NTNU
Norwegian University of Science and Technology

Study of TCP friendliness of CEAS routing system in comparison with Distance Vector Routing and Link State Routing

Sandeep Tamrakar

Master in Security and Mobile Computing

Submission date:	June 2009
Supervisor:	Bjarne Emil Helvik, ITEM
Co-supervisor:	Otto J Wittner, ITEM
	Sasu Tarkoma, External (TKK)

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Ensuring that a network treats TCP traffic in a "friendly" manner has been an important topic for at least a decade. A promising stochastic path management (routing) system known as CEAS (Cross Entropy Ant System) has been developed at the department and Q2S. An important issue is the level of "TCP friendliness" CEAS may provide. To investigate this and related questions, it is suggested to perform comparative studies of the performance of CEAS based and distant vector based routing.

Work toward this objective ha started as an autumn project, using ns-2 as a tool. The master thesis is
a continuation of this work, where it will be look at more complex network scenarios. For instance:
- various topologies.
- dependence on the traffic source characteristics.
- multiple TCP streams and background traffic.
- a range of failure characteristics, including link failures statistic observed in operational networks

An expected outcome of the study is a mapping of the pros and cons of CEAS routed network relative to distant vector routing for this kind of transport.

Assignment given: 15. January 2009
Supervisor: Bjarne Emil Helvik, ITEM

Abstract

With the continuous development of the Internet technologies new routing requirements have surfaced. In response, several adaptive, stochastic routing algorithms have been purposed. The Cross Entropy Ant System (CEAS) is an adaptive, robust and distributed routing and management system based on the swarm intelligence. Several prototype implementations and enhancements have been made on this system, however the level of TCP friendliness the CEAS may provide is yet an important issue.

In order to investigate the level of TCP friendliness, the behavior of the CEAS system during different network dynamics needs to be understood. For this reason, the behavior of the CEAS system under different network event and its corresponding effects on TCP performance is examined first using a simple network. Later the level of TCP performance is measured on complex networks. Also the load sharing capabilities of the CEAS system is investigated the efficiency of the system to manage and update according to the network load. Additionally the results are compared against the results obtained from the standard Link State Routing protocol and the Distance Vector Routing protocol under similar conditions.

In this work, we find that the update process in response to the change in network dynamics is slower on CEAS compared to the other systems. However, the update process speeds up with the increase in the ant rates. During such period the use of multiple path reduces the TCP performance. We also find that large amount of packets loop around some links during link failures. Such looping reduces the TCP performance significantly. However, implementing previous hop memory technique removes such loops and also help TCP resume transmission immediately after the link failure.

Compare to the LSRP and the DVR, we find that CEAS manages network resources more efficiently to produce higher TCP performance. We find that the CEAS diverts the data traffic on the basis of the quality of the path rather than the length of the path. We also find that the CEAS system handles multiple TCP stream independently with equal priority. But the smaller transition delay on the ants compared to the data packet reduces the TCP performance to some extent. However, forcing the ants to experience longer queuing delay according to the traffic load improves the TCP performance as well as helps CEAS update more accurately.

Acknowledgements

This text is submitted as the concluding part of my Master of Science degree in Security and Mobile Computing in the NordSecMob program. This work has been carried out at the Department of Telematics (ITEM), Norwegian University of Science and Technology (NTNU) during the spring of 2009.

I would like to thank my supervisor Professor Bjarne E. Helvik and my tutor Otto J. Wittner, Post Doc at the Center of Quantifiable Quality of Service in Communication System, Centre of Excellence (Q2S), for all their guidance, assistance and valuable feedbacks throughout the period of this thesis. Additionally, I would like to thank Professor Sasu Tarkoma at the Helsinki University of Technology (TKK) in Finland for his supervision. Special thanks to Laurent Paquereau, Phd at Q2S for providing the required materials for this thesis, his guidance and assistance.

Sandeep Tamrakar

Norwegian University of Science and Technology (NTNU), Trondheim

June 2009

Abbreviations and Acronyms

ABC	Ant Based Control
ACK	Acknowledgement
AIMD	Additive Increase and Multiplicative Decrease
CBR	Constant Bit Rate
CE	Cross Entropy
CEAS	Cross Entropy Ants System
cwnd	congestion window
DUPACK	Duplicate Acknowledgement
DVR	Distance Vector Routing
FIB	Forwarding Information Base
Gbps	Giga bits per second
IGP	Interior Gateway Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
Kbps	Kilo bits per second
LSA	Link State Advertisement
LSRP	Link State Routing Protocol
Mbps	Mega bits per second
MSS	Maximum Segment Size
NS 2	Network Simulator 2
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
RTO	Retransmission Time Out
RTT	Round Trip Time

rwnd	receiver window
SACK	Selective Acknowledgement
SMSS	Sender Maximum Segment Size
ssthresh	slow-start threshold
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction and Motivation

With the continuous development of the Internet technologies, large amount of new applications based on the Internet is being evolved. As a consequence, the volume of data traffic is growing enormously. Such huge traffic calls for new routing requirements those that are scalable, self adaptive and self manageable. In other words, we require routing algorithms that detect and utilize its available network resources, distribute data traffic across multiple paths and quickly adapt to the changing network loads and the network topologies to produce low latency, low loss and high throughput. A study by Tangmunarunkit et al. [35] shows that the current IP routing does not make good utilization of its available network resources in order to provide high throughput. For example, in a network there may exist some path other than the routing policy defined path that has more number of intermediary nodes and yet produces better performance than the predefined path.

