Adaptive protection methods

ATTILA MITCSENKOV, DIÁNA MESKÓ, TIBOR CINKLER

Budapest University of Technology and Economics
High-Speed Networks Laboratory, Department of Telecommunications and Media Informatics
{mitcsenkov, mesko, cinkler}@tmit.bme.hu

Keywords: adaptive, shared, protection, resilience, rearrangement

The bandwidth requirements of modern integrated networks solidly grow, and at the same time the reliability plays an increasingly important role. Various methods are known and under development to ensure survivability. Our methods deal with this issue. The main feature of the proposed protection rearrangement framework is that since the protection paths do not carry any traffic until a failure occurs, they can be adaptively rerouted (rearranged) as the traffic and network conditions change.

1. Introduction

Computer networks since its beginning has gone through strong evolution: in the late 1970's the network was used for communication between a few scientific institutions – and today it is part of our daily life, plenty of new services have arisen (e.g. peer-to-peer systems, grid computing, Video on Demand, Voice over IP, e-banking services, etc.), the number of users increases rapidly, at a guess it doubles in every year. The convergence of computer, telecommunication and broadcast networks also pose new challenges [10,11].

Therefore modern transport networks raise new problems, not only in the field of bandwidth requirements, but also the quality and the resilience of the offered services plays an increasingly important role. Service disruption is no longer tolerated by business or industry; therefore survivable services have to be provided. The failure of any part of the network has to remain invisible for the customers.

The network management can ensure survivability using various methods – the network operator should decide which one to use. The alternatives will be discussed in Section 2. Sections 3, 4 and 5 present the investigated solutions, designed for different conditions and offering different performance. Section 6 presents and evaluates the numerical results, Section 7 concludes our work.

2. Resilience Strategies

According to the previously mentioned requirements for survivability, the network should be prepared for failures, to be able to make them invisible for customers by eliminating their effects (e.g. service interruption, data loss, increasing delay). Various resilience strategies are known that deal with these requirements. A brief overview of them is needed for the subsequent description of our methods. Here we focus only on single failures, which is a widespread assumption in the literature as well. However, after some modifications our methods can deal with multiple failures [6].

Protection vs. Restoration: While using protection methods, protection (backup) paths are defined in advance, and in case of failure the traffic is immediately switched to the corresponding backup path. In case of restoration the protection paths are sought only when failures occur. It results in a thriftier operation, but it might fail when establishing backup paths due to insufficient resources.

In case of *Dedicated Protection* each working path has a separate backup path, with exclusively reserved resources. Conversely, *Shared Protection* means commonly used resources among backup paths of different working paths. It results in a thriftier but slower method with higher complexity.

Regarding the part of the network to be protected, the following classification can be made: (1) *path protection (or end-to-end protection),* means the working path is protected by one path that is totally disjoint from the working one [1,2]; (2) *link protection,* when all the traffic from the failed link is re-routed between the ends of that link; (3) *sub-network protection,* where the network is clustered into protection domains (sub-networks) that define the ends of protection segments; and (4) *segment protection* [3], when the working path is divided into segments, and protected by a few backup paths covering them. These backup paths (segments) should of course jointly cover the whole working path, and should be at least partially disjoint from the working path.

In case of *static* protection predefined working and protection paths are used for each node-pair. When these paths are redefined from time to time, we refer to this method as *dynamic* protection/restoration. *Adaptive* methods are able to alter the previously routed protection paths, and define new paths for every new demand, adapting to the varying network and traffic conditions. This is the slowest and most complex approach, but it offers the strongest control over the resources of the network.

According to the above definitions, our protection methods

– are shared protection methods
– are dynamic or adaptive
– offer path or segment protection
– guarrantee survival of any single failure, but work for some multiple failure patterns as well.

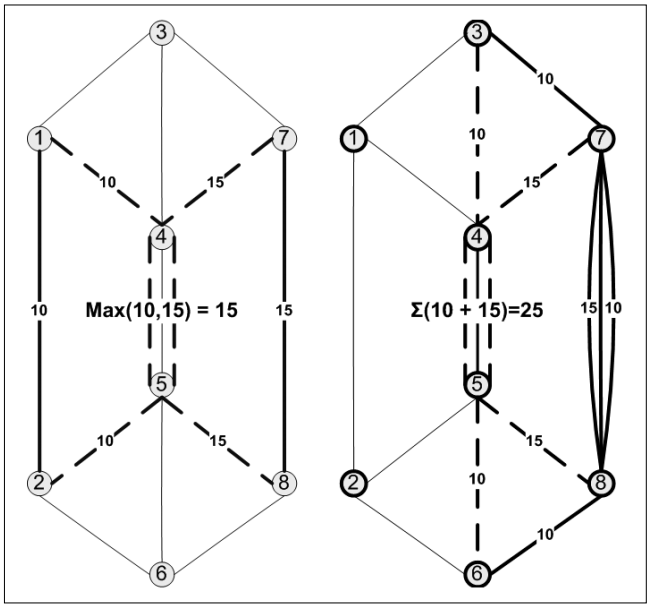
2.1. Spare Capacity Allocation

The main benefit of shared protection methods is the thrifty resource utilization. Resource sharing is allowed among different protection paths. However, we should avoid the case when different demands would switch over to the same protection path simultaneously. If only single failures occur in the network, then two disjoint (independent) working paths cannot fail simultaneously, therefore protection paths belonging to disjoint working paths can share resources. It results in a thrifty resource usage, without losing the survivability in case of any single failure [4].

