

Robust Connections in IP Networks Using Primary and Backup Paths

Nina Hesby¹ Poul E. Heegaard^{1 *} Otto Wittner²

¹Norwegian University of Science
and Technology,
Dept. of Telematics,
Trondheim, Norway

hesby@stud.ntnu.no
poulhe@item.ntnu.no

²Centre for Quantifiable Quality of
Service in Communication Systems.
Norwegian University of Science
and Technology,
Trondheim, Norway[†]

wittner@q2s.ntnu.no

Abstract

Provisioning of differentiated services and Quality of Service guaranties in the Internet has lately changed and increased enormously. With applications requesting strict real-time requirements, virtual connections (VCs) are required and with high dependability requirements multiple VCs should be available. This paper looks at strategies for establishment and maintenance of link disjoint primary and backup paths. The strategies rely on a distributed ant-based routing algorithm called the CE-Ants Algorithm. Three different strategies are tested and evaluated in a dynamic environment.

The results indeed indicate presence of the ability of providing traffic streams with VCs specifying a minimum bandwidth constraint when such paths exist. The results also show abilities of providing continual service and adaptation to changes when single link failures occur.

1 Introduction

The use of the Internet and other Telecommunication Networks has increased and changed enormously over the last few decades. From providing “best effort” services for time insensitive applications like e-mail, the Internet is now facing more challenging tasks. Real time

*This work was also partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).

[†]“Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence” appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT. <http://www.ntnu.no/Q2S/>

applications like Voice Over IP (VoIP) are time sensitive and introduce specific Quality of Service (QoS) requirements which need to be fulfilled by the network. In order to serve these applications, virtual connections (VC) satisfying the requirements must be established.

The virtual connections considered in this paper are established in a dynamic environment. This means that the management of these VCs must be adaptive and robust to changes in the topology (links and/or nodes coming and going) as well changes in traffic congestion. Furthermore, a distributed management service is preferred to avoid the vulnerability of centralized network functions.

In previous work [3], an adaptive, robust, and distributed management service has been developed to establish and maintain link disjoint, high quality primary and backup paths in networks. This service is based on swarm intelligence, and combines the behavior of foraging ants with an optimization technique from rare event theory called the Cross-Entropy method. This service denoted the CE-Ants Algorithm, uses ant-like agents which search for paths in an asynchronous and distributed manner.

In this paper the Multiprotocol Label Switching (MPLS) protocol has been used to realize the link disjoint primary and backup paths found by the CE-Ants Algorithm as virtual connections. Three different strategies concerning how this realization should be performed has been identified and tested.

The rest of this paper is organized in five sections. In Section 2 a brief outline of the principles of CE-Ants is given, followed by a description of MPLS. The strategies are presented in Section 3 and an implementation for testing is described in Section 4. In Section 5 the test results are described and discussed and finally concluding remarks are given in Section 6.

2 Technology

2.1 The CE-Ants Algorithm

The *CE-Ants* Algorithm is inspired by several researchers' earlier work. In [10] Schoonderwoerd et al. introduce the concept of multiple mobile agents cooperatively solving routing problems together and in [9] Rubinstein presents an optimization technique called the Cross Entropy Method.

The mobile agents mimic the foraging behavior of ants, hence no central intelligence is in control. Ants communicate indirectly through the environment by leaving pheromones other ants can sense or "smell". This form of indirect communication is known as the principle of *stigmergy* [10].

When a trail is developed from the nest to the food source by an ant colony, each ant work by the following principle:

- Select a path stochastically - smell pheromones and select a path with a stronger smell with a higher probability.
- If no pheromones are sensed - choose randomly.
- Walk the route to the food source, return and leave pheromones on the way back.

Since pheromones evaporate after some time, recent paths will be favored. Also the shortest paths to the food source will be found first and hence will get more pheromones leading to it *earlier*. This will ensure that the current shortest paths are found.

