Fuzzy Systems and Soft Computing
ISSN : 1819-4362

IMPLEMENTATION OF MULTIPLE ROUTING CONFIGURATIONS FOR FAST RECOVERY IN IP NETWORKS

S. N. Santhalakshmi (HOD/BCA), Department of Computer Applications, Nandha Arts and Science College (Autonomous), Erode-52.
Dr. S. V. Elangovan (HOD/CHEMISTRY) Department of Chemistry, Nandha Arts and Science College (Autonomous), Erode-52

ABSTRACT:

As the Internet becomes increasingly integral to our communications infrastructure, the issue of slow convergence in routing protocols following network failures becomes more pressing. The main goal of this paper is to address the need for fast recovery from link and node failures using a new recovery scheme called **Multiple Routing Configuration (MRC)**.

MRC is designed to be connectionless, relying solely on destination-based hop-by-hop forwarding. It operates by storing additional routing information within the routers, enabling them to immediately forward packets over an alternative output link as soon as a failure is detected, without needing to wait for full network convergence.

Additionally, we propose a method for using estimates of traffic demands within the network to optimize the distribution of the recovered traffic. By doing so, we aim to reduce the likelihood of congestion when MRC is activated, ensuring more efficient network operation during failure recovery.

INTRODUCTION:

In recent years, the Internet has evolved from a specialized network into a widely used platform for everyday communication services. As a result, the demand for reliability and availability has increased significantly. A disruption in a critical network link can impact a large number of phone calls or TCP connections, leading to serious consequences. Ensuring rapid recovery from failures has always been a fundamental design objective of the Internet.

IP networks are inherently resilient because Interior Gateway Protocols (IGPs), such as OSPF, update routing information when the network topology changes due to a failure. This process, known as re-convergence, requires distributing the updated link state to all routers in the network domain. Each router then recalculates its routing table based on the new topology. However, this process is time-consuming, and failures typically cause a temporary period of routing instability. Since most network failures are short-lived, frequent re-convergence can lead to route flapping and further instability.

The IGP convergence process is inherently slow because it is reactive and affects all routers in the domain. To address this limitation, we introduce Multiple Routing Configurations (MRC), a proactive and localized failure recovery mechanism that restores network functionality within milliseconds. MRC enables immediate packet forwarding through pre-configured alternative next-hops upon detecting a failure. This allows the normal IP convergence process to be delayed, activating only when necessary to address non-transient failures. Since MRC does not require global re-routing, fast failure detection methods, such as fast hellos or hardware alerts, can be used without compromising network stability.MRC ensures recovery from any single link or node failure, which accounts for most network disruptions. It does not rely on identifying the specific cause of failure, whether it is a faulty link or a failed router. The core concept of MRC is to generate a set of backup network configurations based on the network topology and associated link weights. These configurations are designed so that when a failure occurs, the detecting node can safely reroute traffic through an alternate link without requiring immediate global re-convergence.

MRC assumes the use of shortest-path routing and destination-based hop-by-hop forwarding. However, shifting traffic to alternate links may cause congestion and packet loss in certain parts of the network. This limits the duration for which the proactive recovery mechanism can be used before the global routing protocol is updated, reducing the chances of handling transient failures without a full re-convergence.

To optimize traffic distribution after a failure, MRC assigns link weights independently in each backup configuration. This approach provides flexibility in routing recovered traffic. The specific backup configuration used depends on the failure event, allowing for tailored link weight adjustments that optimize traffic flow for different failure scenarios.

EXISTING WORK:

IP networks are inherently resilient, as interior gateway routing protocols like OSPF are designed to update forwarding details based on topology changes following a failure. This re-convergence relies on the complete dissemination of the updated link state across all routers within the network domain. Once the new state data is shared, each router independently computes fresh, valid routing tables. However, this network-wide IP re-convergence is a time-intensive procedure, and a link or node failure is usually accompanied by a phase of routing instability. A major challenge is that, since most network disruptions are brief, excessively rapid initiation of the re-convergence process can result in route oscillations and heightened network instability. The IGP convergence mechanism is sluggish because it is both reactive and comprehensive—it responds to a failure only after it occurs and requires the participation of all routers within the domain

PROPOSED WORK:

Multiple Routing Configurations (MRC) is a preventive and localized protection technique that enables recovery within milliseconds. MRC operates by constructing a limited set of backup routing configurations, which are utilized to redirect recovered traffic through alternative paths following a failure. These backup configurations differ from the standard routing setup as they assign specific link weights to prevent traffic from passing through certain areas of the network.