Let us show a detailed example in *Figure 1*: two demands are given with capacities of 10 and 15 units. Working paths are denoted by solid lines and protection paths by dashed lines. In the first case (figure on the left side) these demands have to be routed between nodes 1-2 and 7-8 – it means the working paths have no common resources, i.e. the protection paths can not be simultaneously activated if only single failures occur. Therefore the protection paths can share resources among link 4-5, thus max (10;15) = 15 units of capacity is allocated. In the second case (figure on the right) these demands have to be routed between nodes 3-8 and 6-7, and both working paths use link 7-8. In this case, if link 7-8 fails, both demands will use their protection paths, therefore 10+15=25 units of capacity have to be allocated for protection paths on that link 7-8.

The capacity C_l of each link in the network is divided into three parts *(Figure 2)*: the allocated capacity for working paths (*C'*), the capacity used by protection paths, that can be used by another ones according to the above described criteria (*C"*), and the free, unused capacity: *C-C'-C"*. Different costs are assigned to these parts, while the re-use of capacity reserved for protection paths means no extra allocation, unlike the use of the free capacity-range. It results in a cost function with two linear segments.

Figure 1. Spare Capacity Allocation



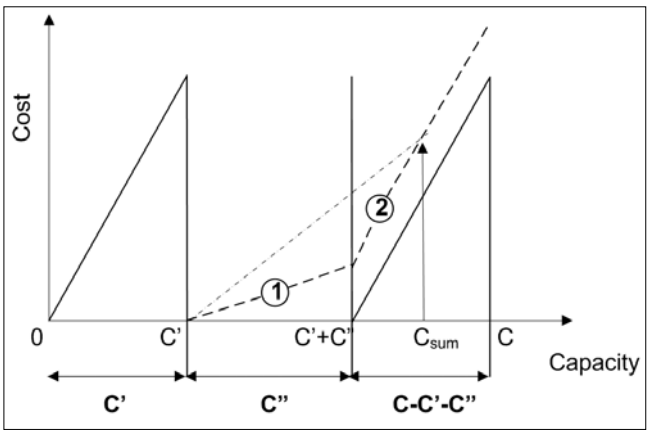


Figure 2. Different Capacity Domains

3. Dynamic Algorithms

Three different versions of a shared protection method were implemented as references and as the basic elements of more complex algorithms, with different restrictions on the protection paths, ensuring more and more flexibility in small steps.

The starting point was the Failure-Independent Shared Path Protection (F-I SPP), where after routing the working path, a single end-to-end disjoint protection path is needed, with the lowest cost in the sense of the above described capacity cost function *(Figure 2)*. In case of failure it re-routes the traffic to this single protection path regardless of the location and type of the failure.

The next step was the Failure-Dependent Shared Path Protection (F-D SPP), where after routing the working path, different protection paths are assigned for the failure of each link on the working path. The protection paths of the same working path are allowed to share links with each other and other working paths considering the above described criteria. However, all protection paths have to be end-to-end disjoint from their working ones.

Finally the failure-dependent Partially Disjoint Shared Path Protection (PDSP) was implemented. It assigns different protection paths for a single working path like the F-D SPP, but it has fewer restrictions for the protection paths. This method can be classified as link protection and also as segment protection, while it assigns different protection paths for the failures of any link on the working path. These protection paths have to substitute a certain segment of the working path; anyway they can use the rest of the links of the working path.

The requirement of the previous methods (F-I/F-D SPP) was that protection paths are disjoint from all links of the corresponding working path. PDSP requires disjointness only from the link of which failure it has to protect against. In this manner the protection paths can have common links with their working ones, except for only one link – and this is the difference between SPP and PDSP. Due to the fewer restrictions, ranging from F-I SPP through F-D SPP to PDSP the results are expected to improve in this particular order, especially for the protection path establishment in the network.

The detailed operation of the failure-independent SPP can be read in the next subsection, followed by algorithms F-D SPP and PDSP.

3.1. Failure Independent Shared Path Protection (F-I SPP)

The basis of the more sophisticated methods, the F-I SPP will be described now. It works as follows:

• **Step 1:** For any new demand (o_{new}):
→ Use Dijkstra's algorithm to find the shortest working path. If not successful, the demand will be blocked, go to *Step 1*.
→ Hide all the edges of the working path (*)

• **Step 2:** For all edges *l'* of the working path and for all edges *l"* of the network (except hidden edges) compute capacity $C_{l';l"}$, which represents the amount of available shared capacity for protection paths on link *l"* when link *l'* fails

• **Step 3:** Find the maximum of $C_{l';l"}$ for all *l'* found so far – this will be the value $C_{l"}$ (**). This is a failure-independent method; the same protection path will be used in case of any failure, therefore when determining the available shared capacity the worst case has to be considered to have a suitable solution for every failure.