The concept of using stochastic agents sensing their environment to find paths can be translated into the telecommunication world. Ant-like mobile agents can act as ants, the nodes in a network can be the environment and pheromone trails can be used as routing probabilities in routing tables [11]. By applying a similar set of rules, the ant-like agents can together find shortest paths between nodes.

As stated in the introduction the route management service should be able to find sets of disjoint primary and backup paths. So far systems based solely on swarm intelligence have not concentrated on finding a solution to this problem. Also identification of optimal sets of such pairs is a hard problem where the probability of randomly finding the optimal solution is low, i.e. can be considered a rare event. In order to solve the disjoint path problem and to increase the chances of finding optimal solutions fast, the Cross-Entropy Method has been used.

The Cross-Entropy method uses importance sampling in multiple steps to make desirable rare events more probable. The method can be used to update probabilities in routing tables in a way that emphasize routes with the lowest cost. When the Cross-Entropy method is applied these alterations of probabilities are near optimal. Weighting of path quality is applied to speed up the process. [3] [11].

Rubinstein [9] uses a performance function with a control parameter or temperature γ . This temperature puts a certain emphasis on the shorter routes such that the minimum temperature corresponds to the best routes and highest routing probabilities [3].

The performance function is given as:

$$H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}} \quad (1)$$

This function returns the quality $L(\pi_k)$ of the path π_k . $L(\pi_k)$ is the raw cost of this path. Rubinstein collects these costs over N iterations and calculates the temperature according to

$$\min \gamma_t \text{ s.t. } h(P_t, \gamma_t) = \frac{1}{N} \sum_{k=1}^N H(\pi_k, \gamma_t) > \rho \quad (2)$$

$h(P_t, \gamma_t)$ is the average performance where P_t is the routing probabilities at iteration t . ρ is the search focus parameter and will limit $h(P_t, \gamma_t)$ which will result in a certain amplification of high quality paths. The routing probabilities are also updated. This is done by solving

$$\max_{P_{t+1}} \frac{1}{N} \sum_{k=1}^N H(\pi_k, \gamma_t) \sum_{ij \in \pi_k} \ln P_{t,ij} \quad (3)$$

where $P_{t,ij}$ is the transition probability from node i to j at iteration t . This sequence of information collection and parameter updates are iterated until the performance function $H(\pi_k, \gamma_t)$ does not change anymore and π_k is the best path found.

The Cross-Entropy Method is both centralized and batch oriented which makes it unsuitable for distributed implementation. CE-Ants [3] is the distributed counterpart of the Cross Entropy Method and the temperature and routing probabilities are now updated after each iteration.

The performance function average h_i is calculated autoregressively by

$$h_0 = \beta \cdot h_{-1} + (1 - \beta) \cdot H(\gamma_0, \pi_0) \quad (4)$$

where the last arriving agent is indexed 0, the second last -1 etc. and $\beta \in [0, 1]$ is the autoregressive memory factor. β is typically close to 1 and the larger it is, the more influence older paths have on the temperature. Hence in a highly dynamic network, β should be relatively small (but not too small because then the algorithm could become over-sensitive to traffic fluctuations). For details on the autoregressive schemes derived for updating temperature and routing probabilities see [3]. As a result of the autoregression each agent now performs the following steps:

- When searching for a route, the ant-like agent moves through the network according to routing probabilities and path selection requirements.
- When the destination is reached the path cost is calculated and the performance function is updated autoregressively.
- The agent then backtracks the path updating routing probabilities on each node.

The CE-Ants Algorithm can be used to find near optimal sets of link disjoint primary and backup paths. In order to separate ant-like agents working on the different paths, the concept of species and rank is introduced. All agents working on a specific connection, i.e. finding paths between a certain source and destination node, is said to be of the same *species*. Agents of the same species are divided in to *ranks*, primary and backup being two different ranks.

In order to find disjoint sets of paths, agents of the same species but different ranks should avoid traversing the same links. Since the primary path mainly is the one used by traffic, this should be the optimal path in the network and the backup path should be the optimal without using primary links. When several connections are set up, agents of different species should also avoid each other if link sharing causes congestion. To achieve this, an extra behavioral component, denoted detestation is added to the agents.

