End-to-end Shared Restoration Algorithms in Multi-domain Mesh Networks

Zhiying Gao and Hassan Naser
Lakehead University, Thunder Bay, Ontario, Canada P7B 5E1
zgao@lakeheadu.ca, hnaser@lakeheadu.ca

Abstract

This paper presents two distributed end-to-end shared restoration algorithms in a multi-domain network environment. While the first proposed algorithm yields a pair of “link-disjointed” paths between any given pair of nodes in the network, the second algorithm computes a pair of “domain-disjointed” paths. To address the problem of limited information exchange among the domains, the multi-domain network is topologically aggregated as a single-domain network, called virtual path network, in which each domain is abstracted by its border nodes interconnected by point-to-point virtual paths. The performance of both algorithms are evaluated and compared through simulation experiments.

1. Introduction

Current protection and restoration mechanisms focus on the restoration of network traffic in the event of a physical link failure inside a single domain network. Powerful dynamic protection and restoration algorithms have been developed for single-domain networks with different topology configurations, such as mesh or ring. The majority of these algorithms are based on the exchange of detailed link-state information among the nodes inside the domain [1]. On the other hand, emerging multi-service data applications require high-bandwidth high-quality connectivity across multiple domains. Each domain is controlled by an independent and autonomous service provider. These applications necessitate the need for a new generation of highly intelligent survivable routing mechanisms to compute end-to-end paths and to perform functions of protection and bandwidth management across multiple domains. We introduce an architecture that provides protection services across multiple autonomous domains.

In a multi-domain network environment, considering the scalability of the network and the confidentiality of each domain, a domain may not wish to exchange detailed information about the state of its resources and the topology of the network with other domains. In this scenario, solutions to network protection and restoration based on a detailed information exchange will not be feasible.

Many research studies have explored routing schemes in multi-domain networks. A Path Computation Element (PCE) in each domain is introduced in [2], where an end-to-end path across domains is computed by the collaboration of PCEs in different domains. Reference [3] addresses the challenges of the standardized Border Gateway Protocol (BGP) in Quality of Service (QoS) routing. Then, they introduce an Inter-domain Routing Agent (IRDA) in each domain to advertise QoS information in their new inter-domain routing model. We noticed that the scope of the above referenced papers is limited to the routing of transport connections in multi-domain networks. Network survivability has not been taken into consideration in their proposed routing schemes.

In fact, only a limited number of research studies deal with the problem of multi-domain network survivability. As an example, a multi-domain network protection mechanism is proposed in [4], which is based on the establishment of independent protection mechanisms within individual domains and merging them at the domain boundaries. However, in comparison with end-to-end restoration, individual domain restoration is not capacity efficient, because, in the absence of detailed information, the local domain will restore all the traffic on a failed link even for the demands that did not request protection.

Reference [5] investigates the use of p-Cycle (pre-configured protection cycle) in a multi-domain network. With their p-Cycle protection scheme, the multi-domain survivability problem is decomposed into two levels: the lower intra-domain level and the upper inter-domain level. At the lower level, the intra-domain failure is recovered within its domain. At the upper level, p-Cycles are created and considered unchanged while the network is operated. If an inter-domain link
fails, the traffic will be routed over its corresponding $p$-Cycle. Generally, $p$-Cycle schemes require a large amount of capacity in each domain [5].

In this paper, we propose two novel solutions to end-to-end shared restoration in multi-domain networks. A sketch of a multi-domain network with four domains is shown in Fig. 1. Our first algorithm is designed to find a pair of link-disjointed paths between any given pair of nodes located in different domains in a multi-domain network environment. The second algorithm is designed to find a pair of domain-disjointed paths between these nodes. In both of our schemes, the network traffic is divided into two categories: local and transit (remote) traffic. Local traffic is the traffic that is exchanged between two nodes inside a single domain. The local traffic is routed over the links of the domain’s physical topology. Transit traffic is the traffic exchanged between two nodes in different domains. This traffic may thus cross one or more intermediate domains before reaching the destination domain.

