13, 14].

The RRL scheme ensures all node-pairs can communicate with each other in all layers, and also that safe nodes will not carry any transit traffic, only traffic originating and terminating in the safe nodes, whereby:

1) Links that haven't been chosen as safe links, which are directly connected to a safe node, can carry all types of traffic originating and terminating anywhere.
2) Links that are chosen to be safe can only be used as a first hop or a last hop for a communication.
3) Traffic originating at a safe node can use a safe link as the first hop, and traffic terminated at a safe node can use a safe link as last hop towards the safe node.

Although, the RRL scheme saves memory by storing fewer routing tables, it does not guarantee the shortest path between source-destination pairs, which could be critical for a time-sensitive overlay network. Normally one hop in an overlay network could correspond to a relatively long distance in the underlying network. The shorter an overlay path is the better. Otherwise, more hops may lead to a much longer geographical distance being covered. With our scheme all the time consuming calculations are performed before the overlay network is running, and for each potential failure scenario, the shortest-path alternative is identified. Furthermore, by using a condensed routing table structure for each broker, there is sufficient memory to store all the necessary routing table information.

III. PRE-CALCULATED ROUTING TABLES SCHEME

In a typical OSPF network, with a HelloInterval of 10 seconds and RouterDeadInterval of 40 seconds, the failure detection can take anywhere between 30 to 40 seconds. The LSA flooding times consist of transmission and propagation delays and any delay resulting from the rate-limiting of LSUpdate packets transmitted through an interface. Once a router receives a new LSA, it schedules an SPF calculation. Since SPF calculations using Dijkstra's algorithm [15] constitute a significant processing load, the router waits for some time (spfDelay - typically 5 seconds) before other arriving LSAs trigger an SPF calculation. Moreover, the routers place a limit on the frequency of SPF calculations (governed by spfHoldTime, typically 10 seconds between successive SPF calculations), which can introduce further delays.

The rationale behind the PCRT scheme is to perform all the time consuming calculations before placing the MOM network in its operational state. There is no Hello message or LSA flooding in the PCRT scheme. Instead a new type of message, called a Heartbeat is introduced. Since the topology of a MOM network does not change frequently and the total number of brokers is normally less than fifty [4], we do not employ Hello message to built adjacency between neighbouring brokers. With PCRT, besides the forwarding of data messages, Heartbeat messages are periodically sent between adjacent brokers to detect link failure(s) and keep network state information synchronized.

At time zero, all the routing tables for each broker for all possible / supported failure scenarios and the default normal conditions are automatically generated and then condensed into a single routing table without redundant entries and distributed to the brokers. In addition, colours and their corresponding failures are associated by means of a shared list. Thus each possible entries in the condensed super routing table at a broker can be simply located with knowledge of the current network state colour, corresponding to a given overlay topological state, and the packet's ultimate destination.

A. *Super Broker*

PCRT is a pre-calculated algorithm. One of the brokers, referred to as the "super broker", is used during the configuration phase to pre-calculate the routing tables for both normal and failures situations for each broker, which are then condensed and then sent to that appropriate broker. The super broker knows the complete topology of the network. An example topology is illustrated in Figure 2.

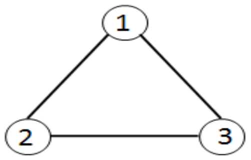


Figure 2. Example Three-Broker Topology

In this paper, we consider single link failures to explain our algorithm and validation was performed using a bespoke Pascal-based simulation tool. The structure of Super Link State Database (SLSD) is shown in Figure 3.

For each topology, the PCRT algorithm generates an SLSD listing of all the brokers, shown on the left. On the right side, the directly connected neighbouring brokers to these brokers are shown.