• **Step 4:** Calculate the cost increment required for routing the protection path of demand o_{new} with bandwidth requirement b_o based on $C_{l"}$ along all the links *l"* of the network. It is the sum (C_{sum}) of the available shared capacity to be used (*C"*) with a lower linear cost (marked with 1 in *Figure 2,* and the amount of unused capacity (*C-C'-C"*) that has to be allocated with a higher linear cost (marked with 2 in *Figure 2*).

• **Step 5:** Use Dijkstra's algorithm to find the optimal protection paths, based on the cost increments described in *Step 4*.

• **Step 6:** If succeeded, allocate the new paths; otherwise de-allocate resources for terminated connections and update capacity allocations; demand o_{new} will be blocked. If there are more demands go to *Step 1*.

In essence, Failure-Independent Shared Path Protection (F-I SPP) is a really fast and simple shared protection method. Unlike the adaptive methods, it cannot reroute the previously allocated protection paths. SPP is a failure-independent method, thus in case of failure the traffic is switched to the same backup path, irrespective of the location and type of the failure.

The following two methods are modifications of F-I SPP.

3.2. Failure Dependent Shared Path Protection (F-D SPP)

The failure-dependent version differs in the number of protection paths to be allocated for a demand – the F-D SPP assigns separate protection paths for the case of failure of any links on the working path. Therefore, a modification of *Step 3* (marked with **) is needed: no maximization of the $C_{l';l"}$ values is necessary, and *Steps 4 and 5* must be applied for every links of the working path. If all executions of Dijkstra's algorithm in *Step 5* succeeded, the working and protection paths have to be allocated, otherwise the demand has to be blocked.

3.3. Partially Disjoint Shared Path Protection (PDSP)

This one is a shared protection method operating on segments of the network, i.e. segments of the working paths. It means a difference in the set of edges not available for the protection paths. It is a failure-independent method, working with separate protection paths for the case of the failure of any link on the working path. Therefore, it works like F-D SPP described above, but a modification is needed in *Step 1* (marked with *): not all the edges of the working path, but the only one link *e'* has to be hidden while establishing the protection path belonging to *e'*. It allows the algorithm to use any other links of the network for the protection path.

Table 1. Classification of algorithms

	Path Protection	Segment Protection
Dynamic	Shared Path Protection **(F-I/F-D SPP)**	Partially Disjoint Shared Path Protection **(PDSP)**
Adaptive	Shared Path Protection with Link Doubling **(SPP-LD)**	Partially Disjoint Shared Path Protection with Link Doubling **(PDSP-LD)**

4. Adaptive Methods

We presented the dynamic methods: F-I/F-D SPP and PDSP. In the next subsections, we explain the adaptive versions of these path and segment-protection methods. First the necessary modeling trick (Link Doubling – LD) to linearize the problem will be presented, followed by the Mixed Integer Linear Program (MILP) formulation of protection rearrangement and by our two proposed methods, namely Shared Path Protection with Link Doubling (SPP-LD) and Partially Disjoint Shared Path protection with Link Doubling (PDSP-LD).

4.1. Link Doubling (LD)

When rearranging the protection paths, a serious problem arises: the calculation of the available shared capacity ($C_{l';l"}$) hardly depends on the previously allo-

cated protection paths. However, in case of adaptive methods multiple protection paths have to be determined or altered simultaneously, and it also affects the previously allocated protection paths. Therefore, the calculation of the available shared capacity cannot be based on the view of previously routed demands.

The *Minimal Cost Multi-commodity Flow* (MCMCF – [9,12]) problem deals with this problem. Numerous solutions can be found in the literature, e.g. some heuristics, iterative solutions or Integer Linear Programming (ILP) [13]. Our methods are based on ILP [14].

Our cost functions are not linear, but have two linear segments *(Figure 2)*. ILP needs linear cost functions, therefore auxiliary variables are needed. These extra variables can be illustrated by the Link Doubling modeling trick.

The composite cost function of the edge between nodes A and B actually belongs to two different parts of capacity, and two appropriate linear cost functions. These two can be separated: any edge of the network should be substituted by two parallel edges, one for each capacity range. The lower cost has to be assigned to the shared resources, the higher cost to the unused resources as shown in *Figure 2*. This way the number of edges doubles, therefore the complexity of the problem gets higher, but ILP can be used for the MCMCF problem.

4.2. MILP Formulation

Protection rearrangement means that a part of the previously allocated protection resources has to be deallocated, and then allocated again simultaneously with the new demand(s). A set of demands to be rearranged is selected for the case of any link failure in the network. This way the system of protection paths can be adapted to the altering network load.