In response to these requirements, several adaptive stochastic algorithms have been purposed that make use of multi-path routing. Further, self-organized, distributed algorithms inspired by the native behavior of ants have been studied for a while. Such systems are referred to as *Swarm Intelligence*[1]. Algorithms based on ant colony opti-

[1]Swarm Intelligence is a decentralized, self-organized systems that results in an optimal solution by collective behavior

mizations have been applied to solve various combinatorial optimization problems such as traveling salesman problem, quadratic assignment, protein folding, graph coloring etc. Such algorithms have been proposed for network routing as well. AntNet [7], Ant-based Control [32], Adaptive Swarm-based Routing [36] etc are some of the examples of ant based routings. In these algorithms the swarming behavior of ants are represented by mobile agents that flow throughout the network and collect information that are later used for managing and controlling the network.

Typically ant based routing are studied with focus on datagram network i.e. User Datagram Protocol (UDP) as a transport layer protocol [7, 10]. However, most of the applications on the Internet use Transmission Control Protocol (TCP) as the transport layer protocol for reliability. The nature of adaptive routing is that it makes use of different available paths to transmit data of the same session. At each node the probability of selecting next node is based on the quality of the path between them. At the destination, the data following different paths may reach out of order. UDP responds the out-of-order packets simply by discarding them. However, beyond certain threshold, TCP assumes such out-of-order delivery is due to the network congestion and responds by decreasing TCP throughput. Therefore, the behavior of the TCP on a network using ant based routing should be well studied.

A promising stochastic path management system, based on the swarming behavior of ants and the rare event optimization method cross entropy, know as Cross Entropy Ant System (CEAS) has been developed at the department of Telematics, NTNU and Q2S. Several prototype implementations and enhancements have been made on the system, however the level of TCP friendliness CEAS may provide is yet an important issue.

The main objective of this work is to understand the behavior of the CEAS during different network events and its corresponding effects on the TCP performance. The time taken by the CEAS to update the system in response to the change in the network dynamics is an important factor. This helps us to understand how well the TCP

of simple local interactions between elements of the system.

performs during such events. Thus a part of the work includes measurement of the CEAS update process and the behavior of the TCP during such period. The other important issues are to find out the performance of TCP during network congestion, the load sharing capabilities of the CEAS system etc. One of the main objective of this study is also to compare the results against other standard routing systems such as Link State Routing and Distance Vector Routing under similar conditions.

1.2 Related Works

Ant Based Control (ABC) by Schoonderwoerd et al. [32] was the first attempt to implement an ant based optimization method to the network routing. However this is limited to the symmetric circuit switched telecommunication network and is not applicable to the packet switching network like the Internet. Dorigo and Di Caro [7] later introduced AntNet which is designed for packet switching, connectionless network. The Cross Entropy Ants System, purposed by Helvik and Wittner [16], forms the groundwork for this work.

Godomska and Pacut in [10] have studied the behavior of TCP on ant based routing like AntNet and Adaptive Swarm-based Routing. They concluded that, although TCP sets higher demands on the adaptation process, the load range of the network could be extended in a way to provide efficient routing policies. However, their study was based on the comparison of TCP performance against UDP.

In Master's project [33], we compared and analyzed the TCP performance in a network applying CEAS based stochastic routing against Link State Routing OSPF. The TCP performance were measured using factors like Connect time, throughput and goodputs. The study was based on a small network with only one TCP connection with various background traffic loads. During the project we found that a large amount of TCP packets were lost due to looping after link failures. Such looping significantly reduced the TCP performance. In this work, we try to avoid such looping by implementing previous hop memory technique that forbids forwarding packets back to the node that recently forwarded.

Further, the project work was carried out only using one TCP connection on a network. In this work, we examine the behavior of the CEAS system during different network events and measure the TCP performance. We also examine the performance of multiple TCP sources using two different variants of TCP.

1.3 Research Methods

In order to evaluate the performance of TCP on CEAS system, the following main questions are investigated.

1. How fast the CEAS system updates and stabilizes in response to the network dynamics and how they affect the TCP performance?
2. Does the use of multipath routing causes reduction in the TCP performance?
3. How well the system handles TCP data traffic during congestion?
4. Does the system re-routes the TCP traffic in response to the congestion or they are continuously forwarded along the congested path?
5. How does the CEAS handles multiple TCP traffic when they share common links?
6. Does the CEAS system divert multiple TCP traffic on separate path or the TCP connections struggle to use the best path?
7. Does the TCP performance on CEAS is better than on the LSRP and DVR?

Each of the questions is investigated using simulations. The simulations are carried out in a simple network. The simple network helps us understand and investigate the questions in an easier way. Further, it is easier to visually examine the simulations to gain more insight.

All the simulations in this work have been carried out using Network Simulator 2 (NS 2) [2]. NS 2 is a widely used discrete event simulator that provides a reproducible and

[2]For more information on NS2, see http://www.isi.edu/nsnam/ns

controllable environment for the evaluation of the Internet protocols. All the previous studies related to the CEAS have been based on the modules developed by Helvik and Wittner [16]. However, for this work a newer version of the CEAS module has been provided by the department of telematics, NTNU. This new version is developed on top of the multi- * network extension by Paquereau and Helvik [26, 27]. The modules are written in C ++ and some modifications have been made according to our requirements. 15 independent replications have been simulated for each scenario, the results have been then synchronized to calculate 95 percent confidence interval. Chapter 3 describes the simulation model in detail as well as the modifications made.

The results related to the TCP performance as well as CEAS behavior are studied. The results related to CEAS behavior during different network events are used to understand the behavior of the TCP. If TCP performance shows unexpected results, the CEAS behavior is then studied to find logical explanations. Further, the simulations are visually examined to gain more insight during such unexpected results. The TCP performance results are then compared with the results obtained from LSRP and DVR under similar configurations.

Later, two case studies are made to measure the performance of TCP in complex CEAS networks. Each case study is based on different topologies. The results from our basic studies are used as references to reason out the behavior of TCP in the complex networks.

