IJDCST @Jan Issue- V-2, I-1, SW-10
ISSN-2320-7884(Online) ISSN-2321-0257(Print)

Efficiently Controlling the Traffic in Wireless Network

P.Ravi[1], M.Satish[2]
[1]M.Tech Student, [2]Associate Professor
[1,2]Dept. of Computer Science and Engineering
[1,2]Akula Sri Ramulu College of Engineering, Tanuku

Abstract— The Internet takes an central role in our communications infrastructure, the slow convergence of steering protocols after a network failure becomes a growing problem. To assure fast recovery from bond and node failures in IP networks, we present a new recovery scheme called Multiple Steering Pattern (MSP). MSP is based on keeping additional steering information in the routers, and allows packet ahead to continue on an alternative output bond immediately after the detection of a failure. Our proposed scheme guarantees recovery in all single failure scenarios, using a single mechanism to handle both bond and node failures, and without knowing the root cause of the failure. MSP is strictly connectionless, and assumes only destination based hop-by-hop ahead. It can be implemented with only minor changes to existing solutions. In this paper we present MSP, and analyze its performance with respect to scalability, backup path lengths, and load distribution after a failure.

Keywords— Internet, Multiple steering patterns (MSP), Topologies, Travel Production, Virtual Path.

I. INTRODUCTION

The Internet has been transformed from a special purpose network to an ubiquitous platform for a wide range of everyday communication services. The demands on Internet reliability and availability have increased accordingly. A disruption of a bond in central parts of a network has the potential to affect hundreds of thousands of phone conversations or TCP connections, with obvious adverse effects. The ability to recover from failures has always been a central design goal in the Internet. IP networks are intrinsically robust, since IGP steering protocols like OSPF are designed to update the ahead information based on the changed topology after a failure. This re-convergence assumes full distribution of the new bond state to all routers in the network domain.

This network-wide IP re-convergence is a time consuming process, and a bond or node failure is typically followed by a period of steering instability. During this period, packets may be dropped due to invalid routes. This phenomenon has been studied in both IGP and BGP context and has an adverse effect on real-time applications. Events leading to a re-convergence have been shown to occur frequently, and are often triggered by external steering protocols

1.1 Travel Production:

TE is the process of pathtravel across to the backbone to facilitate efficient use of available bandwidth between a pair of routers. Prior to MPLS TE, travelproduction was performed either by IP or by ATM, depending on the protocol in use between two edge routers in a network. Though the term "travelproduction" has attained popularity and is used more in the context of MPLS TE today, traditional TE in IP networks was performed either by IP or by ATM.TravelProduction (TE) is an essential aspect of contemporary network management. Offline TE approaches aim to optimize network resources in a static manner, but require accurate estimation of travel matrices in order to produce optimized network pattern for long-term operation (a resource provisioning period each time, typically in the order of weeks or even longer). However, these approaches often exhibit operational inefficiencies due to frequent and significant travel dynamics in operational networks. Take the published travel traces dataset in the GEANT network as an illustration. The actual maximum bond utilization (MLU) dynamics is substantial on a daily basis, varying from less than 40 percent during off-peak time to more than 90 percent in busy hours. As such, using one single travel matrix as input for offline computing a static TE configuration is not deemed as an efficient approach for resource optimization purposes in such dynamic environments.

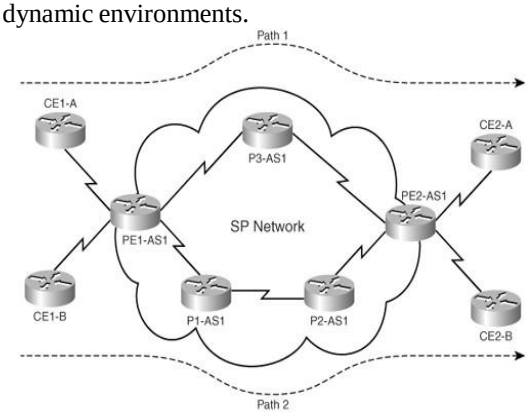


Fig: Traditional IP Networks

As two paths exist between customer routers CE1-A and CE2-A via the provider network. If all bonds between the routers of equal cost, the preferred path between customer routers CE1-A and CE2-A would be the one with the minimum cost (via routers PE1-AS1, P3-AS1, and PE2-AS1) or *PATH1*. The same would apply for the customer routers CE1-B and CE2-B belonging to Customer B. If all the bonds were T3 bonds, for example, in the event of CE1-A sending 45 Mbps of travel and CE1-B simultaneously sending 10 Mbps of travel, some packets will be dropped at PE1-AS1 because the preferred path for both customers is using *PATH1*. The path *PATH2*will not be utilized for traveladvancing; therefore, TE can utilize this available bandwidth.