In order to route the traffic in the multi-domain network, the multi-domain network is topologically aggregated to become a single-domain network, called virtual path network, in which each domain is abstracted by its border nodes interconnected by point-to-point virtual paths. The border nodes are nodes that have links to nodes in the neighboring domains. Virtual paths (VPs) are paths that are pre-computed within each domain between the border nodes of that domain. Fig. 2 illustrates a virtual path network created from the original multi-domain network shown in Fig. 1.

Each domain will only advertise limited information (such as the available capacity) about its VPs to other domains. Therefore, all domains will have the same image of the virtual path network, which consists of the border nodes of all domains, the VPs interconnecting the border nodes, and the inter-domain links connecting the border-nodes of adjacent domains. The following information about VPs must be advertised to support our proposal: 1) the identity of the border-nodes terminating each VP, 2) the available capacity on each VP, and 3) the link-disjointedness relationship between any two VPs in the same domain.

In order to route transit traffic, the multi-domain network is topologically aggregated to become a single-domain network, called virtual path network, in which each domain is abstracted by its border nodes interconnected by point-to-point virtual paths. The border nodes are nodes that have links to nodes in the neighboring domains. Virtual paths (VPs) are paths that are pre-computed within each domain between the border nodes of that domain. Fig. 2 illustrates a virtual path network created from the original multi-domain network shown in Fig. 1.

Each domain will only advertise limited information (such as the available capacity) about its VPs to other domains. Therefore, all domains will have the same image of the virtual path network, which consists of the border nodes of all domains, the VPs interconnecting the border nodes, and the inter-domain links connecting the border-nodes of adjacent domains. The following information about VPs must be advertised to support our proposal: 1) the identity of the border-nodes terminating each VP, 2) the available capacity on each VP, and 3) the link-disjointedness relationship between any two VPs in the same domain.

The rest of the paper is organized as follows: the first algorithm (Algorithm 1) is described in section II. The second algorithm (Algorithm 2) is described in section III. Section IV presents the simulation results, and section V concludes this paper.

2. Multi-domain restoration algorithm 1

For the simplicity of discussion, we assume that every domain is controlled by a local server. At the network configuration stage, the server computes the VPs inside that domain by using the Dijkstra’s algorithm based on the minimum number of hops routing criteria. The server then assigns capacity to each of these VPs, and advertises these VPs to all other domains. On the bases of the advertised information, each domain creates the virtual path network, which consists of the border nodes of all domains, the VPs interconnecting the border nodes, and the inter-domain links connecting the border-nodes of adjacent domains.

The following information about VPs must be advertised to support our proposal: 1) the identity of the border-nodes terminating each VP, 2) the available capacity on each VP, and 3) the link-disjointedness relationship between any two VPs in the same domain.
In each domain $d$, there is a matrix $L_d$ that records the link-disjointness relationship between any two VPs in that domain:

$$L_d = \begin{bmatrix}
0 & l_{12}^d & l_{13}^d & \cdots & l_{1j}^d \\
l_{21}^d & 0 & l_{23}^d & \cdots & l_{2j}^d \\
l_{31}^d & l_{32}^d & 0 & \cdots & l_{3j}^d \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
l_{d_1}^d & l_{d_2}^d & l_{d_3}^d & \cdots & 0
\end{bmatrix}$$

(1)

$J_d$ is the number of VPs in domain $d$. If VPs $i$ and $j$ are link-disjointed, the value of $l_{ij}^d$ will be 1, otherwise, it will be 0. The detailed information, such as the internal network topology and the number of hops of a VP, is hidden from one domain to the other.

For the transit traffic, there is a global matrix $K$ (shown below) that records the backup bandwidth reserved on VPs and inter-domain links. Element $k_{ij}$ in $K$ is the amount of transit backup bandwidth needed on $j$ if $i$ fails. Both $i$ and $j$ are VPs and/or inter-domain links. All domains have a copy of this matrix, which must be advertised frequently.

$$K = \begin{bmatrix}
k_{11} & k_{12} & k_{13} & \cdots & k_{1j} \\
k_{21} & k_{22} & k_{23} & \cdots & k_{2j} \\
k_{31} & k_{32} & k_{33} & \cdots & k_{3j} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
k_{d_1} & k_{d_2} & k_{d_3} & \cdots & k_{d_j}
\end{bmatrix}$$