To handle the above described special MCMCF problem a proper MILP (Mixed Integer Linear Program) formulation is needed. Its role is to find an optimal solution for simultaneous routing of different protection paths in the network based on the two-segment linear cost functions, considering the disjointness criteria for working and protection paths, and the flow conservation and link capacity constraints as well.

Objective:

$$\min \sum_{o \in T_e} \left\{ \sum_{l \in E^{free}} x_l^o w_l + \sum_{l \in E^{sh}} x_l^o \gamma_l \right\} \quad (1)$$

E^{sh} stands for the edges created by LD, representing the shared capacity of the links, and E^{free} stands for edges representing the unused capacity. As described above, a set of demands (T_e) is selected to be rearranged for the failure of any link (e) in the network. The amount of capacity needed by the protection path of demand o on link l is represented by variable x_l^o.

The two-segment capacity cost function is given by the cost-coefficients for shared (γ_l) and unused (w_l) capacity. By setting the γ_l/w_l proportion the priorities for using shared capacity can be affected. If γ_l is close to zero, protection sharing is forced. It leads to thriftier operation, but it may result in longer protection paths, avoiding the edges in E^{free}. If γ_l is close to w_l, the use of sharable capacity is not preferred over unused capacity. Extensive simulations have shown that the best results can be achieved by setting $\gamma_l/w_l \approx 0.1$, $\forall\, l \in E$.

Subject to:

$$\sum_{o \in T_e} x_l^o \le C_l - C'_l - C''_l \text{ for all links } l \in E^{free} \quad (2)$$

$$\sum_{o \in T_e} x_l^o \le C''_l \text{ for all links } l \in E^{sh} \quad (3)$$

$$\sum_{\forall j \in V, j=i} x_{ij}^o - \sum_{\forall k \in V, k=i} x_{ki}^o = \begin{cases} 0 \text{ if } i \ne s_o \wedge i \ne d_o & \text{for all nodes } i \in V \\ b_o \text{ if } i = s_o & \text{and demands } o \in T_e \\ -b_o \text{ if } i = d_o \end{cases} \quad (4)$$

$$0 \le x_l^o \le b_o \text{ for all links } e \in E \text{ and demands } o \in T_e \quad (5)$$

$$\sum_{\forall k \in V, k=i} x_{ki}^o = b_o \cdot z_i^o \text{ for all nodes } i \in V \text{ and demands } o \in T_e \quad (6)$$

$$z_i^o \in \{0,1\} \text{ for all nodes } i \in V \text{ and demands } o \in T_e \quad (7)$$

Equations (2) and (3) represent the capacity constraints for the free and sharable capacities, Eq. (4) is the well known flow conservation constraint. Equation (5) gives a constraint for the x variables, which may not exceed the bandwidth of the corresponding demand. Equation (6) allows flow splitting only between the parallel edges created by LD, using an auxiliary binary variable, z_i^o (Eq. 7).

4.3. Shared Path Protection with Link Doubling (SPP-LD)

This method is the adaptive extension of the failure-dependent shared path protection (F-D SPP).

The basic idea of these adaptive methods is the ability to rearrange the protection paths, from time to time. It allows us to exempt overloaded network elements by moving a segment of the load to other parts of the network. It allows higher control over the network load conditions. Considering that protection paths do not carry real traffic, they are just for backup purposes, altering them does not cause service disruptions, therefore it remains invisible for the customers.

The more protection paths are routed simultaneously, the better results are expected, however the complexity of the problem grows exponentially. Therefore to rearrange all of them at a time is not possible, but selecting a proper subset of protection paths makes it suitable.

New protection paths are assigned to the failure of any link in the network, whereas these are failure-dependent methods. When determining the protection paths for link e, every protection path using e is rearranged. This will be the T_e subset of rearranged paths.

The SPP-LD algorithm works as follows:

- **Step 1:** For any new demand (o_{new}):
  → Use Dijkstra's algorithm to find the shortest working path
- **Step 2:** For the edges e of the working path:
  → Hide temporarily link e.
  → Set T_e to contain the new demand o_{new} and all the demands using e as a part of their working paths.
- **Step 3:** De-allocate protection paths of T_e.
- **Step 4:** Execute the MILP with an added path diversity constraint for all demands in T_e to find the protection paths for the failure of link e
  → The path diversity constraint prevents protection paths to use resources of the working path of corresponding demand, in case of SPP-LD it is as follows:
  $x_e^o = 0$ for all demands $o \in T_e$ and links $e \in WP_o$
  → If the ILP is feasible, it means in case of the failure of link e all the protection paths of the previously routed demands in T_e and the new demand can be allocated. If more links, go to *Step 2*, otherwise allocate the working and protection paths, and go to *Step 1*.
  → If the ILP is infeasible, the demand cannot be allocated, the previous state of the network will be restored and the demand o_{new} will be blocked.

4.4. Partially Disjoint Shared Path Protection with Link Doubling (PDSP-LD)

This method is the adaptive extension of the partially disjoint shared path protection (PDSP), described in 3.3.