It is observed that if all links connected to a node are assigned sufficiently high link weights, traffic will be prevented from traversing that node. Consequently, the failure of that node will only impact traffic originating from or destined for it. Similarly, to exclude a link (or a group of links) from routing participation, an infinite weight is assigned, ensuring that its failure does not affect traffic flow.

The MRC methodology consists of three main steps. First, a set of backup configurations is generated, ensuring that each network component is excluded from packet forwarding in at least one configuration. Second, for each backup configuration, a conventional routing protocol like OSPF is applied to determine configuration-specific shortest paths and generate forwarding tables for every router based on these configurations. The use of a standard routing protocol ensures loop-free forwarding within each configuration. By employing a standard shortest path algorithm, each router constructs multiple configuration-specific forwarding tables.

For simplicity, we state that packets are forwarded according to a configuration, meaning they are routed based on the corresponding forwarding table. This paper discusses the approach of maintaining separate forwarding tables for each configuration, though we anticipate that more optimized solutions can be developed for practical implementation. When a router detects that a neighbouring node is no longer accessible through one of its interfaces, it does not immediately notify the entire network of the connectivity loss. Instead, packets that would have been transmitted through the failed interface are designated as belonging to a backup configuration and rerouted via an alternative interface toward their destination

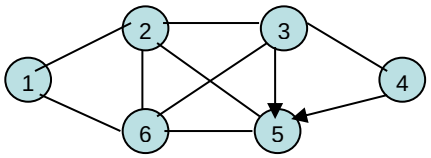
STEP 1: GENERATING BACKUP CONFIGURATIONS

In this section, we will first outline the essential requirements that must be imposed on the backup configurations utilized in MRC. Following this, we will introduce an algorithm designed to automatically generate these configurations. This algorithm is typically executed once during the network's initial deployment and subsequently whenever a node or link is permanently added or removed.

Configurations Structure:

MRC configurations are determined by the network topology, which remains consistent across all configurations, and the corresponding link weights, which vary between configurations. Formally, the network topology is represented as a graph **G = (N, A)**, where **N** is the set of nodes and **A** is the set of unidirectional links (arcs).

To ensure resilience against single failures, the topology graph **G** must be **bi-connected**. A configuration is defined by this topology graph along with its associated link weight function.



Node 5 is isolated by assigning a high weight to all its connected links, preventing it from forwarding transit traffic. Only traffic that is destined for or originating from the isolated node will use these restricted links.

ARGUMENTS:

Graph Definitions and Multiple Routing Configurations (MRC)

A **graph** is represented as **G = (N, A)**, where **N** is the set of nodes and **A** is the set of directed links (arcs).

Key Notations

- **Ci** – A graph configuration with link weights specific to configuration *i*.
- **Si** – The set of isolated nodes in configuration *Ci*.
- **Bi** – The backbone of configuration *Ci*.
- **A(u)** – The set of outgoing links from node *u*.
- **(u, v)** – A directed link from node *u* to node *v*.
- **pi(u, v)** – A shortest path between nodes *u* and *v* in *Ci*.
- **N(p)** – The set of nodes on path *p*.
- **A(p)** – The set of links on path *p*.
- **wi(u, v)** – The weight of link *(u, v)* in configuration *Ci*.
- **wi(p)** – The total weight of all links in path *p* under configuration *Ci*.
- **wr** – The weight assigned to a restricted link.
- **n** – The number of configurations to generate (algorithm input).

Configuration Definition

A **configuration Ci** is an ordered pair **(G, wi)**, where **wi** is a function that assigns an integer weight **wi(a)** to each link **a ∈ A**. The function follows the rule:

wi : A → {1, ..., wmax, wr, ∞}, where:

- **C0** (normal configuration) assigns "normal" weights to all links: **w0(a) ∈ {1, ..., wmax}**.
- **Ci** (backup configurations, i > 0) modifies link weights to prevent certain links and nodes from forwarding transit traffic while still ensuring connectivity for active nodes.