(2)

For the local traffic, each domain $d$ records the backup bandwidth reserved on its intra-domain links in a private matrix $K_d$ shown below. Element $k_{mn}^d$ is the amount of backup bandwidth needed on link $n$ if link $m$ fails. Both $m$ and $n$ are intra-domain links in domain $d$. $N$ is the total number of links in domain $d$.

$$K_d = \begin{bmatrix}
0 & k_{12}^d & k_{13}^d & \cdots & k_{1N}^d \\
k_{21}^d & 0 & k_{23}^d & \cdots & k_{2N}^d \\
k_{31}^d & k_{32}^d & 0 & \cdots & k_{3N}^d \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
k_{N_1}^d & k_{N_2}^d & k_{N_3}^d & \cdots & 0
\end{bmatrix}$$

(3)

The total amount of backup bandwidth ($B_j$) needed on an inter-domain link $j$ or VP $j$ to restore the transit traffic is indeed the maximum of all elements in column $j$ of matrix $K$:

$$B_j = \max_{ij} k_{ij}$$

(4)

If $j$ is a VP in domain $d$, $B_j$ will also be needed on every component link $n$ of $j$. Because, in general, link $n$ can be on more than one VP, we denote by $S_{VP}(n)$ the set of all VPs that cross link $n$. Hence, the total backup bandwidth needed on link $n$ for the transit traffic is:

$$B_{n_{transit}}^d = \sum_{j \in S_{VP}(n)} B_j$$

(5)

The total backup bandwidth needed on link $n$ to restore the local traffic is the maximum of all elements in column $n$ of matrix $K_d$. That is:

$$B_{n_{local}}^d = \max_{mn} k_{mn}^d$$

(6)

From (5) and (6), the total amount of backup bandwidth needed on link $n$ is therefore:

$$B_n^d = B_{n_{transit}}^d + B_{n_{local}}^d$$

(7)

A dynamic two-step path computation algorithm is used to compute a pair of link-disjointed paths for every newly arrived demand $r$ between a given pair nodes in the multi-domain network. The source domain of demand $r$ has access to the information about its local network as well as about the virtual path network outside of the source and destination domains. However, it does not have access to the detailed internal information about the destination local network.

Therefore, for both of the primary and backup path computations, the server in the source domain uses the physical network of the source domain and the virtual path network outside the source and destination domains to compute a path from the source node to every border node in the destination domain. Among these paths, the least-cost path is selected, and a path setup message is sent to the border-node in the destination domain that terminates the selected path. The border-node will forward this message to the server in the destination network, which will use the physical network of the destination domain to compute the path-segment from the selected border node to the actual destination node.

Therefore, for every newly arrived demand $r$, a pair of link-disjointed paths will be computed dynamically using the following two-step path computation algorithm.

**Step 1 (Primary Path Computation):** for links in the source or destination domains, the cost of choosing link $n$ on the primary path is determined according to the following function:
The term ‘d’ denotes a domain, which can be the source or destination domain. b is the amount of requested bandwidth by demand r, and \( A_d \) is the available capacity on link n in domain d. \( C_p(d) \) is the cost of choosing link n in domain d to be on the primary path of demand r. The cost of link n is set to 1 if there is enough available capacity on the link to accommodate demand r. Otherwise, it is set to \( \infty \).

For inter-domain links and VPs outside of the source and destination domains, the cost is determined according to the following function:

\[
C_p(i) = \begin{cases} 
1 & b \leq A_i \\
\infty & \text{otherwise} 
\end{cases} 
\]  

Where \( i \) denotes an inter-domain link or a VP outside of the source and destination domains of demand r, and \( A_i \) is the available capacity on i. If \( i \) is a VP, \( A_i \) is the minimum available capacity of all the component links of i. \( C_p(i) \) is the cost of choosing \( i \) to be on the primary path of demand r. The cost of \( i \) is set to 1 if there is enough available capacity on \( i \) to accommodate demand r. Otherwise, it is set to \( \infty \).