It is also based on the rearrangement of protection paths, however differs from SPP-LD in its disjointness criteria. The working and protection paths do not have to be end-to-end disjoint, as it was in the case of path protection. It assigns different protection paths to any link of the working path for its failure, and as it was stated, these methods deal only with single failures. Therefore, it is enough to make the protection paths disjoint from the corresponding link, and allow them to use any other links, while two different links of the working path may not fail simultaneously.

Although the operation of the PDSP-LD is very similar to SPP-LD, the path diversity constraint of *Step 4* described in the previous paragraph differs:

$$x_e^o = 0 \text{ for all demands } o \in T_e$$

This prevents the protection paths to use the examined link of the working path, but allows them to use any other links of the network, as it was described above.

5. Semi-Adaptive Methods

Adaptive methods afford higher flexibility and more control over network load conditions, as the results will show. From the aspect of performance these methods perform well, as expected.

However there are some problems from the aspect of applicability. The complexity of integer linear programming results in seriously increasing computational times, depending on the size, connectivity and other attributes of the network. As the simulations show the required time for establishing working and protection paths in average for any single demand may be about 1.0-10.0 seconds of order of magnitude. And of course the larger the network is the longer times are needed *(Table 2)*.

It makes adaptive methods useless for certain services and conditions, therefore a trade-off between the performance and flexibility of adaptive algorithms and the low complexity of the dynamic methods is needed.

This trade-off can be realized in multiple ways, we introduce here a really obvious solution. The dynamic and adaptive algorithms use the same routine for working paths, but differ in the method for determining protection paths. Therefore, the working path of any new demand can be established in the common way, and then, first the dynamic routine will be used for the protection paths. In case of success the resources will be allocated.

The adaptive resource rearrangement mechanism will be applied only in case when the dynamic routine fails. Clearly it results in a faster algorithm, while in many cases the resource rearrangement mechanism is unnecessary, however it still has the ability of resource rearrangement.

The detailed operation of the Semi-Adaptive Shared Path Protection (S-A SPP) and Semi-Adaptive Partially Disjoint Shared Path Protection (S-A PDSP) is described in the following paragraphs.

5.1. Semi-Adaptive Shared Path Protection (S-A SPP)

The concurrent use of path- and segment-protection results in a complicated administration through the simultaneously established protection paths because different ILP-constraints are needed for path and segment protection. Furthermore in some cases path protection can be desired because its simpler management requirements, and a hybrid solution is not acceptable.

Therefore the investigation of a semi-adaptive path protection method is reasonable. It works as follows:

- **Step 1:** For any new demand (o_{new}):
  → Use Dijkstra's algorithm to find the shortest working path
- **Step 2:** Try to find protection paths for any link failure as described for the dynamic, failure-dependent shared path protection (F-D SPP) algorithm in its *Steps 2-5*.
- **Step 3:** In case of success allocate resources for working and protection paths, go to *Step 1*, otherwise proceed to *Step 4*.
- **Step 4:** Try to find protection paths for any link failure as described for the adaptive shared path protection (SPP-LD) algorithm in its *Steps 2-4*.
- **Step 5:** If succeeded, allocate resources for the new demand, otherwise o_{new} has to be blocked. Go to *Step 1*.

Network	# Nodes	# Links	Average degree	PDSP-LD per demand computational time
NSFNET	13	19	2,92	**0,74**
COST266_CT	16	23	2,88	**2,68**
o22	22	45	4,09	**2,92**
COST266_RT	28	35	2,50	**6,68**
COST266_BT	28	41	2,93	**6,96**
COST266_TT	28	61	4,36	**7,73**

Table 2. Network characteristics

5.2. Semi-Adaptive Partially Disjoint Shared Path Protection (S-A PDSP)

It is a kind of segment protection, as described for PDSP in 3.3. Of course it uses the protection path determining routine of PDSP and PDSP-LD instead of SPP and SPP-LD, otherwise it works as the above described semi-adaptive shared path protection.

6. Numerical Results

We have compared the performance of these algorithms on six well-known topologies consisting of 13-28 nodes and 19-61 edges. The three COST266 reference networks with the same 28 nodes and different sets of edges are of special interest from the aspect of the effect of variable graph connectivity on the performance. *Table 2* shows the characteristics of the networks. The traffic patterns consisted of sufficient amount of different traffic demands to eliminate the effect of the initial transient loading the empty network.

Blocking rates are an important aspect of the performance analysis; therefore at first the results for this topic will be presented. To investigate different blocking ranges we have scaled the link capacities, not the traffic. Note that increasing uniformly the capacity of each link is analogous to decreasing bandwidth of traffic offered to the network. For every network-traffic pair a ten-step simulation sequence was carried out, with capacity values resulting in blocking rates from roughly 90% to around 0%.