Traffic Regulation in Backup Configurations

To control traffic flow in backup configurations, certain links are assigned high weights:

- A **link a ∈ A** is **isolated** in **Ci** if **wi(a) = ∞**.
- A **link a ∈ A** is **restricted** in **Ci** if **wi(a) = wr**.
- **Isolated links** do not carry any traffic.
- **Restricted links** are used to prevent nodes from forwarding traffic. To isolate a node, all its attached links must be assigned at least the restricted weight **wr**.
- However, a node must not be completely isolated in all configurations to ensure it remains reachable.

The **set of isolated nodes** in **Ci** is denoted **Si**, while the set of non-isolated nodes is **Si = N \ Si**.

Node Isolation Condition

A node **u ∈ N** is considered **isolated** in **Ci** if:

$\forall (u,v) \in A, w_i(u,v) \geq w_r$

and at least one link satisfies:

$\exists (u,v) \in A, w_i(u,v) = w_r$

Rules for Isolated and Restricted Links

For all links **(u, v) ∈ A**:

- If **wi(u, v) = wr**, then the link connects an **isolated node** to a **non-isolated node**: (u∈Si∧ u ∈Si)or(v∈Si∧u∈Si)
- If **wi(u, v) =** ∞, then at least one of the nodes must be isolated: u ∈ Si or u ∈ Si

This means**Restricted links** always connect an **isolated node** to a **non-isolated node**.**Isolated links** either connect an **isolated node** to a **non-isolated node** or connect two **isolated nodes**.A link is always **isolated** in the same configuration as at least one of its attached nodes.

Algorithm:

The algorithm can be implemented either within a network management system or directly in the routers. As long as all routers have a consistent view of the network topology, they will compute the same set of backup configurations.

Description:

Algorithm 1 iterates through all the nodes in the network topology, attempting to isolate each node one at a time. A link is isolated during the same iteration as one of its connected nodes. The algorithm terminates when all nodes and links are isolated in exactly one configuration, or when a node that cannot be isolated is encountered.

a) Main Loop:

Initially, **n** backup configurations are created as copies of the normal configuration. Two queues are initialized: a queue of nodes (**Qn**) and a queue of links (**Qa**). The **Qn** contains all nodes in an arbitrary order, while the **Qa** starts empty but will eventually contain all links in the network. The method starts by returning and removing the first item from the node queue.

When a node **u** is being isolated in a backup configuration **Ci**, the algorithm first verifies that isolating the node will not disconnect the backbone **Bi**, according to the connectivity definition. The method **connected** (line 13) tests if each of **u**'s neighbors can still reach each other without passing through **u**, an isolated node, or an isolated link in configuration **Ci**. If the connectivity test passes, the function **isolate** is called, which tries to find a valid assignment of isolated and restricted links for node **u**.

If the isolation is successful, the modified configuration is returned, and the changes are committed (line 16). If isolation is not successful, no changes are made to configuration **Ci**.

Algorithm 1: Creating backup configurations.

1 for i ∈ {1 . . . n} **do**
2 Ci ←(G,w0)
3 Si ←∅
4 Bi ←Ci
5 end
6 Qn ←N
7 Qa←∅
8 i ←1
9 while Qn 6= ∅**do**
10 u ←first (Qn)
11 j ←i
12 repeat
13 if connected(Bi \ ({u},A(u))) **then**
14 Ctmp←isolate(Ci, u)
15 if Ctmp6= null **then**
16 Ci ←Ctmp
17 Si ←Si ∪ {u}
18 Bi ←Bi \ ({u},A(u))
19 i ←(i mod n) + 1
until u ∈Si **20 or** *i=j*
if u /∈Si **21 then**

22 Give up and abort
23 end
If node **u** is successfully isolated, the algorithm proceeds to isolate the next node. If isolation of **u** fails in one configuration, the algorithm continues trying to isolate **u** across all **n** configurations (line 20). If **u** cannot be isolated in any configuration after all attempts, the algorithm terminates with an unsuccessful result (line 22), indicating that a valid set of configurations with cardinality **n** cannot be built.

b) Isolating Links:
In addition to isolating node **u**, the algorithm attempts to isolate as many of its attached links as possible. It goes through the links **A(u)** attached to **u** (lines 2-3 in function **isolate**). An important invariant in the algorithm is that, at line 1, all links in **Qa** are attached to node **u**. The node **v**, which is at the other end of each link, may or may not be isolated in any given configuration (line 4).
This step ensures that the isolating of links is synchronized with the node isolation process, maintaining network connectivity where necessary and ensuring no invalid configurations are created.