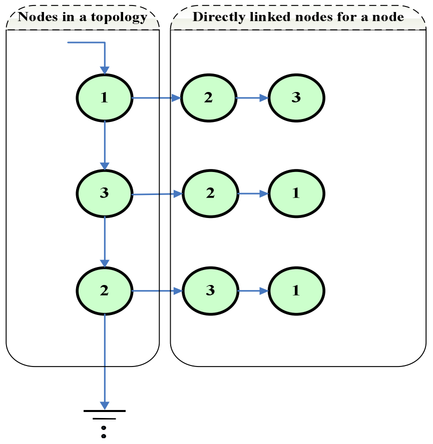


Figure 3. Super Link State Database

B. *Routing Table Generation under Normal and Failure Conditions*

a) *Current Sequence Number List*

A "broker current sequence-number list" exists in every broker. It is a list of all the nodes/brokers in a network and most recently observed sequence-number corresponding to each node/broker. Initially, each broker's sequence-number is set to '1'. When a change (i.e. a failure or recovery) happens, nodes/brokers adjacent to the affected links will increase their sequence number by one and immediately send flood notification messages with the new colour (based on the revised perceived topology state) and sequence-number to their neighbouring nodes/brokers. When a sequence number reaches its largest permitted value, after next change happens, it is reset to 1.

b) *Current Colour*

Each broker stores the current colour of the network, indicating the network state, as it perceives it. If a broker receives a flood notification message (introduced in Section III C) with a different colour from its own current colour, it will compare the sequence number listed in this flood notification message with the one in its current sequence number list. If the sequence number stored in the flood notification message is greater/newer than the number stored in the sequence number list, this broker will update the sequence number list with this new number and search the colour lookup table to determine the failure/failures which relate to this new colour. Together with its own knowledge of failure/failures, this broker will take note of all the failure/failures existing in this network and then lookup the corresponded colour to become the new current colour (by checking the colour lookup table). This new colour may be different from the received one as multiple failures could arise concurrently, unbeknown to the broker sending a particular message. Then the receiving broker forwards flood notification messages to its neighbouring brokers except the one where this flood notification message came from but a flood ACK message is sent in this case. Otherwise, if the sequence stored in this flood notification message is equal to or less/earlier than the sequence number listed in the broker's sequence number list, this broker will discard this flood notification message and send a flood ACK message to the sender of the discarded flood notification message's source.

Conversely, if a broker receives a flood notification message with the same colour as its own current colour, it compares the sequence number listed in this Notification message with its own record of the sender's sequence number. If the sequence number stored in this Notification message is greater than the number stored in the sequence numbers list, this node will update the sequence-number list with this new number and forward this flood notification message along egress interfaces except the one on which this Notification message arrived. Otherwise, this node/broker discards the flood notification message and sends back a flood ACK message (introduced in Section III C).

c) Generating a Super Routing Table

After the Super Broker generates all the routing tables, a routing table set will be generated for each broker where each routing table uniquely corresponds to one colour (associated with a particular link state configuration). These routing table sets are only a temporary data-structure held at the Super Broker; they are not distributed to the remaining brokers but are simply used as an interim step in the construction the Condensed Super Routing Table for each Broker.

Table I and Table II are examples of routing tables temporarily created in sequence at the Super Broker to permit the construction of the condensed super routing table for broker 1 which are based on the topology shown in Figure 2.

TABLE I. A PART OF ROUTING TABLES STORED IN BROKER 1

Colour 1		Colour 2	
Ult. Destination broker	Next hop broker	Ult. Destination broker	Next hop broker
2	2	2	3
3	3	3	3

TABLE II. THE REST PART OF ROUTING TABLES STORED IN BROKER 1

Colour 3		Colour 4	
Ult. Destination broker	Next hop broker	Ult. Destination broker	Next hop broker
2	2	2	2
3	2	3	3

Table III shows the "colour and failures mapping table" where one-to-one mappings from colour to failure are listed. We consider single link failures as an example to keep the explanation simple. A failed link between broker i and broker j, where $i, j \in \{1,2,3\}$, can be represented as $F(i, j)$.