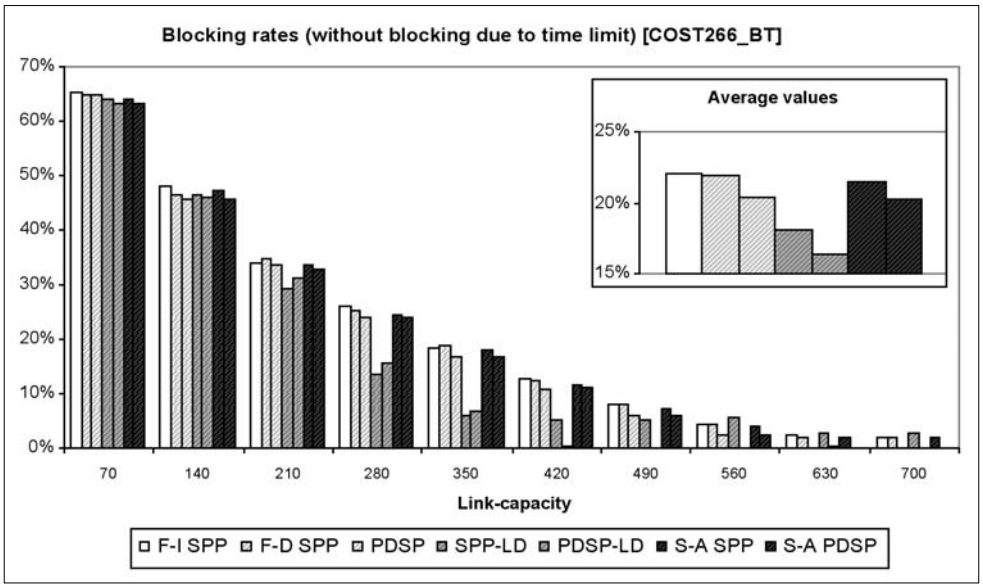
As we have previously mentioned, we face the problem of high complexity and computational time using ILP, and it can distort the results. When processing the incoming demands, the routing method cannot spend a long time to solve any single ILP-problem of a new demand, because it makes the following demands wait for it. Therefore a considerably strict time limit has to be taken into account, and it has a distortion effect: some ILP-problems that are feasible will not be calculated in time, and it means there will be some demands with enough capacity in the network to satisfy them, however due to this time limit, no protection paths will be found for them, and they will be blocked.

That is why we cannot count them as blocked demands, nor as routed ones, because the bandwidth-requirement of them is available, still not allocated in the network. Therefore these demands are simply left out from the statistics. It allows us to examine the potential of the adaptive methods, however in practical applications the limitations of computational time are not negligible, and these demands will count as blocked ones. It is the reason why semi-adaptive algorithms are needed.

The differences between the performances of the described algorithms become more apparent when using larger networks. Accordingly, if we focus on the results of the simulation based on one of our largest networks (COST266_BT – *Figure 3*), the following tendencies are noticeable:

- Failure-dependent SPP provides lower blocking rates than the failure-independent SPP
- The use of so-called “partially disjoint shared protection” results in significantly lower blocking rates than fully-disjoint shared path protection, due to more flexibility in the protection path-building phase (SPP vs. PDSP, SPP-LD vs. PDSP-LD and S-A SPP vs. S-A PDSP)
- Adaptive methods clearly perform better than the dynamic methods because of the protection rearrangement (SPP vs. SPP-LD, PDS vs. PDSP-LD)
- The semi-adaptive methods have lower blocking rates than dynamic versions, but higher than the adaptive algorithms, as expected (SPP < S-A SPP < SPP-LD, PDSP < S-A PDSP < PDSP-LD)

Figure 3. Blocking rates I.



The topology characteristics of network COST266_RT (low graph-connectivity, low average focal degree) make it suitable to demonstrate the weaknesses of path protection. If we take a look at *Figure 4*, and focus on the low-blocking capacity-range, a significant difference is noticeable between the segment and path-protection methods.

Path protection algorithms need two totally disjoint paths for every demand (for the whole source-destination route), and in sparse networks sometimes it is not suitable. At the same time, for segment protection a set of shorter path-duplications to protect different parts of the working path is sufficient. This is the reason why path protection methods cannot reach the 0% blocking rates in a sparse network like COST266_RT.

The adaptive algorithms have further benefits beyond the lower blocking rates. Due to the frequent rearrangement of resources used for protection paths it offers a more even resource usage. The simultaneous protection path establishment for a group of demands results in shorter protection paths, avoiding unnecessary long paths because of bottlenecks. *Figure 5* shows the length of the protection paths (in average): the adaptive algorithms offer clearly the shortest paths, then the semi-adaptive methods, and finally the dynamic versions. Furthermore, the adaptive methods need fewer resources for the protection paths *(Figure 6)*, which means the shorter paths are not the result of less resource sharing but the optimal reconfiguration.