Detestation is implemented by altering the cost of a path in order to make certain paths unattractive. The new cost function for the path π_r^m is

$$L(\pi_r^m) = \sum_{ij \in \pi_r^m} \mathcal{D}_{ij}^{mr} \quad (5)$$

where \mathcal{D} represents the estimated traffic loss on the link. The sub- and superscripts gives the rank and species. r represents the rank, m is the species or connection id and ij represents a link.

The full expression of \mathcal{D} is the sum of capacity requested by agents intending to use the link in focus, minus the capacity of the link, hence

$$\mathcal{D}_{ij}^{mr} = \mathcal{S} \left[a_m + \sum_{\forall ns: ij \in \pi_s^n} P_{ij}^{ns} V_i^{ns} Q_{mr}^{ns} a_n - c_{ij} \right] \quad (6)$$

where a_m is the required capacity by the connection in focus, a_n is the required capacity of competing connections, r is the rank of the agents working on the connection in focus and s is the rank of competing connections' agents. The factor V_i^{ns} is the probability that the competing agent ns will visit node i again. The factor P_{ij}^{ns} is the probability that agent ns

will traverse link ij (the departing link for node i in path π_r^m) given that it visits i . c_{ij} is the available capacity on link ij .

Q is set according to the extent that the different competing connections may induce traffic loss. The values of Q are chosen empirically and in this paper we have used the values

$$Q_{mr}^{ns} = \begin{cases} \hat{Q}_{m,r-1}^{n,s-1}, & n \neq m, s = r = 1 \\ 40, & n = m, s < r \\ 5, & n \neq m, s \neq r \\ 1, & otherwise \end{cases} \quad (7)$$

where rank 0 means primary and rank 1 is backup. $\hat{Q}_{m,r-1}^{n,s-1}$ is used in the case where the agents are backup agents of different species. Q is then set to the approximate probability that the primary paths of both species fail at the same time.

If the sum of requested capacity on link ij is less than the available capacity the resulting link cost may become negative. A function $\mathcal{S}[c]$ is applied to smoothen the cost function and to ensure that it never falls below zero:

$$\mathcal{S}[c] = \begin{cases} \eta \cdot \frac{c}{e^{e \cdot \eta}}, & c < \eta \cdot e \\ c, & otherwise \end{cases} \quad (8)$$

where η is a parameter ($\mathcal{S}[0] = \eta$). This cost function is explained more closely in [5].

2.2 Multiprotocol Label Switching (MPLS)

In order to set up the virtual connections found by the CE-Ants Algorithm a method allowing control of the exact path through a network has to be used. We have chosen to use MPLS Label Switched Paths (LSPs) for this task.

In MPLS a set of packets which should be routed the same way through a network is called a Forward Equivalence Class (FEC). Such sets can be traffic from a specific source to a specific destination and will be given a FEC id. A Label Switched Path (LSP) is the predefined path in which all packets of a certain FEC follow through the network [8]. This paper proposes a mapping from VCs to FEC ids and then the VC is set up as a LSP.

A main difference between routing using the MPLS LSPs and traditional routing is the amount of computing done by each router when forwarding packets. A traditional router must do a route lookup based on the address in a packet's IP header to determine the next hop. In a routing domain consisting of all MPLS routers, a route lookup is done by the ingress node to the domain *only*. When the packet enters the domain, the ingress node assigns a label to the packet based on its FEC. This label is used by all the routers within the MPLS network when forwarding the packet. The forward process for these nodes is not a route lookup but a simple reference of the label against a table called the Label Information Base (LIB). The complexity of this process can be compared to a switching operation which is less complex and time consuming than a route lookup. [8].

When creating or altering the LIBs, the predefined LSPs can be set up. In this work we have exploited this full control of the path traversed by traffic by allowing the CE-Ants Algorithm to alter the LIBs. The ant-like agents can now control the traffic routes and ensure robustness and load balancing in the network.