TABLE III. COLOURS AND FAILURES MAPPING TABLE FOR THREE-BROKER TOPOLOGY

Colour	Failure
1	None
2	$F(1,2)$
3	$F(1,3)$
4	$F(2,3)$

This so-called 'colour and failure mapping table' is a list of mapping from colours to failures. One colour is uniquely mapped onto a particular failure or a combination of failures. The reverse mapping is also true. This table will be copied and stored in all the nodes/brokers so they will have the same 'dictionary' of colours and failures. There are two parts to this table. The colours are represented by several bits, which depend on the total number of links in the network and the number of failure combinations that are to be protected against. Then the remainder of this table stores the failures and the combination of failures corresponding to each colour.

Based on Tables I, II and III, all the different combinations of destination broker and next hop broker are used to form the following super routing table. In this super routing table, shown in Table VI, all possible entries for message processing at broker 1 for all considered failure situations are represented.

TABLE IV. THE SUPER ROUTING TABLE FOR BROKER 1

Super routing table			
Address	Ult. Destination broker	Next hop broker	Colour
1	2	2	1,3,4
2	3	3	1,2,4
3	2	3	2
4	3	2	3

Rather than storing four routing tables in broker 1, we condense these tables into one super routing table instead. Let N be the total number of brokers in a topology and L_i is the number of directly linked broker/brokers of broker i, while $1 \le i \le N$. So there are $(N-1)$ destinations for messages processing in broker i. We will know the worst case of a super routing table contains maximum Max different entries:

$$Max = (N-1) \times L_i \tag{1}$$

C. Heartbeat Messages

The Heartbeat message is a new entity proposed in our PCRT algorithm, which is periodically sent to each broker's neighbouring broker/brokers to detect failures and update the latest topology structure. Figure 4 illustrates the structure of the Heartbeat message. There are three parts to the message. The first part is called 'type flag' containing three possible types of a Heartbeat message, namely: *flood notification* message, *test link notification* message and *regular heartbeat* message. The 'Acknowledgement (ACK) flag' is a Boolean variable to show if this Heartbeat message is a response to a certain type of a previous received message. The last part, 'packet information', may contain network change information when a change (path failure or path recovery) happens or maybe null if it is just a regular Heartbeat message.

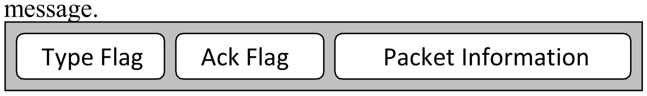


Figure 4. Heartbeat Packet Structure

a) Regular Heartbeat Message

Heartbeat messages are periodically exchanged between adjacent brokers. These normal Heartbeat messages have 'type' showing regular Heartbeat and 'ACK flag' set to FALSE and the 'packet information' part null, are used to detect change of link status. Regular Heartbeat messages are sent periodically and if a broker has not received any Heartbeat message for a specified dead-interval, the link will be assumed to be 'dead' and the state of that link interface will be set to 'failure'. Then this broker immediately generates a corresponding flood notification message and sends it to neighbouring broker/brokers along all operational interfaces. Conversely, if a node receives any kind of Heartbeat message from a 'failed' link, our scheme will test to see if this link has stably recovered by sending link test notification message whilst keeping the link state as 'failure'. There are three states for each interface and they are 'normal', 'waiting' and 'failure'. If an interface receives any type of Heartbeat message within the dead time interval, the dead time interval will be renewed to its largest number, similar to OSPF. In this situation, if this interface is in the state 'normal', it will continue to be 'normal'; if the state of this interface is 'failure', it will send a link test message. If a matched link test ACK received then it will be set to 'normal'; if the state is 'waiting'; only when a matched ACK is received, the state will change to 'normal'. If an interface has not received any type of Heartbeat message before the dead time interval reaches '0', the link for this interface will be set to 'failure' whilst continuing to wait for a Heartbeat message. When If an interface needs to send out a flood notification message (see Section IIIC-b), the state of this interface will be set to 'waiting'. In this state, this interface periodically sends flood notification messages until a suitable ACK, with the right copy of the sent flood notification message, is received.