Obviously by improving performance in finding the optimal solution for protection paths the required computational time grows. Its importance is that in some cases it makes the adaptive methods not applicable: for time-critical applications the permanent use of ILP is not acceptable. Furthermore, these algorithms are centralized routing methods, and therefore these cannot be used for large networks due to scalability problems – if not simplified. Table 2 shows the relationship between the network properties and the time needed in average for any single demand using adaptive algorithms: it strongly depends on the number of nodes (ten times longer for COST266_BT then NSFNet with two times more edges and almost the same average nodal degree), and slightly depends on the number of edges (the three COST266 reference networks with different amount of edges need almost the same time).

We do not have to lose all the benefits of adaptive methods neither for time-critical applications or large networks. The semi-adaptive methods could be the optimal choice for these situations: these have the ability to rearrange the protection paths, but for faster operation resource-reconfiguration is made only if a demand cannot be routed without it. Furthermore, these methods are adaptable to different requirements and compromises between speed and performance by altering the prevalence of rearrangement.

The following figures show the required average path computational times (in sec.) required for any single demand in the smallest and largest network tested (NSFNet – *Figure 7*, COST266_TT – *Figure 8*): the semi-adaptive versions remain around 0.01 seconds ever for large networks.

7. Conclusions

In this paper, a set of network protection algorithms have been proposed. A group of dynamic methods offered fast and simple solution, adaptive methods were presented to improve performance, and semi-adaptive versions as a trade-off between speed and performance. Adaptive methods are based on the idea of rearranging protection paths, since protection (back-up) paths normally do not carry any traffic.

As the results show, due to this reconfiguration of protection paths the adaptive methods achieve better performance in the sense of throughput (lower blocking rates), network utilization and even traffic distribution. The drawback of these methods was the complexity of them: for time-critical applications and large networks these are poorly applicable. Therefore semi-adaptive versions were introduced to reduce the average per-demand computational time, and as the results show, these solutions still have some of the benefits of adaptive algorithms.

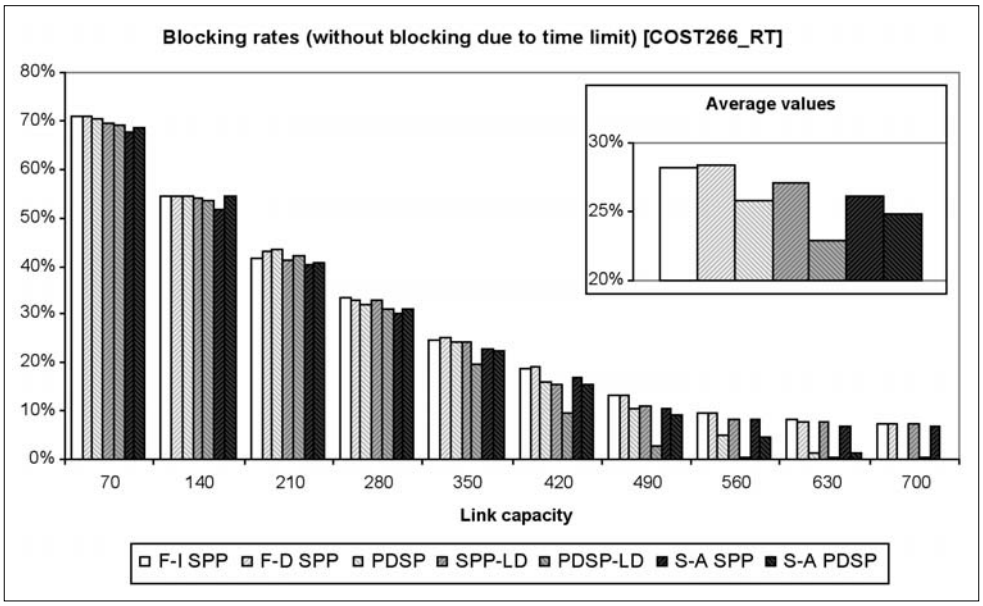


Figure 4.
Blocking rates II.