To implement TE using IP whereby the paths *PATH1* and *PATH2* are either load balanced or used equally, we will need to implement IGP features such as maximum paths with variance or change the cost associated with the suboptimal path, *PATH2*, to make it equal to the current optimal path, *PATH1*. In an SP environment, this is

often cumbersome to implement as the number of routers is much larger. The solution is a lot more feasible; PVCs can be configured between routers PE1-AS1 and PE2-AS1 with the same cost, but this would create a full mesh of PVCs between groups of routers. Implementing ATM for TE, however, has an inherent problem when a bond or a node goes down. During bond or node failure used in conjunction with ATM for TE, messages are flooded on the network. The Layer 3 topology must be predominantly fully meshed to take advantage of the Layer 2 TE implementation. Often, this might prove to be a scalability constraint for the IGP in use, due to issues with reconvergence at Layer 3.

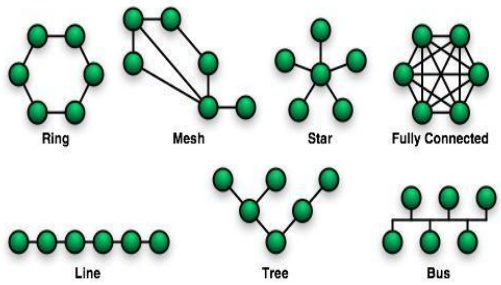
1.2 Topologies:

The physical topology of a network refers to the configuration of cables, computers, and other peripherals.The logical topology of a network refers to the method used to pass information between workstations.

There are two basic categories of network topologies:

1. Physical topologies
2. Logical topologies

The shape of the cabling layout used to bond devices is called the physical topology of the network. This refers to the layout of cabling, the locations of nodes, and the interconnections between the nodes and the cabling.[1] The physical topology of a network is determined by the capabilities of the network access devices and media, the level of control or fault tolerance desired, and the cost associated with cabling or telecommunications circuits.



The logical topology, in contrast, is the way that the signals act on the network media, or the way that the data passes through the network from one device to the next without regard to the physical interconnection of the devices. A network's logical topology is not necessarily the same as its physical topology. For example, the original twisted pair Ethernet using repeater hubs was a logical bus topology with a physical star topology layout. Token Ring is a logical ring topology, but is wired a physical star from the Media Access Unit.

The logical classification of network topologies generally follows the same classifications as those in the physical classifications of network topologies but describes the path that the *data* takes between nodes being used as opposed to the actual *physical* connections between nodes. The logical topologies are generally determined by network protocols as opposed to being determined by the physical layout of cables, wires, and network devices or by the flow of the electrical signals, although in many cases the paths that the electrical signals take between nodes may closely match the logical flow of data, hence the convention of using the terms *logical topology* and *signal topology* interchangeably.

1.3 Multiple Steering Patterns:

The main idea of MSP is to use the network graph and the associated bond weights to produce a small set of backup network pattern. The bond weights in these backup pattern are manipulated so that for each bond and node failure, and regardless of whether it is a bond or node failure, the node that detects the failure can safely forward the incoming packets towards the destination. MSP assumes that the network uses shortest path steering and destination based hop-by-hop ahead.

MSP solves the last hop problem by strategic assignment of bond weights between the backup pattern. MSP has a range of attractive features:

- It gives almost continuous ahead of packets in the case of a failure.
- MSP helps improve network availability through sup press ion of the re-convergence process.
- MSP uses a single mechanism to handle both bond and node failures. Failures are handled locally by the detecting node, and MSP always finds a route to the destination.
- MSP makes no assumptions with respect to the *root cause of failure*,
- An MSP implementation can be made without major modifications to existing IGP steering standards.

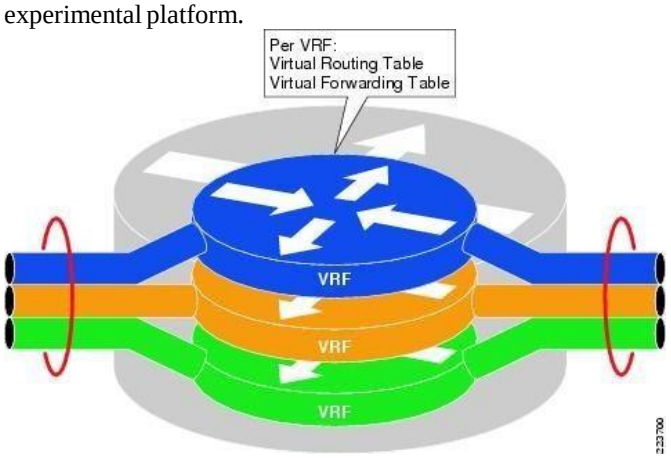
. In MSP, all bonds remain in the topology, but in some pattern, some bonds will not be selected by shortest path steering mechanisms due to high weights.

1.4 Virtual Path:

The Virtual Router scheme can be broken down into two components, the virtual router server, and number virtual router clients. The virtual router server is an application or set of applications running on a physical server that have access to bond layer travel. The virtual router client is a program created by users of a VR scheme. In a typical session, a user will use a VR client to connect to the server via a TCP socket, and a decision is made about what packets the server will send to the client. The user may then run a network application, such as FTP, whose travel can be seen by the server. The server, on capturing travel, will based on the simulated topology decide whether or not a particular client can see the packet. If this is the case it is advanced to the client who can then manipulate the packet (e.g. decrement the TTL field). The client may then send the packet back to the server with instructions to send it out of a particular interface on the network, thus potentially path the packet. With its relationship

with the server, the VR client has full capability to manipulate and route travel flows from user space.