Output:
We demonstrate that the successful execution of **Algorithm 1** results in a complete set of valid backup configurations.

Proposition:
If **Algorithm 1** terminates successfully, the generated backup configurations will adhere to the conditions (2) and (3).

Proof:
Links are assigned weights **wr** (restricted) or ∞ (isolated) only during the isolation of one of their attached nodes, ensuring that condition (3) is met. For restricted links, condition (2) requires that only one of the attached nodes is isolated. This invariant is maintained in **line 7** in the **isolate** function, where it is specified that if a node attached to a restricted link is isolated, the link itself must also be isolated. Therefore, it is impossible to isolate two neighbouring nodes without also isolating their connecting link.

Termination:
The algorithm iterates through all nodes, attempting to isolate each in one of the backup configurations. It will always terminate, either with a successful result or without success. If a node cannot be isolated in any configuration, the algorithm terminates unsuccessfully. However, the algorithm is designed so that any bi-connected topology will always terminate successfully, provided the number of configurations allowed is sufficiently high.

STEP 2: LOCAL FORWARDING PROCESS:
Given a sufficiently large **n**, the algorithm will generate a complete set of valid backup configurations. Let **C(u)** represent the backup configuration where node **u** is isolated, i.e., **C(u) = Ci ⇔ u ∈ Si**. Similarly, let **C(u, v)** denote the backup configuration where the link **(u, v)** is isolated, i.e., **C(u, v) = Ci ⇔wi(u, v) =** ∞. Assuming that **d** is the destination (egress) in the local network domain, we can analyze two cases based on the value of **v**:

1. **Case 1: If v ≠ d:**In this case, forwarding can be done in **configuration C(v)**, where both node **v** and the link **(u, v)** are avoided.
2. **Case 2: If v = d:**When **v** is the destination, the challenge is to handle the failure of the link **(u, v)** while ensuring that node **v** is still operational. The strategy is to forward the packet to **v** via a path that bypasses the failed link **(u, v)**.

Additionally, any packets that have already changed configurations (i.e., their configuration ID is different from the one used in C_0) and encounter a failed component on their forwarding path must be discarded. This prevents packet loops, even if node **d** itself fails.

Proposition:
Node **u** selects configuration **Ci** such that **v ∉N(pi(u, d))**, if **v ≠ d**.

Proof:
Node **u** selects **C(v)** in step 2. Since node **v** is isolated in **C(v)**, it will not appear in the shortest path **pi(u, d)**, as per the conditions outlined in the proposition above.

PERFORMANCE EVALUATION:

MRC requires routers to store additional routing configurations, and the amount of state required is directly tied to the number of backup configurations. Since routing in a backup configuration is restricted, MRC may provide backup paths that are longer than the optimal paths. These longer paths can increase the overall network load and end-to-end delay.

In contrast, full, global IGP re-convergence recalculates the shortest paths in the network after the failed component is removed. We use this global re-convergence performance as a benchmark to assess how closely MRC can match it. It is important to note that MRC achieves its performance immediately after detecting a failure, whereas IP re-convergence can take several seconds to complete.

CONCLUSION:

We have introduced Multiple Routing Configurations (MRC) as a method for achieving rapid recovery in IP networks. MRC enables routers to use preconfigured backup routing configurations, allowing them to reroute traffic without relying on the failed component. It ensures recovery from any single node or link failure in a bi-connected network.

By precomputing backup configurations and relying solely on locally available information, MRC responds immediately upon detecting a failure. It does not require identifying whether the failure is due to a node or link disruption, as it applies a structured link weight assignment to manage traffic redirection.

The effectiveness of MRC has been evaluated through simulations, demonstrating that it provides fast recovery with minimal impact on network performance.

REFERENCES:

1. 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.

2.Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "**Failure inferencing based fast rerouting for handling transient link and node failures,"** in *Proceedings of IEEE Global Internet*, Mar. 2005.

3. P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "**Local restoration algorithms for link-state routing protocols**," in Proceedings of IEEE International Conference on Computer Communications and Networks, Oct. 1999.