b) Flood Notification Message

A *flood notification* message will be generated only when a node detects a change of one of its adjacent links and will be sent to all the directly connected neighbouring brokers. At the same time, each link's state will be set to 'waiting' if it is previously in the 'normal' state. This flood notification message's copies will be stored into each interface's ACK waiting list and a corresponding ACK timer for each copy in different interfaces will be initialized. If a matching ACK message is been received within the ACK time, the corresponding flood notification message will be removed from the ACK waiting list, otherwise another copy of the flood notification message will be sent out through that interface to make sure this change can be learnt properly. Only when the ACK waiting list is empty, the state of that interface will be returned to 'normal' again.

It should be noted that an ACK list is needed for each interface as multiple changes may arise causing several flood notification messages to be emitted prior to any acknowledgements. Each flooded message must therefore be matched to a specific ACK message

Other brokers who receive a valid flood notification message forward this message to all their neighbouring brokers apart from the one sending this message. Nothing in the 'packet information' section will be changed when forwarding the message. A corresponding *flood notification* ACK will be sent back through the interface on which it was received no matter if this flood notification message is up-to-date or not. For a flood ACK message, the packet information section will remain the same as the original flood notification message, other than the ACK flag being set. A timer (ack_timer) for this flood message starts and is added to an 'ack_list record' prepared for each interface. Each flood message has its own ack_timer. If the interface does not receive a matched flood ack message while the ack_timer goes to '0', this interface will resend the corresponded flood message again and reset the ack_timer. If the interface receives a matched flood ack message before the ack_timer goes to '0', this record stored in 'ack_list record' will be deleted.

When a broker confirms a change (i.e. a failure or a recovery) has happened, it will go through the following steps:

1) Check the failure lookup table, find the colour, which corresponds to this failure or recovery, and update its CurrentC to the appropriate colour and increase its own SequenceNo by one.
2) Generate flood notification messages with CurrentC, NodeID and SequenceNo written into the 'packet information' section. The 'type' field shows 'flood' and 'ACK flag' is set FALSE. These new flood notification messages are sent to all the directly connected neighbouring brokers even through the 'failure' link (in case it recovers).

c) Link Test Notification Message

A *test link notification* message is generated when a message is received along a link that has been considered to be a 'failure'. For a robust system, it is necessary to have a mechanism to double check whether a link is fully recovered from a failure. This research proposes a method of sending a test link notification message through the 'failed' link and waiting for a response from the broker on the other end to make sure the link is functioning well by return of a specific acknowledgement.

d) ACK Messages

There are three different types of Heartbeat message, so there could be three corresponding ACK messages. However, for the regular Heartbeat message, a response is not expected. In future work, the research could use the combination of regular Heartbeat notification massage with ACK flag 'TRUE' to be a message which can record Round Trip Time [16] as in Transmission Control Protocol (TCP) to help dynamically configure the expiry timer in our algorithm. Table V shows the meaning of each message / acknowledgement combination.

TABLE V. LIST OF ALL TYPES FOR THE 'ACK MESSAGES':

Type list	ACK flag	ACK type
Flood	TRUE	Flood notification ACK message
LinkTest	TRUE	Link test ACK message
Heart-beat	TRUE	Round Trip Time test message

The packet information field in an ACK message is the same as the one in a notification message. The current colour, broker ID and the latest sequence number of that broker are listed in the packet information section. When a broker receives a Heartbeat message with 'flood' and 'TRUE' in the TYPE and ACK tag fields, respectively, it means that this is an ACK message responding to a flood notification message from a neighbouring broker. This broker will compare the information this message holds with its own knowledge in ACK list. If the ACK message matches the record in this interface's ACK list, the record will be removed. If a link test notification message is received through an interface, this broker will generate a link test ACK message sending it back through that interface. At the same time, if this interface is associated with a 'failure' and this is the first Heart-beat message that this interface received after the failure has happened, a link test notification message will be sent through that interface.