The virtual router has successfully been used to teach students basic functionality of routers. Students had to implement virtual router clients that make path decisions, use the ARP protocol to communicate with other routers and route actual web travel from their desktop computers. However the scalability of the virtual router (up to several hundred simulated routers) makes it attractive as a research tool. Currently there are efforts under way to investigate the behaviour of path protocols using the virtual router as an experimental platform.



The virtual router is an effort to develop a platform to facilitate research and teaching of network path. It allows to set up a virtual topology of routers and network bonds and to run user space programs on the simulated hardware. Unlike existing network simulators such as ns2, the scheme operates with real IP packets in real time. This makes it possible for students or researchers to generate travel with real, standard clients and servers and evaluate performance over the simulated topology In virtually all modern operating schemes, path is implemented within kernel space. Providing a good kernel debugging environment and assuming a basic knowledge of kernel level development may be restrictive requirements for a project focused on path. The virtual router project provides full access to actual network travel, but bypasses the trouble of having to work within the kernel. Some experiments such (e.g. path protocols such as RIP) require more than a single router. The virtual router makes it possible to simulate large number of routers on one physical host and enforce a virtual network topology between them.

II. SYSTEM OUTLINE:

A salient novelty is that the optimization of the MT-IGP bond weights does not rely on the availability of the travel matrix a priori, which plagues existing offline TE solutions due to the typical inaccuracy of travel matrix estimations. Instead, our offline bond weight optimization is only based on the characteristics of the network itself, i.e. the physical topology.

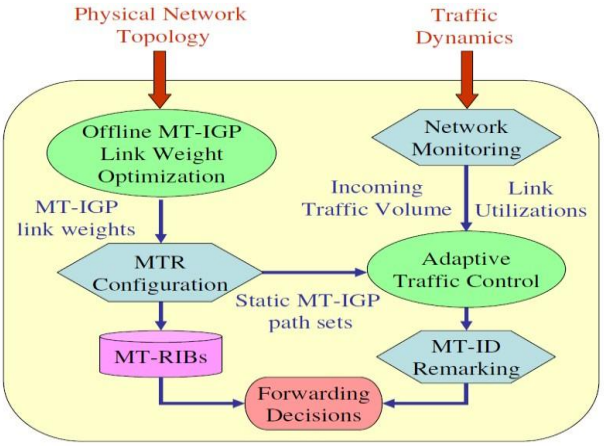


Figure: AMPLE Scheme overview

The computed MT-IGP bond weights are configured in individual routers and the corresponding IGP paths within each VRT are populated in their local path information bases (MT-RIBs). While OLWO focuses on static pathconfiguration in a long timescale (e.g. weekly or monthly), the ATC component provides complementary functionality to enable short timescale (e.g. hourly) control in response to the behaviour of travel that cannot be usually anticipated. As previously mentioned, the ultimate objective of OLWO is to provision offline maximum intra-domain path diversity in the path plane, allowing the ATC component to adjust at short timescale the travel assignment across individual VRTs in the advancing plane.

The input for ATC includes:

1) Diverse MT-IGP paths according to the bond weights computed by OLWO, and
2) Monitored network and travel data such as incoming travel volume and bond utilizations.

At each short-time interval, ATC computes new travel splitting ratio across individual VRTs for re-assigning travel in an optimal way to the diverse IGP paths between each S-D pair. This functionality is handled by a centralized TE manager who has complete knowledge of the network topology and periodically gathers the up-to-date monitored travel conditions of the operating network. The TE manager function can be realized as a dedicated server, but for robustness and resilience it can be implemented in a distributed replicated manner for avoiding the existence of a single point of failure. In the next section we present the detailed design of individual components in the AMPLE scheme.

III. EXISTING SYSTEM

In existing scheme consists of two complementary components: *offline bond weight optimization* that takes as input the physical network topology and tries to produce maximum path diversity across multiple virtual path topologies for long term operation through the optimized

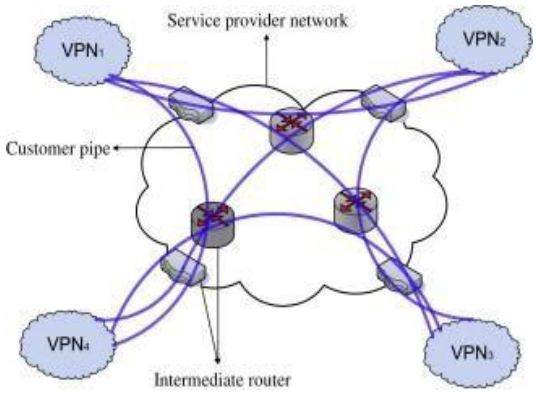
setting of bond weights. Based on these diverse paths, *adaptive travel control* performs intelligent travel splitting across individual path topologies in reaction to the monitored network dynamics at short timescale. According to our evaluation with real network topologies and travel traces, the proposed scheme is able to cope almost optimally with unpredicted travel dynamics and, as such, it constitutes a new proposal for achieving better quality of service and overall network performance in IP networks.