**Step 2 (Backup Path Computation):** if there exists a primary path for demand \( r \), a backup path will be computed based on the cost functions described below. For the links in the source and destination domains, the link cost for backup path computation is determined according to the following function:

\[
C_b(d) = \begin{cases} 
\infty & n \in S^d_p(r) \\
\varepsilon & T_n^d \leq B_{\text{new}}^d \\
T_n^d - B_{\text{new}}^d - B_{\text{new}}^d & T_n^d - B_{\text{new}}^d \leq A_n^d \\
\infty & \text{otherwise} 
\end{cases} 
\]  

\( C_b(d) \) is the cost of choosing link n in domain d to be on the backup path. \( S^d_p(r) \) is the set of links in domain d that are on the primary path of demand r. \( T_n^d \) is the maximum amount of backup bandwidth needed on link n if a link in \( S^d_p(r) \) fails. \( T_n^d \) will simply be:

\[
T_n^d = b + \max_{\forall n \in S^d_p(r)} \{ k_n^d \}. 
\]  

In equation (10), if link \( n \) is in \( S^d_p(r) \), the cost of link \( n \) is set to \( \infty \). Otherwise, the cost is set to a very small number (\( \varepsilon \)) if \( T_n^d \) is less than \( B_{\text{new}}^d \). In this case, demand \( r \) can be restored on link \( n \) without need to reserve any additional backup bandwidth on this link. If neither of the above conditions is satisfied, the cost is set to \( T_n^d - B_{\text{new}}^d \) if this quantity is less than the available capacity on link n. In this case, \( T_n^d - B_{\text{new}}^d \) is the amount of additional backup bandwidth required on link \( n \) in order to restore demand \( r \). If the available capacity on link \( n \) is not adequate to accommodate this additional bandwidth, the cost of link \( n \) is set to \( \infty \) in the 4th term.

For inter-domain links and VPs outside of the source and destination domains, the link cost for backup path computation is determined according to the following function:

\[
C_b(j) = \begin{cases} 
\infty & j \in S^d_p(r) \\
\varepsilon & \forall d \in M_p(r) \land \forall i \in S^d_p(r) : i, j \in V(d) \land t_i^d = 0 \\
T_j - B_j & T_j - B_j \leq A_j \\
\infty & \text{otherwise} 
\end{cases} 
\]  

\( C_b(j) \) is the cost of choosing \( j \) to be on the backup path of demand \( r \). \( S^d_p(r) \) is the set of all VPs and inter-domain links that are on the primary path of demand \( r \). \( M_p(r) \) is the set of all intermediate domains that the primary path of demand \( r \) crosses through. \( V(d) \) is the set of VPs in domain \( d \). \( T_j \) is the maximum amount of backup bandwidth required on \( j \) if a VP or an inter-domain link in \( S^d_p(r) \) fails:

\[
T_j = b + \max_{\forall n \in S^d_p(r)} \{ k_n^d \}. 
\]  

In equation (12), the cost of \( j \) is set to \( \infty \) if it is already on the primary path. The cost of \( j \) is also set to \( \infty \) if it is not link-disjointed with any of the VPs on the primary path. If neither of the above conditions is satisfied, the cost is set to a very small number (\( \varepsilon \)) if \( T_j \) is less than \( B_j \). In this case, demand \( r \) can be restored on \( j \) without need to reserve any additional backup bandwidth on \( j \). Otherwise, the cost is set to \( T_j - B_j \) if this quantity is less than the available capacity on \( j \). If the available capacity on \( j \) is not adequate to accommodate the additional bandwidth (\( T_j - B_j \)), the cost of \( j \) is set to \( \infty \) in the 5th term.
3. Multi-domain restoration algorithm 2

Algorithm 2 computes a pair of “domain-disjointed” paths between two given nodes using the same stages of processing as used in Algorithm 1. We only describe the differences between the two algorithms below.

Because with Algorithm 2, the primary and backup paths will be domain-disjointed, the information about the link-disjointness relationship between VPs in a domain will not be used by the path computation engine. Therefore, domains do not need to store or advertise the content of matrix (1).