IV. SIMULATION RESULTS

Some researchers who focus on optimizing networks, express concerns that pre-calculating all possible routing tables and storing them in routers could lead to a router "out of memory" problem or slower performance [10]. This paper proposes to condense routing tables and remove redundant entities, so each broker will only need to store a single super condensed routing table.

A broker is an application platform, which needs to be embedded into a third part container. The memory that the broker is allowed to use is not determined by the amount of memory allocated to the platform (i.e. it could be a Java Virtual Machine). Although the broker is constrained by the amount of memory given to the platform, the broker manages its memory independently. If any issues with an OutOfMemory were to happen, the administrator could increase the Broker storage memory and additional data storage files until reach the limit of the container. Normally the container (being a server/PC) can have quite a large memory capacity.

Potentially, each broker would require its own routing tables set for both normal and failures situations. If there were n links, including all single link failure situations and the normal situation, there would be *(n+1)* routing tables. However, we propose constructing a condensed super routing table with all possible unique entries from those *(n+1)* routing tables. Redundant entries are omitted, saving space; so our main concern becomes the fast selection of the suitable next hop entry for arriving messages.

We consider two different selection mechanisms. One is a well-known hardware device called a content-addressable memory (CAM) and another is building a two-dimension array.

A CAM provides a lookup table function within a single clock cycle by using dedicated comparison circuitry. It compares input search data against a table with pre-stored data, and then returns the address of the matched data [17, 18]. Nowadays, the most popular commercial application of CAMs is for classifying and forwarding Internet Protocol (IP) packets in network routers [19].

However, instead of employing expensive hardware for our brokers, we propose to use a software program, a two-dimension array, to locate the address in the pre-stored lookup table of the matched input data. Without using CAM to compare input data with data stored in lookup table, a two-dimensional array will assistant locating the address of data, avoiding the need for a search mechanism. In this two-dimensional array, the first dimension indicates the current colour and the second dimension shows the destination broker. Based on the condensed super routing table, we fill the two-dimension array with pointers to the appropriate super routing table next hop entry. The structure of the two-dimension array is as follows:

$$A = \begin{array}{cc} & \text{Colour}\rightarrow \\ \overset{Dest.Broker}{\downarrow} & \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \end{array} \qquad (2)$$

Given the topology in Figure 2 as an example, the lookup two-dimension array will contain the following information.

$$A = \begin{array}{cc} & \text{Colour}\rightarrow \\ \overset{Dest.Broker}{\downarrow} & \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 2 & 2 \\ 3 & 3 & 2 & 3 \end{bmatrix} \end{array} \qquad (3)$$

Figure 5 shows how the combination of the 2-D lookup array and a single condensed routing table provides an efficient data structure arrangement. Multiple colour / destinations can be mapped to a single next hop entry. In addition no searching of the routing table is undertaken. Providing the index information (i.e. the broker's current colour and the packet's ultimate destination) into the array quickly obtains the appropriate pointer that is then used to obtain the next hop information.

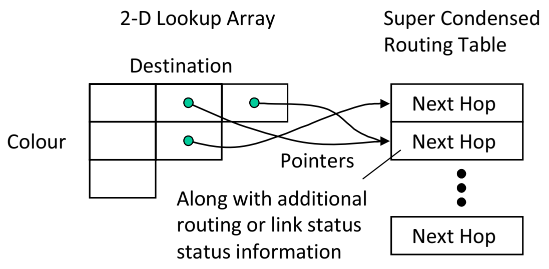

2-D Lookup Array
Super Condensed Routing Table
Destination
Colour
Next Hop
Next Hop
Pointers
Along with additional routing or link status status information
Next Hop

Figure 5. Relationship between 2D Lookup Array and Condensed Routing Table showing Many to One Mapping

The super broker prepares the routing tables for different failure situations for all nodes, followed by the creation of the condensed super routing table and the lookup two-dimensional array. This is relatively time-consuming. However, as this is performed during initialisation, our principle concern is whether it is time consuming to find the right next hop entry via this array. Figure 6 shows the time, in milliseconds on a computer with an Intel i3 CPU, M 350 @ 2.27GHz and 2.00 GB RAM, of accessing different sized two-dimensional arrays a million times. We can see from this figure, for running the lookup once, it will take average 3.9×10^{-12} seconds. This time is so small we can thus ignore it.