Modules:

1. Virtual travel allocation
2. Offline Bond Weight Optimization
3. Network Monitoring
4. Adaptive Travel Control

1. Virtual Travel Allocation:

In this Module, the diverse MT-IGP paths according to the bond weights computed by OLWO. Monitored network and travel data such as incoming travel volume and bond utilizations. At each short-time interval, ATC computes a new travel splitting ratio across individual VRTs for re-assigning travel in an optimal way to the diverse IGP paths between each S-D pair.

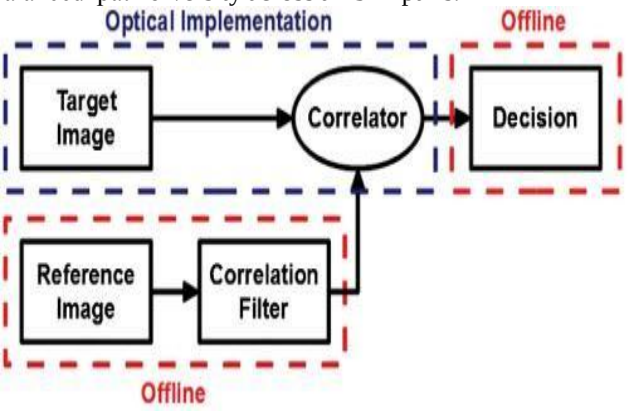


This functionality is handled by a centralized TE manager who has complete knowledge of the network topology and periodically gathers the up-to-date monitored travel conditions of the operating network. These new splitting ratios are then configured by the TE manager to individual source Pop nodes, who use this configuration for remarking the multi-topology identifiers (MTIDs) of their locally originated travel accordingly.

2. Offline Bond Weight Optimization :

In this module, to determine the definition of "path diversity" between Pops for travelproduction. Let's consider the following two scenarios of MT-IGP bond weightconfiguration. In the first case, highly diverse paths (e.g. end-to-end disjoint ones) areavailable for some Pop-level S-D pairs, while for some other pairs individual paths arecompletely overlapping with each other across all VRTs. In the second case, none of theS-D pairs have disjoint paths, but none of them are completely overlapping either.

Obviously, in the first case if any "critical" bond that is shared by all paths becomes congested, its load cannot be alleviated through adjusting travel splitting ratios at theassociated sources, as their travel will inevitably travel through this bond no matter whichVRT is used. Hence, our strategy targets the second scenario by achieving "balanced"path diversity across all S-D pairs.



3. Network Monitoring:

Network monitoring is responsible for collecting up-to-date travel conditions in real-time and plays an important role for supporting the ATC operations. AMPLE adopts a hop-by-hop based monitoring mechanism that is similar to the proposal.

4. Adaptive Travel Control

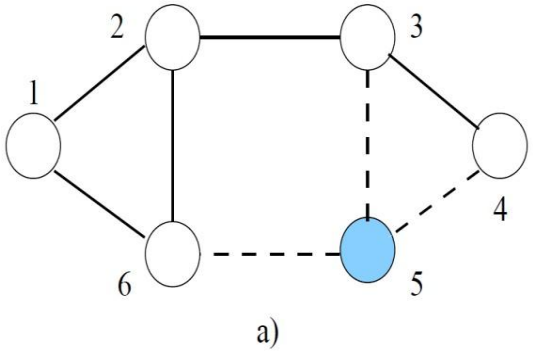
In this Module, Measure the incoming travel volume and the network load for the current interval as compute new travel splitting ratios at individual Pop source nodes based on the splitting ratio configuration in the previous interval, according to the newly measured travel demand and the network load for dynamic load balancing.

IV. PROPOSED SYSTEM

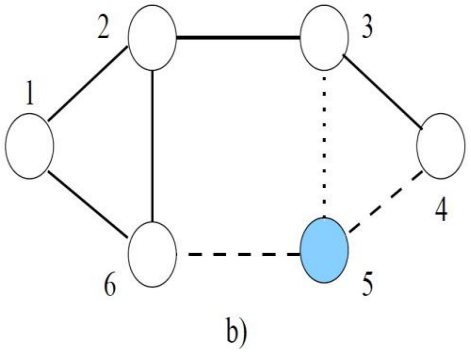
Our MSP approach is threefold. First, we create a set of backup pattern, so that every network component is isolated in one configuration. Second, for each configuration, a standard steering algorithm like OSPF is used to calculate configuration specific shortest path trees and create ahead tables in each router, based on the pattern. The use of a standard steering algorithm guarantees loop free ahead within one configuration. Finally, we design a ahead process that takes advantage of the backup pattern to provide fast recovery from a component failure.

MSP is based on using a small set of backup steering pattern, where each of them is resistant to failures of certain nodes and bonds. Given the original network topology, a *configuration* is defined as a set of associated bond weights. In a configuration that is resistant to the failure of a particular node n, bond weights are assigned so that interchange routed according to this configuration is never routed through node n.