1.4 Structure of thesis

In this chapter, we briefly outlined the research area and our approach. The rest of the thesis is organized as follows.

Chapter 2 provides the necessary background on stochastic routing, LSRP and DVR. It also provides brief overview of CEAS system. Section 2.6, gives a brief overview of TCP and the underlying congestion control algorithms. Further, Section 2.7 defines the problems related to TCP on stochastic networks. Finally, in Section 2.8 we explain the congestion control algorithm used in different versions of TCP

In Chapter 3, we outline our measurement platform and simulation models. The modification made on extension is explained in Section 3.2. The rest of the sections defines the implementation, parameters, topologies, network dynamics used in our study.

In Chapter 4, we measure the TCP performance on CEAS system under influence of different network events. The sections are organized according to the network events. The results from each events are explained using graphs. The results are also compared with the results from LSRP and DVR.

Chapter 5, presents two case studies based on two different networks. The studies in this chapter mainly focus on observing the behavior of the TCP in complex networks. The results from Chapter 4 are used as references to reason out the behavior of the TCP in complex networks. The chapter is divided into two sections based on the network topologies; a 12 node network and the 58 node topology from the Uninett network.

The main results of this work are summarized in Chapter 6 and the future works are outlined.

Chapter 2

Background

2.1 Stochastic Routing

Basically IP routing is a set of protocols responsible for determining the path that data follows from its source to the destination. The path from a source to its destination may consist of series of routers. The IP routing protocol helps routers maintain a set of rules that they refer while forwarding data packets to the next node towards the destination [34].

The state of art Link State Routing protocols such as OSPF[1] and IS-IS[2] are deterministic. Such routing algorithms are fine tuned in order to pre-determine the best path from source to the destination. Each router along the path maintains a next-hop routing table that consists of the addresses of its neighboring nodes along with the associated path cost. The selection of the next hop is based on the cost associated with the path towards the destination. Similarly Distance Vector Routing such as RIP [3] uses hop count as routing metric to find the best path between a pair of nodes. Therefore, the data packets are always routed through the same path defined by the routing policy even though there may exists multiple paths towards the destination

[1]Open Shortest Path First (OSPF) is an Internet domain routing based on Dijkstra's algorithm. See IETF RFC 2328.

[2]Intermediate System to Intermediate System (IS-IS) is similar to OSPF based on Dijkstra's algorithm. See IETF RFC 1195.

[3]Routing Information Protocol (RIP) is a distance-vector routing that uses hop count as routing metric. See IETF RFC 1058

[28].

However, in stochastic routing, the routing tables maintained by each router consist of possible next node addresses based on a probability distribution. During packet forwarding each node randomly selects next node from its routing table irrespective of the previous selections. However, the probability of selecting a particular node among other possible nodes is proportional to the quality of the path between them. In stochastic routing there are no pre-determined path defined by the routing policy. Thus, the data packets do not always follow the same path which makes this routing non-deterministic [28, 6]. However, swarm based routing algorithms use stochastic optimization methods in order to minimize the time finding the optimized path [22, 15].

2.2 Ant Routing

A collection of numerous simple local interactions between elements of self-organizing system that results in complex collective behavior is known as *Emergent behavior*. Emergent behaviors are very useful to solve optimization problems [11, 18]. Such behavior can be found in nature, for example; a colony of ants sorting eggs without having ants the knowledge of sorting. In this example the optimization process is not centrally controlled but in fact the solution is produced by the collective behavior of individual elements of the distributed system.

Swarm based routing algorithms make use of the emergent behavior of Ants searching for food. In nature, ants continuously forge around in search of food. They form a self-organizing system by interacting with each other using a chemical signal called *pheromones*. During food search ants leave behind pheromones on their way towards the food. These pheromones direct other ants towards the food. Paths with the stronger pheromones are likely to be followed by most of the ants. As time passes by, the pheromone gradually evaporates, yet the shorter paths tend to have stronger pheromones than the longer paths. Thus, the shortest path is more likely to be followed by further ants. The resulting shortest path is the outcome of simple local interaction

[18, 11, 7, 15].

2.3 Cross Entropy Ant System

The Cross Entropy Ants System (CEAS), purposed by Helvik and Wittner [16], forms the basic foundation of this work. The CEAS routing is based on two key principles; *emergent behavior* and the *cross entropy method* for stochastic optimization.

In CEAS, emergence can be explained by the behavior of numerous ant-like mobile agents that flows randomly throughout the network in search of possible paths between two nodes. When a path from a source to its destination is found the ants retreats back following the same path as well as updating a parameter, denoted as pheromone. This pheromone is a crucial element in path finding as it guides other ants traveling towards the destination. Each nodes along the path maintains a table that incorporates addresses of its neighboring nodes along with their corresponding pheromone values towards them [16, 15].

The CEAS uses a Cross Entropy (CE) method introduced by Rubinstein [30] to update pheromones. The CE method has been widely used as powerful technique for solving combinatorial optimization problems. The CE method uses adaptive sampling technique to probabilistically converge random sequence of solutions to an optimal solution. Thus, it is very helpful in finding optimal solutions in case of rare events where the probability of occurring an event is very low. For example, on a large network, the probability of finding an optimal path between two nodes by searching randomly throughout the network is very low. Hence the use of adaptive sampling helps gradually and iteratively minimize the cross-entropy between randomly found paths and converge various found paths into an optimized path on the basis of the path cost [30, 16, 15].