The Label Distribution Protocol (LDP) is a protocol specifically designed to distribute the information needed to alter LIBs in order to set up LSPs. LDP defines a number of control messages exchanged between the nodes in an MPLS network. In addition to messages used for setting up LSPs, this pool of control messages include notification messages sent when an event has occurred in the network. In the context of this paper an event will be a link break or link repair.

To be able to switch traffic from a primary LSP to a backup LSP when a failure occurs, we have utilized a feature called MPLS Path Restoration. This feature is operating in parallel with MPLS Path Protection. The Path Protection monitors links and nodes along protected LSPs and sends LDP notification messages if failures should occur [2]. These features could be used by establishing a primary LSP protected by MPLS Path Protection and bound to a corresponding backup LSP. If the source node should receive a link-failure LDP notification message, traffic can *automatically and immediately* be switched over to the backup LSP. The notification LDP message is sent by the node closest to the failure.

Summarized, the motivation for using MPLS for realizing virtual connections found is:

- Virtual connections can easily be translated into Label Switched Paths and all traffic belonging to the same connection maps naturally into the same FEC.
- The forwarding of a packet belonging to a FEC and bound to a LSP only requires one route lookup since all processing of the packet's IP header is done at the ingress node to the MPLS domain.
- MPLS allows Path Restoration and Path Protection which open up for quick responses to failures.
- MPLS is a widely known and used protocol and adopted by leading producers of router software and hardware.
- MPLS is already implemented in the simulation environment used.

3 Strategies for Setting Up Virtual Connections

The paths found by the CE-Ants Algorithm should be utilized in the best possible way. Traffic is initially routed over the primary path and if a failure on this path occurs, the traffic is shifted to the pre-planned backup path. The algorithm will then initiate a new search for a set of link disjoint primary and backup paths. In this paper we suggest and evaluate three strategies for extracting VCs from the output produced by the algorithm in the most favorable way. In [5] several other challenges and strategy choices are described.

When several ant-like agents cooperate in searching for a path for a specific connection, several possible alternatives will be found. The term *path cost* is the cost of a certain path at a specific point in time. Path *selection probability* is the probability of an agent choosing the specific path at a specific point in time. Path costs and selection probabilities are reported by the agents after each tour. This means that each species and rank has a set of possible paths with corresponding cost and selection probability. Each such set of paths also has a related temperature which is updated by all the agents in cooperation. This amount of information challenges the management service which needs to decide which path to choose for setting up

the VC. If a change in the topology or traffic situations happens, new and better paths may arise. In such cases the management service needs to decide when and if a change the current VC should occur. Generally the strategy for VC selection should:

- Find the path with the best cost possible as early as possible.
- Switch paths as rarely as possible.

Since these properties sometimes can be mutually exclusive, a satisfying strategy must represent a tradeoff between the two. The rest of this section presents the three strategy choices.

3.1 System State Independent

The Check Periodically Strategy One of the simplest and least resource demanding ways of identifying the best path found by the ant-like agents, is to simply check periodically which is the current best path. The “best” path is here defined as the path with the currently highest selection probability. This method will not take the system state into account and is therefore not necessarily satisfying the criteria of finding the best path as early as possible. However it guaranties maximum one path switch for each interval e.g. if the management function checks for new best paths every second, maximum one path change per second may occur. This strategy is denoted the *Check Periodically* Strategy.

3.2 System State Dependent

To identify the best path as quickly as possible, the state of the system should be considered. Relevant state information must be extracted by analyzing the outputs of the CE-Ants Algorithm. One way of identifying stability is to monitor the temperature. This can be done by calculating the variance of a window of observations. When stability is reached the variance of the temperature will be low. A stable temperature usually indicates convergence to the best path. If several equally good best paths exist, the temperature can stabilize without converging to one specific path. Nevertheless the system has found one or several near optimal paths, and should set up a VC.

When the system is stable, the best path should be chosen and kept. When the system is unstable there is a question of how often a path change should occur when a new and better path is found.