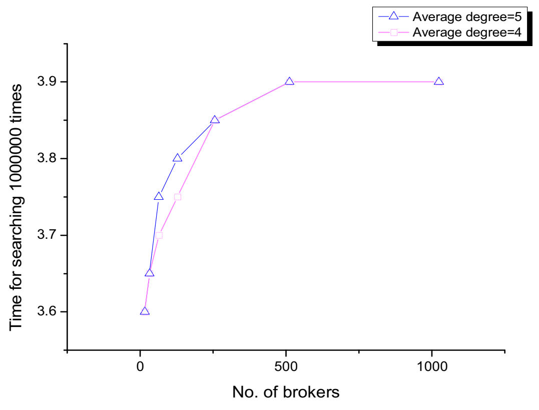


Figure 6. Time for Locating Address in Two-Dimensional Array

A mission critical MOM overlay network, such as a stock market application, could exist over quite a large geographical area or maybe worldwide. Between two brokers, an overlay link may be supported by long underlying network path. Therefore, to guarantee the shortest path first routing in the overlay is highly desirable. One hop more in the overlay network might lead to a message traversing a considerably longer physical distance. Compared with many proposed network recovery algorithms, to our knowledge, our scheme is the only one that guarantees shortest path when building the routing table for various failure situations. Figure 7 shows a comparison of path length for a 32 node - 64 link network for different network recovery abstractions.

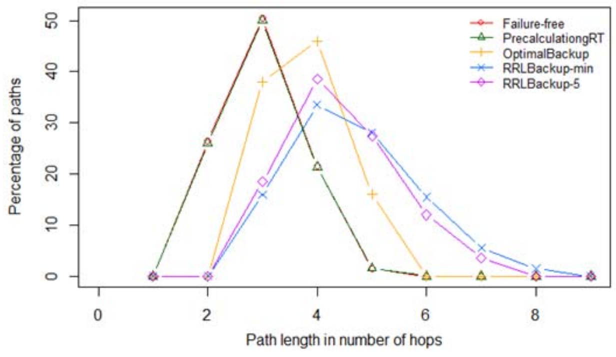


Figure 7. Path Length of a 32 Node-64 Link Topology

Experiments are based on single link failure for a 32 nodes and 64 links network topology. By failing links one per time, we can calculate total number of each different path length and then calculate each path length's percentage among all the paths.

Using a bespoke packet-level simulator, to demonstrate the benefit of our scheme for fast recovery from failures, various topologies were constructed, including the simple 6-node one, shown in Figure 8.

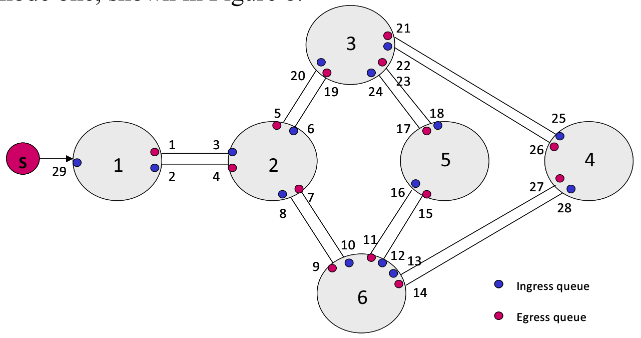
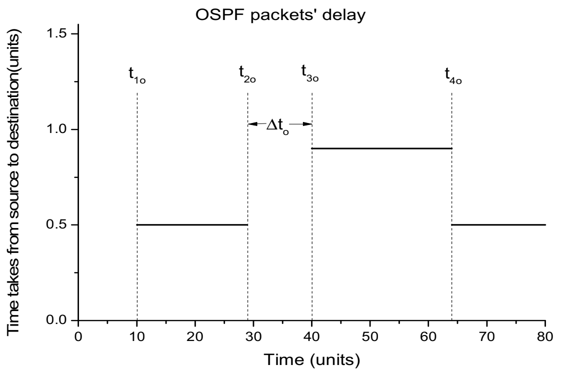