In CEAS system, the source node is responsible for generating simple agents denoted as ants. All ants are initially generated as *forward ants* that explore the entire network searching paths towards the destination. The forward ants are of two types:

explorer ants and *normal ants*. At each intermediate node, the explorer ants choose the next node randomly. These explorer ants are responsible for finding new paths. Unlike explorer ants, normal ants choose the next node according to the probability distribution i.e. the probability $p_{ij,t}$ for a normal ant at visit t in node i, is calculated according the random proportional rule 2.1. Each node maintains a database known as message database to store the outcome of the random proportional rule [15].

$$P_{ij,t} = \frac{\tau_{ij,t}.I(j \notin U)}{\sum_{(i,l)\notin E, l \notin U} \tau_{il,t}} \tag{2.1}$$

where, $\tau_{ij,t}$ is the pheromone value of link $(i,j) \in E$ at update t, U is a list of forbidden nodes, and E is the set of all links.

When a forward ant reaches its destination, a path cost, denoted by $L(\omega)$, is calculated as in 2.2 . The path cost $L(\omega)$ is the summation of all the link costs along the path that ant followed. This path cost may vary with traffic loads and time.

$$L(\omega) = \sum_{\forall (i,j) \in \omega} L((i,j)) \tag{2.2}$$

where, $L((i,j))$ is a link cost between nodes i and j.

In addition to this, a control variable, know as temperature, is also updated. Over time the temperature variable decreases but as more ants arrive at the destination, the temperature variable stabilizes.

Backward ants then travel back from the destination node to the source node along the same path but in reverse order. At each node along the path these backward ants update the pheromone values. The pheromone values are updated according to the path cost and the temperature variable. Thus, over time the pheromone values along the better path get stronger there by increasing the probability of normal ants passing through the path (from equation 2.1). As more ants pass through the better paths, the routing table converges to find an optimal path among multiple paths [16, 15].

2.4 Link State Routing (OSPF)

Open Shortest Path First (OSPF) is a link-state routing protocol and is a family of Interior Gateway Protocol (IGP). As a link-state routing protocol, routers discover their neighbors and their states by exchanging link-state messages known as Link State Advertisements. Initially LSA messages are flooded throughout the network (*usually Autonomous System* [4]) to discover neighboring nodes. Once the initialization phase is completed the LSA messages are exchanged periodically or in response to any change in the topology. LSA messages helps nodes maintain a link state database regarding their local and neighbor's information, topological information and the cost associated with each link towards the neighbors. Upon receiving an LSA message each node compares the message with the entry in their database and updates them provided that the LSA message is newer. Thus, every node within the Autonomous System has information about the entire network topology which they use to calculate end-to-end path cost using Shortest Path First or Dijkstra's algorithm. During routing, the next node along the path is the one with the lowest path cost towards the destination [23, 34].

The complete knowledge of the entire network topology allows the OSPF nodes to easily calculate the shortest path from a source node to its destination node. In addition to this, it also allows the system to quickly respond to any change in the topologies such as link failures and link re-establishments. However, link state routing protocols are not scalable; increasing the number of nodes in the network increases the volume of the LSA messages exchange as well as increases the time require to calculate entire end-to-end path. Additionally, the protocol is not suitable in the environment where the frequency of link failures is very high. At such high rate of link failures the amount of LSA message exchange to update the system increases and also the overhead of recalculating the entire end-to-end path cost increases. A study by Heegaard and Wittner [14] shows how stochastic routing CEAS outperforms link state routing in case of frequent transient failures in the network.

[4]Autonomous System (AS) is a collection of connected routers within a control of common network administrator. See IETF RFC 1930

2.5 Distance Vector Routing (RIP)

Routing Information Protocol (RIP) is one of the IGP protocols that are based on Distance Vector Routing algorithm. Similar to link state routing RIP is used within a single Autonomous System. Also RIP nodes send routing-update messages periodically or in response to any change in the topology. Upon receiving a new update message each node updates its routing table to reflect the changes. However, RIP nodes only maintain information regarding the best path towards the destination. Further, RIP uses hop count as routing metric to calculate the distance between a source and a destination node. In order to prevent routing loops in the network, RIP defines the maximum number of hops in a path to be 15 which in fact limits the size of the network. Any destination node beyond this limit is considered unreachable.

2.6 TCP

The Transmission Control Protocol (TCP) is widely used connection-oriented Transport Layer protocol. TCP ensure end-to-end reliable connection over the Internet. By connection oriented it means that a connection must be established between two nodes before transferring data. Further, TCP includes error detection and error correction mechanism in order to provide the end-to-end reliability.

TCP transfer data in a form of segment which includes both the TCP header and the payload (user data). The segment size may vary according to the payload. However, it should not exceed the Maximum Segment Size (MSS) of the connection. Each TCP segment is assigned a unique sequence number. When a destination receives a segment it sends back an acknowledgment (ACK) for the received segment. The acknowledgment also consists of the next sequence number that the receiver expects to receive [29].

2.6.1 TCP congestion control

TCP uses feedback control algorithm, additive increase and multiplicative decrease

(AIMD) algorithm, to avoid congestion in the network. Congestion builds up when the traffic load exceeds beyond the network capacity. The idea is to increase the transmission rate until loss occurs. The transmission rate is governed by two entities *Congestion window (cwnd)* and receiver's a*dvertised window (rwnd)*. *cwnd* limits the maximum amount of data that sender can transmit while *rwnd* limits the maximum amount of data the receiver is willing to receive. In general, the transmission rate should not exceed the minimum of cwnd and rwnd [2].

For every segment transmitted, a timer known as retransmission time-out (RTO) is set. When this timer expires, TCP assumes that the segment is lost. In addition to this, TCP receiver generates DUPACK for every out-of-order segment received. Beyond certain threshold, TCP interprets continuous DUPACKs as packet loss.