The Check When Crossing Limit Strategy This strategy checks for better paths when the temperature variance falls below (or crosses) a certain limit. If a new path is better than the current operating LSP, a VC change occurs and traffic is switched to the new LSP. If a change happens causing the temperature variance to rise above the limit the management function waits until stability is reached again before a new path is set up letting the backup path deal with traffic in the meantime.

This strategy causes changes to happen soon after the system has reached stability. The check is performed only one time for each unstable period avoiding an unnecessary amount of path changes. It is however important to choose a suitable limit for indicating temperature variance stability in order to identify true stable and unstable periods. In order to minimize possible harms caused by this parameter sensitivity, the strategy has been simulated together

with the *Check Periodically* strategy.

The Check When Above Limit Strategy This strategy monitors temperature variance and defines stable state in the same way as the *Check When Crossing Limit* Strategy. When the system is in an unstable state this strategy switches paths *each time* a better path than the operating is found. When the system is stable, no changes are triggered.

This strategy has the advantage of reacting as fast as possible to system changes, allowing the traffic to use the best path possible at any given time. The amount of path changes can however be large. The strategy has the same sensitivity issues to the temperature variance limit as the *Check When Crossing Limit* strategy and has therefore also simulated together with the *Check Periodically* strategy.

4 Implementation

The CE-Ants Algorithm involves a large number of stochastic processes working in parallel. The outcome is therefore difficult to analyze analytically. In order to demonstrate the behavior of the algorithm and performance of the strategies the algorithm has been simulated in the open source simulator package Network Simulator 2 (NS2) [6].

The NS2 package is well tested and supports necessary functionality like dynamic topologies. The package is implemented as a mix of C++ classes in a kernel performing all heavy computations and OTcl classes which acts as the interface between the user and the kernel. In [3] CE-Ants has been implemented in a version of NS2 supporting Active Networking (AN) [7]. AN allows code carried by the ant-like agents to be executed on the routers with the purpose of altering their routing tables.

In this work we have used a NS2 extension supporting MPLS [1]. This extension has been utilized by allowing the mobile agents to trigger events in the MPLS extension.

In addition to the behavior of the CE-Ants Algorithm some other functionality has been added to the simulator. In previous work [4] *Explorer Agents* have been introduced to the simulator. These are agents performing searches without using routing probabilities but randomly choosing their next hop. The purpose of these agents is to discover new paths appearing after convergence has occurred. The ratio of Explorer agents to regular agents must be set properly in order to not disturb the convergence process too much. In earlier work [4] 1:10 has been empirically chosen. We have chosen to use the same ratio.

In order to test the strategies presented in the previous section, the temperature variance has to be calculated for each new temperature reported by the agents. The variance should reflect temperatures from a certain window of observations. In our implementation the variance over n observations is calculated at the source node according to (X_i is the i th last temperature)

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n X_i^2 - \frac{n}{n-1} \hat{X}^2 \quad (9)$$

and then normalized by S^2/\hat{X} . n is the size of the window of observations. The implementation is more thoroughly described in [5].

5 Simulation Results

We have designed a dynamic scenario for demonstrating the algorithm’s adaptation abilities and the performance of the strategies. The topology consists of 10 nodes with different capacity links and is divided into 4 periods by events like link failures and repairs. The parameters used in our simulations are: $\beta = 0.950$, $\rho = 0.01$, $\eta = 2e6$, $\text{var_window} = 20$, $\text{var_limit} = 0.001$ and $\text{periodic_check} = 1.0$.

Figure 1 shows the different periods of the scenario indicating the best available path for each period. Traffic is set up between the source node 2 and the destination node 3. The traffic is generated by a Constant Bit Rate (CBR) source which offers 80 Mb/s throughout the simulation.