Figure 8. 6-Node Example Topology

This paper compares PCRT and fast OSPF based on this 6-node topology and has the following set up for all queues: the notional service bitrate is 10000bits/unit time with a buffer capacity of 10 packets; the data packet generation rate at the source is 1 per unit time at a fixed interval; the data packet size is 1000 bits. This simulation runs for 80 time units and the link between node 2 and node 3 fails at time 30 and it recovers at time 60.



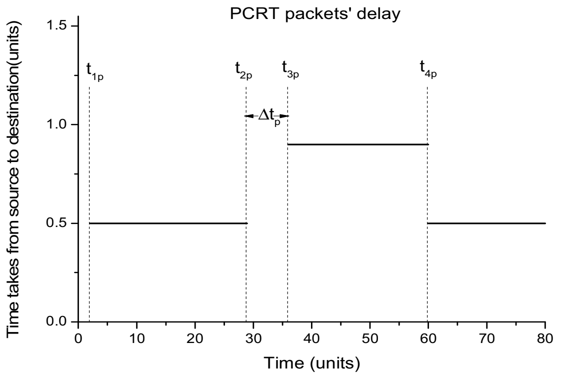


Figure 9. End-to-End Packet Transfer Latency for OSPF/PCRT

t_{1o} and t_{1p} are the time at which this system starts to send packets; t_{2o} and t_{2p} are the time at which the link between node/broker 2 to node/broker 3 fails; t_{3o} and t_{3p} are the time at which this system reconverges and successfully delivers packets again; t_{4o} and t_{4p} are the time in which the system starts to use the recovered link and successfully deliver packets via the original path; $\vartriangle t$ s are the time it takes for this system to reconverge. The PCRT does not need to exchange LSA messages as the whole network has been set up in advance. PCRT gets converge faster than OSPF, although for "fairness" we set the heartbeat interval to be the same as the OSPF hello interval, so the detection times are the same. Even so the processing time of PCRT is less leading to a quicker switch to the recovery shortest path. In addition, PCRT is able to use the healed link quicker than for OSPF. Figure 10 shows the package loss during the time the system reconverges.

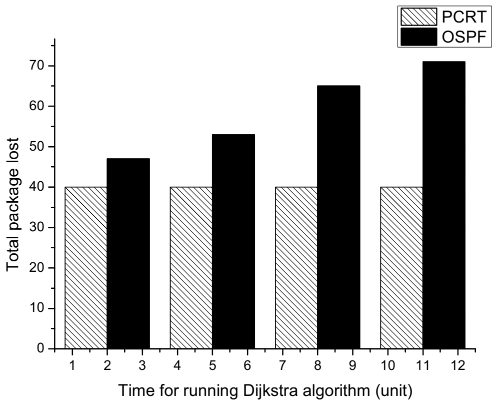


Figure 10. Reconvergence Packet Loss for Fast OSPF / PCRT

As the reconvergence time is still notably longer for "fast" OSPF, more packets are lost until an alternative path is found to the destination than for PCRT. With PCRT, the updated routing table information is ready to-hand as soon as the failure is discovered.

References

[1] J. Wang, P. Jiang, J. Bigham, B. Chew, M. Novkovic, and I. Dattani. "Adding resilience to message oriented middleware". In Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems (SERENE '10). ACM, New York, NY, USA, pp89-94, 2010

[2] X.F. An, L.Y. Bian. "Design of Message-Oriented Middleware of Distance Teaching Platform Based on Distributed Message Control". Computational Aspects of Social Networks (CASoN), International Conference on. IEEE, pp141-143, 2010.