When a segment is lost, TCP congestion control mechanism interprets the loss as a result of network congestion [17], therefore it responds by decreasing the congestion window. The decrease in congestion window size depends on the TCP variation yet most of the TCP variants use the following inter-wined algorithms to handle congestion; slow-start, congestion avoidance, fast retransmit and fast recovery.

2.6.1.1 Slow start / congestion avoidance

Slow-start begins when a TCP connection is established. Initially a congestion window is set to not more than twice the *maximum segment size* (SMSS) that sender can transmit.

$$cwnd \leq 2 * SMSS\ bytes \tag{2.3}$$

For each ACK received, the cwnd value is increased by one SMSS until it exceeds the slow-start threshold (ssthresh) or congestion occurs. After this the TCP enters congestion avoidance phase. During this phase, cwnd is increased by one full sized segment for each round-trip time (RTT). In general, cwnd is increased as in equation 2.4. Thus, cwnd grows exponentially during slow-start phase but grows linearly during congestion avoidance.

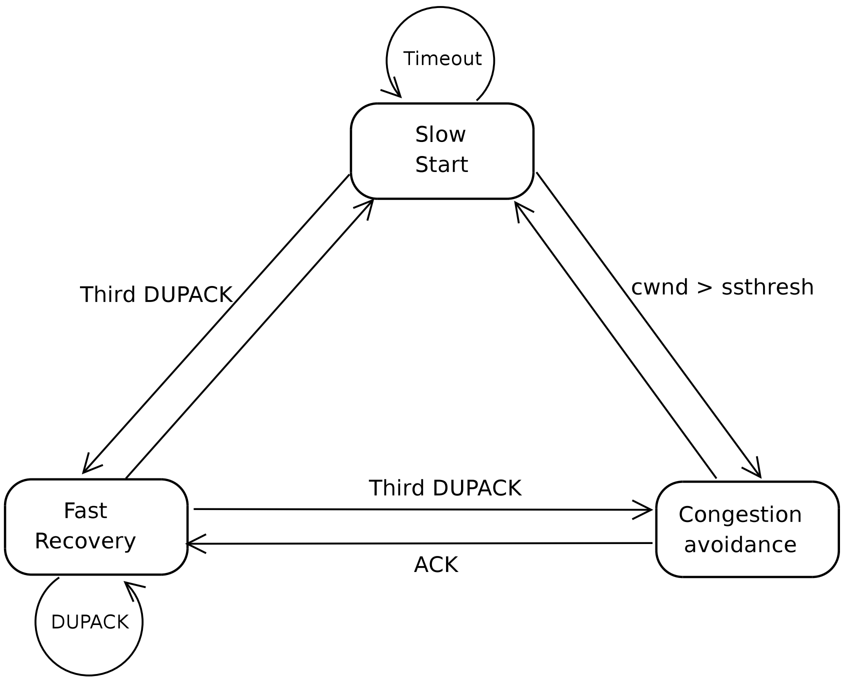

Figure 2.1: TCP congestion control mechanism.

$$cwnd+ = SMSS * SMSS/cwnd \tag{2.4}$$

When the retransmission timer RTO expires, TCP interprets this as congestion. Then, the lost segment is retransmitted and the ssthresh is set to the half of the current cwnd value. Thereafter, TCP re-enters slow start phase with cwnd value set to one full sized segment [17].

2.6.1.2 Fast Retransmit / Fast Recovery

For each out-of-order segment received, a duplicate acknowledgment (DUPACK) is sent by the receiver indicating segment(s) missing. However, as any of the missing segments is received, the receiver should immediately acknowledge indicating that the missing segment has been received.

TCP sender waits for 3 conjugative DUPACK before retransmitting the first unacknowledged segment. Thereafter, ssthresh is set to the half of cwnd, then the cwnd is set as in equation 2.5 and Fast recovery phase is activated.

$$cwnd = ssthresh + 3 * SMSS \tag{2.5}$$

For each additional DUPACK received, cwnd is incremented by one SMSS to reflect that an additional segment has been sent. A new segment may also be transmitted if permitted by the new cwnd and the receiver's advertised window. When a next ACK is received, cwnd is set to ssthresh. This indicates that all the missing segments have been received. Figure 2.1 illustrates the whole procedure [17].

2.7 Effect of Stochastic Routing on TCP

In stochastic routing, the next node along the path is selected randomly irrespective of their previous selections. Therefore, data packets from a same TCP session may follow different paths to the destination. This random behavior has negative effect on the TCP performance. For example; assume p and p' to be two different paths that exist between two nodes A and B, where the path p' has longer propagation delay than the path p. Also assume that a TCP connection is established between the nodes. If a TCP packet generated at time t takes the route through the path p' and next TCP packet generated at time $t+x$ takes the route through the path p then the later packet reaches the destination before the former packet. Thus, packets following different path causes re-ordering of the packets. Although the out of order packet delivery is not due to the packet loss, beyond certain threshold TCP treats such out of order packet receive as packet loss and activates the congestion control mechanism [17].

In our example the out-of-order packet delivery is neither due to the packet loss nor due to the congestion yet TCP enters the congestion control mechanism causing reduce in the TCP performance. As explained in Section 2.6.1, TCP respond packet loss by retransmitting the lost packets and reducing the window size in order to reduce the overload on the network. This unnecessary Fast retransmit / Fast recovery wastes available bandwidth as well as reduces Data transmission rate [1, 31, 24].