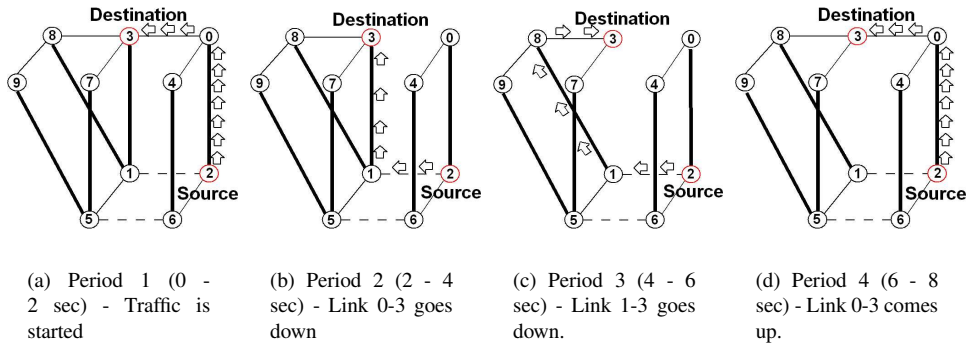


Figure 1: The different periods and events in the scenario. The arrows indicate the best available path in each period. The link capacities are: 140 Mb/s for the thick links, 100 Mb/s for regular links and respectively 60 Mb/s (for link 2-1) and 30 Mb/s (for link 6-5) for the dashed links.

Each of the three strategies *Check Periodically*, *Check When Crossing Limit* and *Check When Above Limit* are simulated 10 times in this topology. In addition a run where the best available paths shown in Figure 1 is used by manually setting up MPLS LSPs has been made for comparison purposes. This run is called the *Perfect Strategy* and is of course not feasible in real networks.

The average received throughput at the destination node for these four cases is shown in Figure 2. The maximum and minimum observations are given as errorbars for each point in the graph.

Plot (a) in Figure 2 shows the received bandwidth in the perfect case. Between time 2.0 and 6.0 (Periods 2 and 3) there is no path with sufficient capacity in the network and traffic has to traverse link 2-1 forcing received bandwidth to be ≈ 60 Mb/s. The “teeth” on the line are due to packet drops at certain intervals. At time 4.0 a failure happens causing a drop in received bandwidth. This drop, seen as a spike in the graph, is due to full buffers on the low capacity link 2-1. Packets sent after the failure will traverse the backup path while the packets in this buffer will be sent on the old link and dropped. At time 6.0 a path with enough