[3] J. Wang, J. Bigham, and J. Wu, "Enhance Resilience and QoS Awareness in Message Oriented Middleware for Mission Critical Applications", Information Technology: New Generations (ITNG), 2011 Eighth International Conference on. IEEE, 2011: 677-682.

[4] Y. Jia, E. Bodanese, J. Bigham, "Checking the Robustness of a Publish/Subscribe Based Message Oriented System", IV International Congress on Ultra Modern Telecommunications and Control Systems, St. Petersburg, pp 291-296, October 2012.

[5] M. Goyal, K.K. Ramakrishnan, W. Feng. "Achieving faster failure detection in OSPF networks", Communications, 2003. ICC'03. IEEE International Conference on. IEEE, pp 296-300, 2003.

[6] Y. Liu, A.N. Reddy. "A fast rerouting scheme for OSPF/IS-IS networks", Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on. IEEE, pp 47-52, 2004.

[7] A. Iselt, A. Kirstadter, A. Pardigon, et al. "Resilient routing using MPLS and ECMP". High Performance Switching and Routing, 2004. HPSR. 2004 Workshop on. Phoenix, Arizona, USA, IEEE, pp345-349, 2004.

[8] G. Schollmeier, J. Charzinski, A. Kirstadter, et al. "Improving the resilience in IP networks", High Performance Switching and Routing, HPSR. Workshop on. IEEE, Torino, Italy, pp91-96, 2003.

[9] A.F. Hansen, A. Kvalbein, T. Čičić, et al. "Resilient routing layers for network disaster planning", Networking-ICN 2005. Springer Berlin Heidelberg, pp1097-1105, 2005.

[10] A. Kvalbein, A.F. Hansen, T. Cicic, S. Gjessing, O. Lysn: "Fast recovery from link failures using resilient routing layers". 10th IEEE Symposium on Computers and Communications (ISCC 2005), La Manga, Spain, pp 554-560 , 2005.

[11] A.F. Hansen, A. Kvalbein, S. Gjessing, et al. "Fast, effective and stable IP recovery using resilient routing layers" The 19th international teletraffic congress (ITC19). 2005.

[12] R. Bartos, M. Raman. "A heuristic approach to service restoration in MPLS networks". In Proc. ICC, pages 117–121, June 2001.

[13] K. Menger. Zur allgemeinen kurventheorie. Fund. Math., 10:95–115, 1927.

[14] F. Otel. "On fast computing bypass tunnel routes in MPLSbased local restoration". In Proceedings of 5th IEEE International Conference on High Speed Networks and Multimedia Communications, Jeju, Korea, pp234–238, 2002.

[15] E. Dijkstra, "A note on two problems in connection with graphs,"Numerische mathematik, pp269-271, 1959.

[16] H. Jiang, C. Dovrolis. "Passive estimation of TCP round-trip times". ACM SIGCOMM Computer Communication Review, pp75-88,July 2002.

[17] K. Pagiamtzis, A. Sheikholeslami. "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey[J]." Solid-State Circuits, IEEE Journal of, 2006, 41(3): 712-727.

[18] S. Stas, "Associative processing with CAMs". Northcon/93 Conf. Record, pp. 161-167 1993

[19] G. Qin, S. Ata, I. Oka, and C. Fujiwara, "Effective bit selection methods for improving performance of packet classifications on IP routers". Proc. IEEE GLOBECOM, vol. 2, pp. 2350-2354, 2002

[20] T. Clausen, P. Jacquet, C. Adjih, et al. "Optimized link state routing protocol (OLSR)[J]". 2003.F.

[21] S. Kini, S. Ramasubramanian, A. Kvalbein, et al. "Fast recovery from dual link failures in IP networks[C]"//INFOCOM 2009, IEEE. IEEE, 2009: 1368-1376