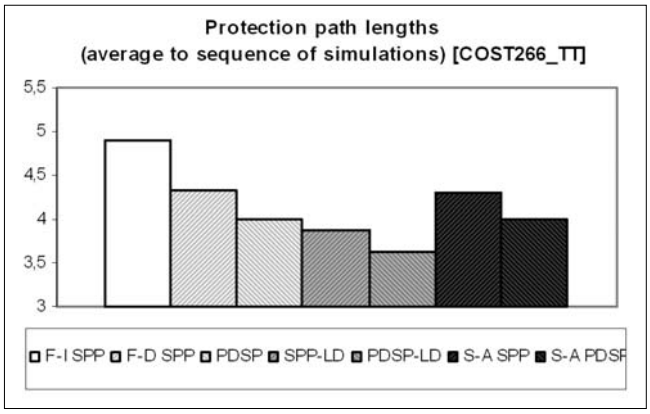


Figure 5. Average length of protection paths

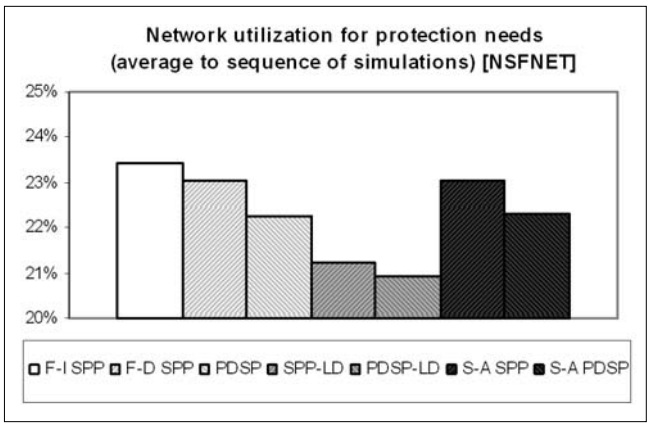


Figure 6. Network utilization for protection needs

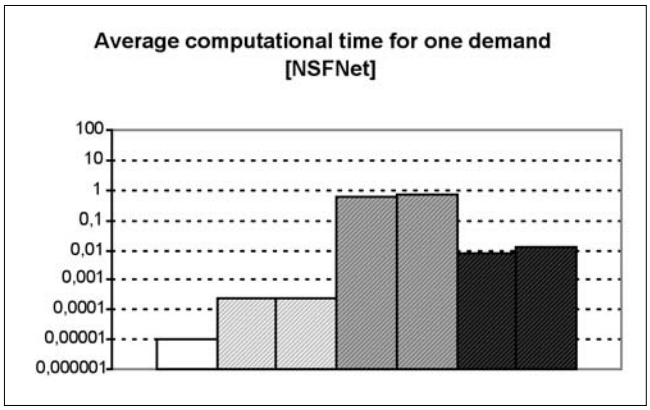


Figure 7. Computational times I.

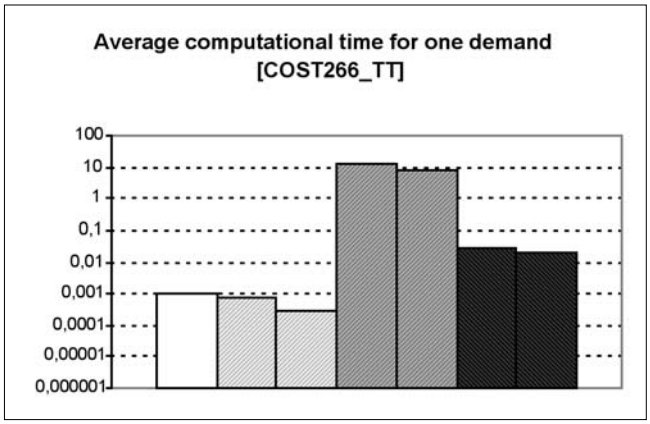


Figure 8. Computational times II.

Acknowledgements

This work has been done as a part of the European 6th Framework Research Project IP NOBEL (http://www.ist-nobel.org)

References

[1] J.W. Suurballe: "Disjoint Paths in a network", Networks, Vol. 4., pp.125–145., 1974.
[2] R. Bhandari: "Survivable networks: algorithms for diverse routing" Kluwer Academic Publishers, ISBN 0-7923-8381-8, 1999.
[3] P.H. Ho, H.T. Mouftah: "A framework for service-guaranteed shared protection in WDM mesh networks", IEEE Communication Magazine, Vol. 40., No.2, pp.97–103., Februar 2002.
[4] T. Cinkler, P. Laborczi, Á. Horváth: "Protection through thrifty configuration", 16th International Teletraffic Congress, Edinburgh, UK, June 1999.
[5] T. Cinkler, D. Meskó, A. Mitcsenkov, G. Viola: "Adaptive Shared Protection Rearrangement", Design of Reliable Communication Networks, Naples, October 2005.
[6] Zs. Pándi, M. Tacca, A. Fumagalli: "A threshold based on-line RWA algorithm with reliabilty guarantees", Conference on Optical Network Design&Modelling, Milan, Italy, 2005.
[7] J.-P.Vasseur, M. Pickavet, Piet Demeester: Network Recovery, Elsevier, pp.39–131, 203–297, 297–423., 2004.
[8] Wayne D. Grover: Mesh-Based Survivable Networks, Prentice Hall, pp.149–172, 173–268, 377–467., 2004.
[9] M.Pióro, D. Medhi: Routing, Flow and Capacity Design in Communication and Computer Networks, Elsevier, 2004.
[10] Internet Software Consortium (ISC), http://www.isc.org/
[11] London Internet Exchange (LINX), http://www.linx.net/
[12] Ravindra K. Ahuja, T. L. Magnanti, J. B. Orlin: Network Flows, Prentice Hall, pp.108–113, 649–695., 1993.
[13] Mokhtar S. Bazaraa, John J. Jarvis, Hanif D. Sherali: Linear Programming and Network Flows, John Wiley and Sons, pp.587–601., 1977.
[14] Schrijver: Theory of Linear and Integer Programming, Wiley, 1998.