Generally, a TCP sender waits for 3 DUPACKs before retransmitting the first unacknowledged packet. After this, TCP enters the fast recovery phase. In order to avoid TCP from entering Fast retransmission and Fast recovery phase the propagation delay on the longer path p' should be less than the time required by next 3 conjugative packets to reach the destination. Let x be the interval of packet generation, d and d' be the total propagation time for a packet to reach from A to B via p and p' respectively. Assume a packet generated at time t takes the longer path p' and other 3 conjugative packets generated at $t+x$, $t+2x$ and $t+3x$ all take shorter path p. Therefore, to avoid retransmission the packet via path p' should reach destination B before time $t+3x+d$. i.e.

$$t + d' < t + 3x + d \; or$$

$$d' - d < 3x \tag{2.6}$$

Let t_{pab} and $t_{p'ab}$ be the total time, excluding queuing delays and router processing delays, required for a packet to travel from A to B via p and p' respectively. Assuming t_q and t_r be the queuing delay and the processing time for a router respectively. Also assuming that the path p consist of n numbers of routers and p' consist of $n+k$ numbers of routers then from Equation 2.6;

$$t_{p'ab} + (n+k)(t_r + t_q) - \{t_{pab} + n(t_r + t_q)\} < 3x, \; or$$

$$t_{p'ab} - t_{pab} < 3x - k(t_r + t_q) \tag{2.7}$$

The packet generation interval, x is inversely proportional to the Data transmission rate but directly proportional to the packet size. Further, $t_{p'ab}$ and t_{pab} is also proportional to the packet size. Therefore, for a path p' with longer delay we can avoid spurious retransmission by reducing the Data transmission rate and increasing the packet size. However, the Maximum Transmission Unit available for Ethernet frame is 1500 bytes. Further, k in Equation 2.7 indicates the number of additional routers in the longer path p'which refers that the delay difference decreases as we increase the number of additional routers in the longer path p'.

In addition to this, a study by Blanton and Allman [4] lists further negative effects of packet reordering

- TCP's standard congestion control algorithms prohibits to transmit any packet when a DUPACK is received until fast retransmit is triggered. However TCP stores permission to send new data and if an ACK covering new data arrives before the fast retransmit is triggered then the burst of data is sent on this ACK will be larger than if reordering had not occurred.
- TCP reordering causes RTT sampling ambiguous; Generally RTT is estimated using a timer that starts just before a given segment S is transmitted and then stopped as the ACK covering the segment arrives. During retransmission, the sender can not make sure whether the ACK covering the segment is in response of the first transmission of the segment or in response to the retransmission.

2.8 TCP variants

TCP Reno is widely used TCP protocol on the Internet. It uses congestion control mechanism described in Section 2.6.1. However, fast recover used in Reno does not recover efficiently when there are multiple packet losses in a single flight [2]. In order to deal with multiple packet losses, NewReno [9] is designed with modification in the original Fast Retransmit and Fast Recovery algorithm.

NewReno recovers multiple losses one packet per round trip time using partial acknowledgments concept. During multiple packet losses, the first non-duplicate acknowledgment received does not necessarily indicate that all the packets transmitted prior to the Fast retransmit have been successfully received. However, Reno leaves Fast Recovery phase as soon as it receives first non-duplicate acknowledgment. Thus, in case of multiple packet losses, the additional DUPACKs received after the first non-duplicate ACK causes Reno to enter another cycle of the Fast Retransmit and Fast recover phase with further decrease if *cwnd* and *ssthresh*. Unlike Reno, NewReno remains in the Fast Recovery phase retransmitting one packet per RTT until every lost packet have been retransmitted and acknowledged.

During transmission, NewReno saves the highest sequence number that has been transmitted. When a packet loss is indicated by 3 continuous DUPACK, it performs Fast Retransmission and Fast Recovery as usual. Reno exits recovery phase as soon as a non-duplicate ACK arrives however, NewReno performs additional verification to confirm that the ACK actually covers the highest sequence number stored. If the verification fails, it assumes the ACK as partial ACK and retransmits the first unacknowledged packet.

The major drawback of NewReno is that it cannot predict multiple packet losses until it verifies the ACK against the highest sequence number stored. Thus, this may take substantial amount of time to recover from a series of loss depending on the number of packets lost and the size of RTT [9].

In general, an ACK only confirms that all the packets up to the number indicated by that ACK has been successfully received but does not provide any information about further received packets. In other word, TCP does not send ACK for those packets that are received beyond the one it expects [2].

TCP Selective Acknowledgment (SACK) [21] is specially designed to handle multiple packet losses correctly. TCP SACK implementation uses the conventional congestion control algorithm as used by TCP Reno. Similarly, the Fast Retransmit and Fast Recovery phase in SACK implementation is triggered after receiving 3 continuous DUPACK. The main difference between the two implementations is their behavior during multiple packet losses in a single flight.

An ACK with SACK option is sent by receiver informing the sender about the arrival of non-contiguous segments. The SACK option contains a list of the contiguous data blocks received and queued at the receiver. Two 32 bit unsigned integer are used to indicate the starting and the ending of each blocks. In general, a 40 byte ACK can include maximum of 3 SACK blocks. The SACK option must always provide the information about the most recently received segment to inform sender about the currently missing segments. When missing segments are received they must be acknowledged immediately. In this way, SACK method helps to inform the sender about the correctly received packets as well as the missing packets. The receiver uses SACK option as reference during retransmission to correctly retransmit only those segments that has been reported missing [21].

Chapter 3

Simulation Module

We used simulations to compare the performance of TCP over CEAS, LSRP and DVR routing protocols. We carried out all our simulations using Network Simulator 2 (NS2)[1]. NS2 is a discrete event simulator widely used for research of IP network on the packet level.

3.1 Simulator Basics

NS2 simulator is a widely used tool for evaluation of Internet Protocols which is based on two languages; Object oriented C++ classes and OTcl[2]. The compiled C++ classes acting as the kernel of the simulator holds the necessary details and the operations of different protocols. On the other hand, the OTcl acts as the user interface allowing user to define network topologies, specify protocols and applications that one wish to simulate. Thus NS2 provides a reproducible and controllable environment for the evaluation of the implemented protocols. The standard distribution of NS2 includes all the necessary support for the Link State Routing Protocol and the Distance Vector Routing.