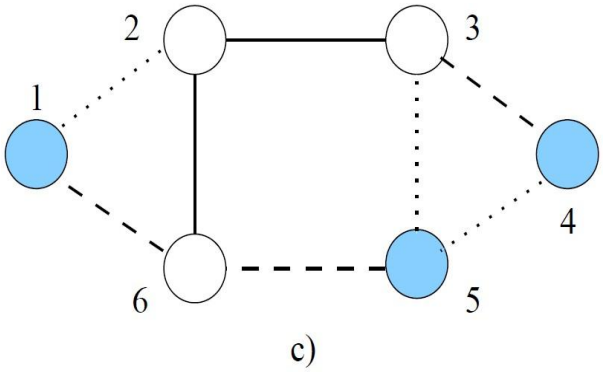
The failure of node n then only affects interchange that is sent from or destined to n. Similarly, in a configuration that is resistant to failure of a bond l, interchange routed in this configuration is never routed over this bond, hence no interchange routed in this configuration is lost if l fails. In MSP, node n and bond l are called *isolated* in a configuration, when, as described above, no interchange routed according to this configuration is routed through n or l.



a) Node 5 is isolated (shaded color) by setting a high weight on all its connected bonds (stapled). Only interchange to and from the isolated node will use these restricted bonds.



b) The bond from node 3 to node 5 is isolated by setting its weight to infinity, so it is never used for interchange ahead (dotted).



c) A configuration where nodes 1, 4 and 5, and the bonds 1-2, 3-5 and 4-5 are isolated.

Using a standard shortest path calculation, each router creates a set of configuration-specific ahead tables. For simplicity, we say that a packet is forwarded according to a configuration, meaning that it is forwarded using the ahead table calculated based on that configuration.

It is important to stress that MSP does not affect the failure free original steering, i.e. when there is no failure, all packets are forwarded according to the original configuration, where all bond weights are normal. Upon detection of a failure, only interchange reaching the failure will switch configuration. All other interchange is forwarded according to the original configuration as normal.

V. GENERATING BACKUP PATTERN

In this section, we will first detail the requirements that must be put on the backup pattern used in MSP. Then, we propose an algorithm that can be used to automatically create such pattern. The algorithm will typically be run once at the initial startup of the network, and each time a node or bond is permanently added or removed.

A. Configuration Constraints

To guarantee single-failure tolerance and consistent steering, the backup pattern used in MSP must adhere to the following requirements:

1) A node must not carry any transit interchange in the configuration where it is isolated. Still, interchange must be able to depart from and reach an isolated node.
2) A bond must not carry any interchange at all in the configuration where it is isolated.
3) In each configuration, all node pairs must be connected by a path that does not pass through an isolated node or an isolated bond.
4) Every node and every bond must be isolated in at least one backup configuration.

The first requirement decides what weights must be put on the restricted bonds attached to an isolated node. To guarantee that no path will go through an isolated node, it suffices that the restricted bonds have a weight W of at least the sum of all bond weights w in the original configuration:

$$W > \sum_{e_{i,j} \in \mathcal{E}} w_{i,j}$$

The second requirement implies that the weight of an solvated bond must be set so that interchange will *never* be routed over it. Such bonds are given infinite weight. Given these restrictions on the bond weights, we now move on to show how we can construct backup pattern that adhere to the last two requirements stated above.

B. Algorithm

We now present an algorithm, designed to make all nodes and bonds in a arbitrary disconnected graph isolated. Our algorithm takes as input the undirected weighted graph G, and the number n of backup pattern that is intended created.

TABLE I
NOTATION

$\mathcal{G}(\mathcal{V},\mathcal{E})$	Graph with nodes $\mathcal{V}$ and undirected links $\mathcal{E}$
$\mathcal{G}_p$	The graph with link weights as in configuration p
$\mathcal{S}_p$	Isolated nodes in configuration p
$\mathcal{E}_i$	All links from node i
$e_{i,j}$	Undirected link from node i to node j ($e_{i,j} = e_{j,i}$)
$w^p_{i,j}$	Weight of link $e_{i,j}$ in configuration p
n	Number of configurations to generate (input)
W	Weight of restricted links

The third property above results in the following two invariants for our algorithm, which must be evaluated each time a new node and its connected bonds are isolated in a configuration:

- A configuration must contain a backbone
- All isolated nodes in a configuration must be directly connected to the backbone through at least one restricted bond.

The first invariant means that when a new node is isolated, we must make sure that the sub-graph of non-isolated nodes is not divided. If making a node isolated breaks any of these two requirements, then the node cannot be isolated in that configuration.

When a node vi is attempted isolated in a backup configuration p, it is first tested that doing so will not break the first invariant above. The div method (for "divide") at line 11 decides this by testing that each of vi's neighbors can reach each other without passing through vi, an isolated node, or an isolated bond in configuration p.

Along with vi, as many as possible of its attached bonds are isolated. We run through all the attached bonds (line 13). The node vj in the other end of the bond may or may not be isolated in some configuration already (line 15). If it is, we must decide whether the bond should be isolated along with vi (line 20), or if it is already isolated in the configuration where vj is isolated (line 27). A bond must always be isolated in the same configuration as one of its end nodes. Hence, if the bond Was not isolated in the same configuration as vj , it *must* be isolated along with node vi. Before we can isolate the bond along with vi, we must test (line 18) that vi will still have an attached non-isolated bond, according to the second invariant above. If this is not the case, vi can not be isolated in the present configuration (line 23). Giving up the node in the present configuration means restarting the outer loop (line 9).