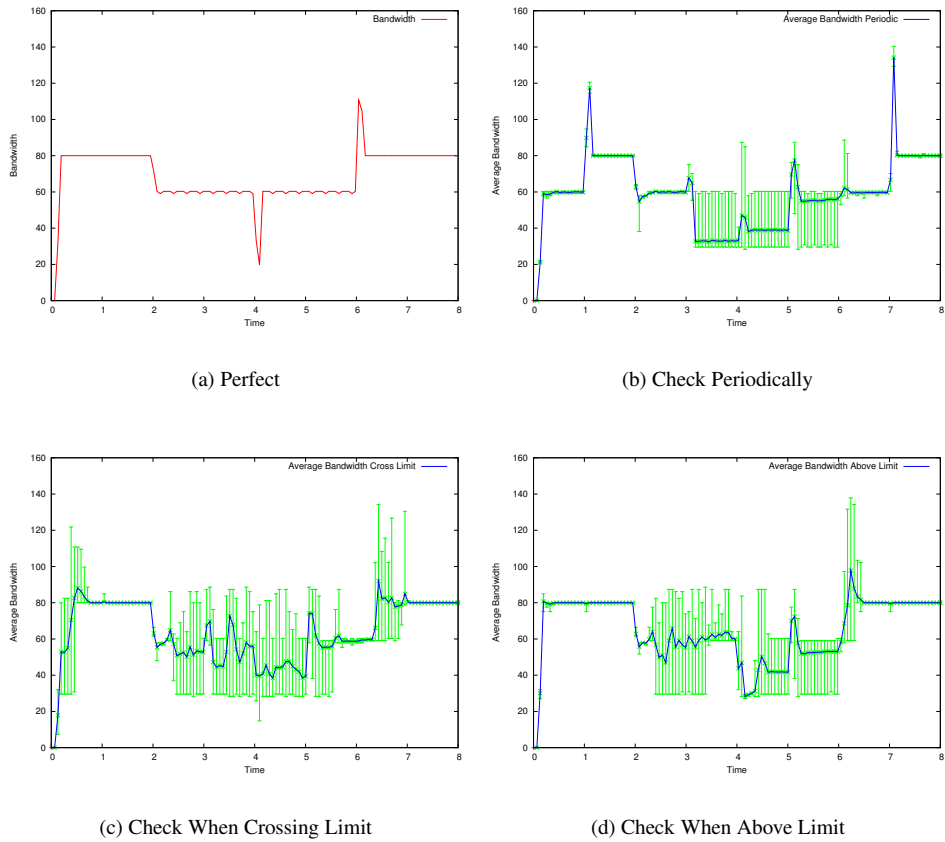


Figure 2: The average bandwidths are plotted for the different strategies together with errorbars showing the maximum and minimum observations. Offered traffic is 80 Mb/s at a constant rate.

capacity appears and traffic is switched to the new path. At this point the destination node experiences an increased bandwidth due to packets from the full buffers along the old path still arriving. Even if an increased bandwidth is experienced, the packets may arrive out of order being useless for some applications.

In Plots (b), (c) and (d) we see the graphs for the three tested strategies. Generally it can be seen that all strategies are able to provide a connection to the traffic stream even when failures happen. For the Periods 1 and 4 we see that the *Check When Above Limit* Strategy produce results most similar to the *Perfect* Strategy. The *Check Periodically* Strategy has a somewhat delayed reaction to topology changes. This is due to the strategy's sensitivity of the timing of the periodic checkpoints. The *Check When Crossing Limit* Strategy has a slightly longer reaction time than *Check When Above Limit*. In Periods 2 and 3 the ant-like agents experience an overloaded system with no solution but will still try to find the best path possible. In these periods the agents have to utilize one of the two insufficient links in the topology. Since traffic

will cause congestion on the chosen link, agents will not be able to traverse this link and hence switch to the other. This procedure repeats itself causing an oscillating effect between paths using these links. Since one of the links has a capacity of 60 Mb/s and the other has 30 Mb/s this can be seen in the plot for the *Check Periodically* Strategy as separate periods of about 60 and 30 Mb/s received bandwidth. For the two other strategies changes between paths will happen much faster which can be seen in the figure. At time 4.0 link 1-3 goes down. This link is part of the best path at this point but is not necessarily utilized by the strategies due to the oscillating effect seen around this time. Since the *Check When Above Limit* Strategy is more likely to use the best path due to more rapid updates this strategy is more likely to experience this failure. This can be seen as a larger average bandwidth drop for this strategy right after time 4.0.

Below the average received bandwidth and average number of path (LSP) changes can be seen for the different periods and strategies. The columns “BW” is the time average received bandwidth and “LSP” is the average number of LSP switches. The best observed values for each period are bolded.

| | Period 1 | | Period 2 | | Period 3 | | Period 4 | |
|-------------|--------------|------------|--------------|------------|--------------|------------|--------------|------------|
| Strategy | BW | LSP | BW | LSP | BW | LSP | BW | LSP |
| Perfect | 77.73 | 0.0 | 60.34 | 1.0 | 57.85 | 1.0 | 81.78 | 1.0 |
| Periodic | 68.10 | 1.0 | 47.01 | 0.9 | 48.65 | 1.2 | 71.54 | 1.2 |
| Cross Limit | 73.01 | 2.1 | 55.53 | 2.7 | 51.20 | 2.1 | 76.87 | 1.0 |
| Above Limit | 75.63 | 2.5 | 58.90 | 2.8 | 47.53 | 2.2 | 79.86 | 1.6 |