For the backup path computation, Algorithm 2 uses the cost function (14) described below for the inter-domain links and transit VPs, instead of the cost function (12) used in Algorithm 1.

\[
C_p(j) = \begin{cases} 
\infty & j \in S_p(r) \\
\infty & \forall d \in M_p(r) : j \in I(d) \\ 
\epsilon & T_j \leq B_j \\
T_j - B_j & T_j - B_j \leq A_j \\
\infty & \text{otherwise}
\end{cases} \quad (14)
\]

The cost of inter-domain link or VP j is set to \( \infty \) if it is already on the primary path of demand \( r \). With the second term, the cost is also set to \( \infty \) if \( j \) is an inter-domain link that emanates from a domain that is on the primary path of demand \( r \). \( I(d) \) denotes the set of all inter-domain links emanating from domain \( d \). With the third term, the cost is set to a very small number (\( \epsilon \)) if \( T_j \) is less than \( B_j \). If neither of the above conditions is satisfied, the cost is set to \( T_j - B_j \) if this quantity is less than the available capacity on \( j \). If the available capacity on \( j \) is not adequate to accommodate the additional bandwidth \( (T_j - B_j) \), the cost of \( j \) is set to \( \infty \) in the fifth term.

4. Simulation results

The proposed algorithms are simulated over a multi-domain network which is based on the NSF (National Science Foundation) network. NSF is one of the representative North American backbone networks. The simulated multi-domain network is shown in Fig. 3, which resembles the NSF network topology. In our simulation, every node in the figure represents a domain. Hence, there are 16 domains interconnected by 25 inter-domain links in the multi-domain network shown. All domains have identical topology, consisting of 10 nodes interconnected by 13 intra-domain links (shown in the dashed callout in Fig. 3.).

Fig. 3. Network topology

Capacities of all inter-domain and intra-domain links are set to 2.5 Gbps. All demands request an identical amount of 100 Mbps bandwidth, which is the current fast Ethernet data rate. For each demand, the source and destination domains are generated uniformly randomly over all domains. The source node and the destination node inside the source and destination domains are also generated uniformly randomly over all the nodes in the source and destination domains.

We used the following metrics to evaluate the performance of the proposed schemes: the average reserved primary capacity per link (which is the sum of the reserved primary capacity on each link divided by the number of links), the average reserved backup capacity per link, the average link load, and the number of blocked demands in the network. The average link load is computed as the average of the sum of the reserved primary and backup capacities on each link divided by the total capacity of that link, averaged over all links.

Fig. 4 shows the average reserved primary and backup capacities per link, Fig. 5 shows the average link load, and Fig. 6 shows the number of blocked demands, each as a function of the number of generated demands, which is varied from 50 to 300 at a step of 50. Figs. 4 and 6 show that Algorithms 1 and 2 have very close performances in terms of the average reserved primary capacity and the number of blocked demands. However, when the backup capacity is considered, Fig. 4 shows that Algorithm 2 consumes more backup capacity than Algorithm 1, for the same number of generated demands. One reason for this could be that the domain-disjointed constraint satisfaction problem of Algorithm 2 is more difficult to meet than the link-disjointed constraint problem associated with Algorithm 1. As a result, the backup paths computed by Algorithm 2 are generally longer in
length (hops) and the backup capacities on these paths are less shared than the corresponding backup paths computed by Algorithm 1. For this reason, the average link load of Algorithm 2 is slightly higher than that of the Algorithm 1, as shown in Fig. 5.

5. Conclusion

We proposed two capacity-constrained shared restoration algorithms for computing failure-disjoint primary and backup paths in a multi-domain network environment. With both algorithms, the multi-domain network is topologically aggregated to become a single-domain network, in which each domain is represented by its border nodes interconnected by point-to-point virtual paths. While the primary and backup paths computed by the first algorithm are "link-disjointed", they are domain-disjointed when the second algorithm is used. One advantage of the proposed algorithms over the existing restoration algorithms is that they require much less amount of link-state information to be advertised between the domains. This will reduce the routing message overhead and make the proposed algorithms to be scalable to large inter-domain networks.

6. References