It is important to note that this also involves resetting all changes that has been made in configuration p trying to isolate vi. In the case that the neighbor node vj was *not* isolated in any configuration (line 29), we isolate the bond along with vi if possible (line 34). If the bond can not be isolated (due to the second invariant), we leave it for node vj to isolate it later. To make sure that this bond can be isolated along with vj ,

If vi was successfully isolated, we move on to the next node. Otherwise, we keep trying to isolate vi in every configuration, until all pattern are tried (line 9). If vi could not be isolated in any configuration, requirement four in Sec. III-A could not be fulfilled. The algorithm will then terminate with an unsuccessful result (line 48). This means that our algorithm could not isolate all network elements using the required number of pattern, and a higher number of pattern must be tried.

Algorithm 1: Creating Backup Configurations

```
1  for p ∈ {0 ... n − 1} do
2      G_p ⇐ G
3      S_p ⇐ ∅
4  end
5  p ⇐ 0
7  forall v_i ∈ V do
9      while v_i ∉ S_p and not all configurations
       tried do
11         if !div(v_i, G_p) then
13             forall e_{i,j} ∈ E_i do
15                 if ∃q : v_j ∈ S_q then
16                     if w^q_{i,j} = W then
18                         if ∃e_{i,k} ∈ E_i \ e_{i,j} : w^p_{i,k} ≠ ∞ then
20                             w^p_{i,j} ⇐ ∞
21                         else
23                             break 9
25                     else if w^q_{i,j} = ∞ and p ≠ q then
27                         w^p_{i,j} ⇐ W
29
30                 else
32                     if ∃e_{i,k} ∈ E_i \ e_{i,j} : w^p_{i,k} ≠ ∞ then
34                         w^p_{i,j} ⇐ ∞
35                     else
37                         w^p_{i,j} ⇐ W
39                         firstInNodeQ(v_j)
41                         firstInLinkQ(v_j, e_{j,i})
43             commit edge weight changes
44             S_p ⇐ S_p ∪ v_i
45         p++ mod n
46     if v_i ∉ S_p then
48         Give up and abort
49 end
```

5.1 Local Ahead Process:

The algorithm presented creates a set of backup pattern. Based on these, a standard shortest path algorithm is used in each configuration, to calculate configuration specific ahead tables. In this section, we describe how these ahead tables are used to avoid a failed component. When a packet reaches a point of failure, the node adjacent to the failure, called the *detecting node*, is responsible for finding the configuration where the failed component is isolated, and to forward the packet according to this configuration.

A node must know in which configuration the downstream node of each of its network interfaces is isolated. Also, it must know in which configuration it is isolated itself. This information is distributed to the nodes in advance, during the configuration generation process.

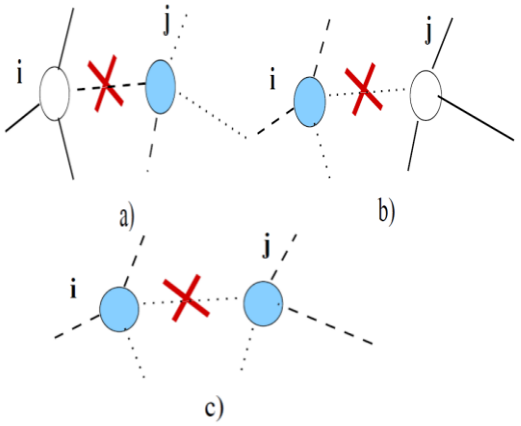


Fig : Isolated nodes are given a shaded color. When there is an error in the last hop, a packet must be forwarded in the configuration where the connecting bond is isolated

The packet, and the neighbor is indeed dead, will the packet reach a dead interface for the second time (in a single failure scenario). It will then be discarded in another node.

If, however, the dead bond *is* returned from the lookup in the configuration where the neighbor is isolated, we know that the neighbor node must be the egress node for the packet, since packets are never routed through an isolated node. In this case, a lookup in the configuration where the detecting node itself is isolated must be made. Remember that a bond is always isolated in the same configuration as one of its attached nodes. Hence, the dead bond can never be returned from this lookup. Again, if the neighbor (egress) node is indeed dead, the packet will be discarded in another node upon reaching dead interface for the second time.

A. Last Hop Failure Example

For an example of how packet ahead is done in the case of a failure in the last hop, consider the situation depicted. In the last hop, packets will be forwarded in the configuration where either node i or node j is isolated, depending on where the bond between them is isolated. In Fig. 3a, the bond is not isolated in the same configuration as node j. A route lookup in this configuration will return the same broken bond. Hence, a lookup must be made in the configuration where node i is isolated,

Note that if nodes i and j are isolated in the same configuration, the bond connecting them is also isolated in that configuration, as shown in Fig. 3c. Packets will then always reach the egress in that configuration, even if it is the last hop bond that fails, unless, of course, the egress node itself has failed.