Average packets dropped and arriving out of order is shown below. In Period 1 1900 packets are sent and 2000 are sent the other three periods. The columns “Lost” is average number of packets lost and “Order” is average number of packets arriving out of order.

| | Period 1 | | Period 2 | | Period 3 | | Period 4 | |
|-------------|------------|------------|--------------|-------------|--------------|-------------|-------------|-------------|
| Strategy | Lost | Order | Lost | Order | Lost | Order | Lost | Order |
| Perfect | 0.0 | 0.0 | 434.0 | 0.0 | 481.0 | 0.0 | 69.0 | 0.0 |
| Periodic | 184.1 | 47.9 | 660.8 | 38.1 | 834.7 | 54.2 | 270.2 | 53.0 |
| Cross Limit | 66.0 | 31.3 | 538.2 | 124.1 | 682.3 | 86.2 | 125.6 | 47.2 |
| Above Limit | 0.0 | 1.1 | 463.3 | 116.7 | 752.6 | 72.5 | 66.2 | 53.0 |

In all periods except Period 3 (which is a special case explained above) the *Check When Above Limit* Strategy (marked “Above Limit”) achieve the highest throughput and less packet loss. This strategy does however also generate more LSP changes. The *Check Periodically* Strategy (marked “Periodic”) gets the opposite result while the *Check When Crossing Limit* Strategy (marked “Cross Limit”) falls in between. More thorough descriptions of these results along with simulations performed in a larger scenario can be found in [5].

6 Concluding Remarks

In this work we have investigated the use of link disjoint primary and backup paths in order to increase the robustness of routing services. The paths are set up and maintained by strategies using output from the CE-Ants Algorithm. CE-Ants’ good ability of finding link disjoint primary and backup paths which provide traffic with a connection even when failures happen is demonstrated for all the strategies.

The *Check When Above Limit* Strategy provides traffic with the highest bandwidth connection but also generate more path changes and hence more unsequenced packets. This strategy should therefore be used for applications insensitive of unsequenced packets. The *Check Periodically* Strategy is the easiest to implement and require no storage of temperatures or calculation variances. It also requires fewer path changes making it suitable in situations where such changes are very costly. The *Check When Crossing Limit* Strategy achieve received bandwidths close to the *Check When Above Limit* Strategy but without the same amount of path switches. This strategy can be suitable for applications sensitive of unsequenced packets but also requiring short delays.

In [5] a scenario with two connections has been simulated. Results show that the management service is capable of avoiding congestion between traffic of different connections.

Future work includes testing of the strategies in larger scenarios with many connections. Different strategies for handling the situation after a link failure occurs and traffic is switched to the backup path should also be tested.

References

- [1] Gaeil Ahn. MPLS Network Simulator(MNS). <http://flower.ce.cnu.ac.kr/~fog1/mns/>. Visited June 21, 2004.
- [2] Gaeil Ahn and Woojik Chun. Simulator for MPLS Path Restoration and Performance Evaluation. Work at Department of Computer Engineering, Chungnam National University http://flower.ce.cnu.ac.kr/~fog1/mns/mns2.0/doc/MNS_v2.0_path_restoration.pdf. Visited June 21, 2004.
- [3] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*, Montreal, Canada, August 14-16th 2001. Springer Verlag.
- [4] Nina Hesby. Case Studies of Primary and Backup Path Management in Networks. Project work at the Department of Telematics, Norwegian University of Science and Technology, November 2003.
- [5] Nina Hesby. Robust Connections in IP Networks Using Primary and Backup Paths. Master's thesis, Department of Telematics, Norwegian University of Science and Technology, June 2004.
- [6] DARPA: VINT Project. UCB/LBNL/VINT Network Simulator - ns (version 2). <http://www.isi.edu/nsnam/ns/>. Visited June 21, 2004.
- [7] Konstantinos Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, First Quarter 1999.
- [8] E. Rosen, A. Viswanathan, and R. Callon. RFC3031: Multiprotocol Label Switching Architecture. IEFT, January 2001.
- [9] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.
- [10] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-Based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.
- [11] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.