All previous studies regarding CEAS were based on the NS2 module developed by

[1]For more information on NS2, see http://www.isi.edu/nsnam/ns

[2]OTcl, short for MIT Object Tcl, is an extension to Tcl/Tk for object-oriented programming. (http://otcl-tclcl.sourceforge.net/otcl/)

Helvik and Wittner [16]. However, for this work we received a new version of CEAS module from the department of telematics, NTNU. This new CEAS module is designed on top of the NS2 extension that supports *multi-** networks developed by Paquereau et al. [27, 26]. The *multi-** network extension provides better support for wireless and mobile networks. This extension enables nodes to support multiple interfaces of different types. Such multiple interfaces allow the nodes to communicate over multiple channel at the same time. All new features are added in a form of modules making it easier to enable and disable such features by enabling and disabling the modules [26].

Typically a node in NS2 consists of two Tcl objects: an address classifier and a port classifier. These classifiers are responsible for distributing incoming packets either to a corresponding agent or to an outgoing link [8]. The extension provided by Paquereau et al. [26, 27] adds two new objects; NetworkLayerManager and NetworkLayerUnit. The NetworkLayerUnit consists of a ForwardingUnit and a RoutingUnit which handles the data packets and the routing packets respectively The multi-* network extension is briefly explained in Appendix A.

3.2 CEAS Extension Modification

The new CEAS extension that we received for this work was still under developing process. It lacked support for self-tuned refresh rates [13], also the ForwardingUnit module for CEAS had not been implemented and the Link cost calculation ignored the traffic loads on the path. However, features like *elite-selection*, *routing-loops*, *temperature approximation* and *pheromone approximation* were implemented as modules making it easier to configure through OTcl script.

At the time we begin our work, no studies related to data packets had been carried out with the new CEAS extension. Moreover, the incomplete ForwardingUnit made our study difficult as the unit is responsible for handling data packets. Further, the traffic loads on the link had no effect on the link cost which made the system difficult to handle load balancing and re-routing during congestion. So, before proceeding to the studies our first objective is to implement a working ForwardingUnit module and to adjust Link cost calculation.

The new CEAS extension is divided into two modules; common module and plain module. The common module acts as the core of the CEAS system. It provides the basic properties and functionalities of the system. It also defines the premises of the system. The plain module is a subclass of the common CEAS module which defines and expands the functionalities of the system.

3.2.1 CEAS ForwardingUnit Implementation

The CEAS ForwardingUnit is an inherited class of the ForwardingUnit (3.1). This unit handles all data packets that pass through the CEAS NetworkLayerUnit. All the packets that are destined to the node are passed upwards to the corresponding agent. Likewise, the packets that are in transit are either forwarded to a neighboring node based on the pheromone level associated with the corresponding path or dropped if no neighbor node exist. The selection of next neighbor node along the path is made using stochastic sampling with replacement method (Roulette wheel method) [3] among the neighboring nodes taking associated pheromone values as the parameter of selection. (See Appendix B.1 for detail.)

3.2.2 Cost Path Modification

A path cost $L(\omega)$ is calculated using the Equation 2.2, which is the sum of all the link costs along the path. The link cost is measured using short term average delay on the link. The average delay changes according to the traffic load on the link. Therefore the calculation of end-to-end delay as the path cost reflects the quality of the path at the given time.

The CEAS extension that we received for our work however did not account any traffic load on a path during the path cost calculation. Thus, the path cost only depends on the transition delay on the links but not on the quality of the path.

In order to achieve the variable path cost that changes with respect to the traffic load we added a time stamp on each ant at the time of their creation. This time stamp enables us to measure the actual end-to-end delay on the path that the ant traveled. We used this information during the path cost calculation.

3.2.3 Record Single Hop Route Address Implementation

In our previous work [33], we found that large number of packets are lost due to looping when there is link failures. Such loopings are found in this work as well. (*see Section 4.3 for detail.*) To avoid such loopings we introduce a technique that prevents forwarding packets back to the node that recently forwarded them. To implement this we use *prev_hop()* method of common packet header to retrieve the address of the recently forwarded node. We then remove the node from the list of neighboring nodes so that the next node selected would not be the one that recently forwarded the packet. However, in conditions where there remain no neighboring node other than the previously forwarded node the packet would have to be ultimately dropped by the routers. Thus, in such conditions we allow forwarding the packet back to the previous node.

3.3 Production

All the simulations in this report related to the CEAS system are run with 15 replications per scenarios using different seeds for the random number generator. Other simulations related to the Link State Routing and the Distance Vector Routing are carried out once per scenario. The simulation output from all the replications are synchronized and post processed using the AWK[3] programming language. The graphs from the simulation results have been plotted using gnuplot[4].

[3]AWK is a general purpose programming language designed for processing text-based data

[4]Gnu plot is a versatile command-line driven interactive utility for generating plots of data and functions.

3.4 Parameters

The input parameters to the simulator are given using the TCL user script. The parameters used for CEAS system are listed in Table 3.1. Similarly the default parameters used for Link State routing protocol and Distance vector routing protocol are listed in Table 3.2 and Table 3.3.

The *elite selection* was introduced by [12] to reduce the overhead of the backward ants carrying insignificant updates. With elite selection only those ants following the path with the best cost values so far are returned to update pheromone. During forward search explorer ants select next node randomly. Excluding the explorer ants update from the elite selection allows the explorer ants to return back. While returning, the explorer ants update pheromone level along their way. In a small network like 4.1 where there are only few paths available and the differences between the path costs are very small, such updates would increase the pheromone level along the alternative paths as well. Although allowing the explorer ants to return back helps system reacts faster to network dynamics, the increase in the probability of selecting alternative path causes reordering of the packets. Thus in this study, we use elite selection on all ants.