B. Implementation issues

While the backup pattern can be generated off line, and information can be represented in the network using Multi Topology steering mechanisms, the described ahead process needs additional software functionality in the routers. The described ahead process consists however of simple tests and next-hop lookups only, and should be easy to implement. The routers will need a mapping between each interface and a specific backup configuration. This mapping can be built when the pattern are created.

VI. EXPERIMENTAL RESULTS

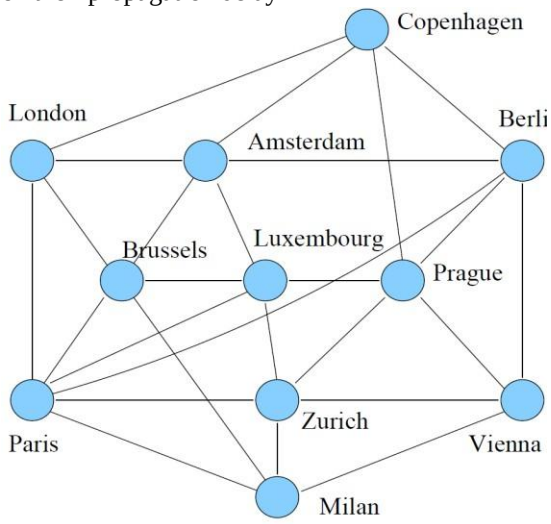
MSP is a local, proactive recovery scheme that resumes packet ahead immediately after the failure is detected, and hence provides fast recovery. State requirements and the influence on network interchange are other important metrics, which will be evaluated in this section

1) Steering simulation:

We have developed a Java software model that is used to create pattern as described by the algorithm in Sec. III-B. The pattern are created for a wide range of topologies, obtained from the BRITE

2) Interchange simulation:

To test the effects our scheme has on the load distribution after a failure, we have implemented our scheme in a discrete-event packet simulator based on the J-Sim All bonds have been given a common base weight (dominant), plus an individual addition based on their propagation delay



For all simulations, three backup pattern were used with MSP. We evaluate bond loads before the failure, and after recovery using OSPF or MSP.

B. Steering Results

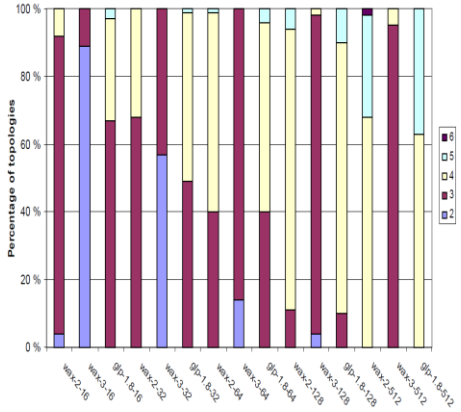
1) Minimum number of backup pattern:

*T*he minimum number of backup pattern that are needed to make all bonds and nodes isolated in a wide range of synthetic topologies. Each bar in the figure represents 100 different topologies given by the type of generation model used, the bonds-to-node ratio, and the number of nodes in the topology.

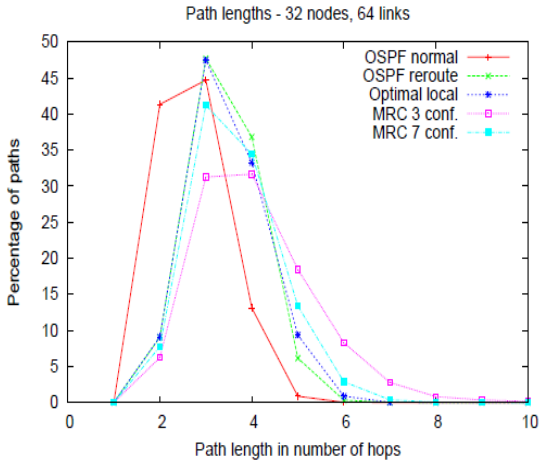
C. Interchange Results

1) Total network load:

This metric is related to the backup path length and represents the total interchange load in the network



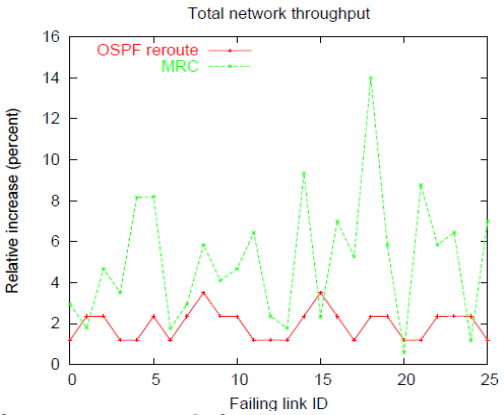
The sub-optimal backup paths given by MSP should result in an increased load in the network. Fig. shows the aggregate throughput of all the bonds in the COST239 network after a bond failure. The bond index on the x-axis shows which of the 26 bidirectional bonds has failed. The relative increase in the load compared to the failure-free case is given on the y-axis.



The simulations show that the load in the network increases about 5% on average after a failure when using MSP with 3 backup pattern, compared to a 2% increase with OSPF resteering. All interchange is recovered in this scenario, so the increased network load is solely caused by the longer paths experienced by the rerouted interchange.

2) Bond load distribution: Fig. 8 shows the bond load

Distribution in the COST239 network. Again, results for the



Failure free situation and for OSPF resteering are given. Results for OSPF reroutes and MSP using 3 backup pattern are averages for all 26 possible bond failures.

The simulations suggest that the bond load distribution in the network is similar when using MSP and after complete OSPF re-convergence.

VII. CONCLUSION

We have presented Multiple Steering Pattern as an approach to achieve fast recovery in IP networks. MSP is based on providing the routers with additional steering pattern, allowing them to forward packets along routes that avoid a failed component. MSP guarantees recovery from
any single node or bond failure in an arbitrary biconnected network. By calculating backup pattern in advance, and
operating based on locally available information only, MSP
can act promptly after failure discovery. MSP operates without knowing the root cause of failure, i.e., whether the ahead disruption is caused by a node or bond failure. This is achieved by using careful bond weight assignment according to the rules we have described. The bond weight assignment rules also provide basis for specification of a ahead procedure that successfully solves the last hop problem.

The performance of the algorithm and the ahead mechanism has been evaluated using simulations. We have shown that MSP scales well: 3 or 4 backup pattern is typically enough to isolate all bonds and nodes in our test topologies. MSP backup path lengths are comparable to the optimal backup path lengths MSP backup paths are typically zero to two hops longer. In the selected COST239 network, this added path length gives a network load that is marginally higher than the load with optimal backup paths. MSP thus achieves fast recovery with a very limited performance penalty.

References

[1] *D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols,"bSIGCOMM, Computer Communications Review, vol. 18, no. 4, pp. 106 114, Aug. 1988.*

[2] *A. Basu and J. G. Riecke, "Stability Issues in OSPF Steering," in Proceedings of SIGCOMM 2001, August 2001, pp. 225–236.*

[3] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed InternetnSteering Convergence," IEEE/ACM Transactions on Networking, vol. 9, no. 3, pp. 293–306, June 2001.

[4] C. Boutremans, G. Iannaccone, and C. Diot, "Impact of bond failures on VoIP performance," in Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video, 2002.

[5] D. Watson, F. Jahanian, and C. Labovitz, "Experiences with monitoring OSPF on a regional service provider network," in ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems. IEEE Computer Society, 2003, pp. 204–213.

[6] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," ACM SIGCOMM Computer Communication Review, vol. 35, no. 2, pp. 35 – 44, July 2005.

[7] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone network," in Proceedings of INFOCOM 2004, Mar. 2004.

[8] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs. reactive approaches to failure resilient steering," in Proceedings IEEE INFOCOM'04, Mar. 2004.

[9] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate bond overload as observed on an IP backbone," in Proceedings of INFOCOM'03, Mar. 2003, pp. 406–416.

[10] T. Przygienda, N. Shen, and N. Sheth, "M-ISIS: Multi topology (MT) steering in IS-IS," Internet Draft (work in progress), May 2005, draftietfisis-wg-multi-topology-10.txt.

[11] A. Asgari et al., "Scalable Monitoring Support forResource Management and Service Assurance," IEEE Network Mag., vol. 18, issue 6, Nov. 2004, pp. 6–18.

[12] N. M. MosharafKabirChowdury and R. Boutaba, "A Survey of Network Virtualization," Computer Networks,vol. 54, issue 5, Apr. 2010, pp. 862–76.

[13] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," IEEE JSAC, vol. 20, no. 4, May 2002, pp. 756–67.

[14] S. Uhlig et al., "Providing Public IntradomainTravel Matrices to the Research Community," ACM Sigcomm Comp. Commun. Rev. (CCR), vol. 36, no. 1, Jan. 2006, pp. 83–86.

[15] M. Caesar et al., "Dynamic Route Computation Considered Harmful," ACM Comp. Commun. Rev. (CCR), vol.40, no. 2, Apr. 2010, pp. 66–71.

[16] D. Xu, M. Chiang, and J. Rexford, "Bond-State PathWith Hop-By-Hop Advanceing Can Achieve OptimalTravelProduction," Proc. IEEE INFOCOM, Apr. 2008.

[17] P. Psenak et al., "Multi-Topology (MT) Path in OSPF," RFC 4915, June 2007.

[18] N. Wang, K-H. Ho, and G. Pavlou, "Adaptive Multitopology IGP Based TravelProduction with Near-Optimal Performance," Proc. IFIP Networking 2008.

[19] A. Kvalbein et al., "Multiple Path Pattern for Fast IP Network Recovery," IEEE/ACM Trans. Net., vol. 17, no. 2, 2009, pp. 473–86.

[20] *P. Psenak, S. Mirtorabi, A. Roy, L. Nguen, and P. Pillay-Esnault, "MTOSPF:Multi topology (MT) steering in OSPF," IETF Internet Draft (work in progress), Apr. 2005, draft-ietf-ospf-mt-04.txt.*