During forward search ants list all the addresses of the nodes that they pass by. At the destination, the list is analyzed to remove cyclic paths that the ants have traveled. Since our path cost calculation depends on the entire end-to-end delay, removing such cyclic paths does not reduce the delay experienced by the ants along the cyclic paths.

The ants following the cyclic path experience longer end-to-end delay than those that travel straight. Suppose after removing the cyclic paths both the paths become identical then this might create an ambiguity during the path cost calculation. However, this is not the case in *subpath* method introduced by Kjeldsen [19], where the ants not only stores the addresses of the nodes but also record each link cost along the path. At the destination, the reduction of the cyclic path also reduces the link cost around the cyclic path.

Parameter	**Description**
Seed	179324 + timid (for each replication timid is incremented by 1)
Simulation Time	Simulation time varies according to the scenarios
Initialization Phase	50 seconds
Initialization Phase ant rate	100 ants per second (All ants are explorer in this phase)
Ant-rate normal	1 to 20 ants per seconds
Ant-rate explorer	10 % of normal ant rate
Processing delay	0 (The total delay is specified for each link in the topology)
Beta (β)	0.98 (Evaporation, for detail see [16])
Rho (ρ)	0.01 (Search focus, for detail see [16])
Elite Selection	All Ants
Cycle treatment	Allow cycles

Table 3.1: CEAS parameters

Parameter	**Description**
preference	120
advertInterval	1800 seconds (Periodic route update interval)
Spf wait time	0.01+0.250 (Shortest path first wait time)

Table 3.2: LSRP parameters

Parameter	**Description**
preference	120
advertInterval	2 seconds (Periodic route update interval)
INFINITY	32 (to determine the validity of route, see NS2 manual for detail)

Table 3.3: DVR parameters

3.5 Topologies

Three different topologies are used in our studies. A simple eleven node network as shown in Figure 4.1 is used to understand the behavior of CEAS during different network events and to study its corresponding effect on the TCP performance. This eleven node network is chosen in order to have a simple structure in the topology which is easier to study as well as easier to visually examine the simulations. Further, we have used identical bandwidth of 100 Mbps and transmission delay of 1 milli-second on each links to have a regular structure on the topology. However, in a particular case we varied bandwidth on a link which is explained later. We use the results from this network as a reference to understand and analyze the behavior of TCP under various network conditions in more complex networks.

The remaining two topologies are used as case studies; a twelve node network and the 58 node topology extracted from the Uninett[5] network. The twelve node network, as shown in Figure 5.1, is chosen to have more complex network than the one we use for our basic studies. We introduced various levels of background traffics and multiple link failures to analyze its effect on TCP performance. Finally, we used 58 node Uninett network topology as depicted in Figure 5.12. This topology is chosen to provide a realistic setting for our study. The routers in all of our topologies use maximum queue size of 60 segments and a drop-tail queuing strategy.

3.6 Network Dynamics

NS2 provides functionality to simulate network dynamics like link failures. In this work, we study the system using multiple link failures as well as with multiple level of background traffic. We have categorized link failures into two types on the basis of the link failures and their re-establishment time. The first types of failures that we call steady link failures takes longer time to re-establish. On the other hand, the second types of link failures that we call transient link failures takes very short time, usually

[5]www.uninett.no

less than a second, to re-establish. Further, we introduce such transient link failures for certain time interval to exhibit flapping or unstable behavior of the link. These transient links are used in our case studies.

A link cost is calculated on the basis of an average delay on a link. The delays on each link are specified on the topology and remain constant throughout the simulation. However, the path cost varies according to the traffic load along the path. Data rates, Connection types, packet size etc are defined using TCL scripts which also remain constant throughout the simulation.

3.6.1 TCP Connections

Each scenario is studied using two variants of TCP; TCP Reno and TCP Sack. TCP Reno is one of the most widely used TCP congestion avoidance algorithms on the Internet. However, TCP Reno suffers during multiple packet losses in a same flight of transmission. On the other hand, TCP Sack is better designed to handle multiple packet losses. In this work we have compared the results from both the variants. All of our simulations are conducted using a bulk TCP transfer. Although such TCP connections are not realistic, it allows us to measure the ideal TCP performance on the CEAS system.

The TCP sources are created using a Constant Bit Rate (CBR) source that generates data at a constant rate. In all of our simulations the TCP data rates generated are more than 70% of the link capacity. The TCP source uses the maximum congestion window up to 200 segments. Time to Live (TTL) values for smaller networks are set to 10 whereas for the large network they are set to 30. The TCP Reno receiver uses TCP Sink with delayed ACKs and the TCP Sack receiver uses the *sack1* TCP sink with delayed ACKs. The delayed ACK timer is implemented with 100 ms granularity. The default allowed maximum DUPACK is 3 as well as the TCP sender uses the default clock granularity of 60 ms for the retransmission timer. The TCP packet size used in our simulations is usually 1500 bytes which is the *Maximum Transmission Unit* used in the Ethernet. However, packet size of 512 bytes and 9000 bytes[6] are also used in our simulations.

3.6.2 Background Traffic

Some of our simulations are conducted under influence of background traffics. All the background traffic in this study is generated using CBR UDP source. During our case studies we use various levels of background traffic. Such background traffic is generated using multiple CBR UDP sources. The background connection pairs are selected in such a way that at least one of the connection shares a common link with the TCP connection when there are no link failures. In order to have such a shared links some of the neighboring nodes of TCP source and destination as well as themselves are selected as background sources or sinks.

[6]Jumbo frames allows packet size up to 9000 bytes (http://sd.wareonearth.com/~phil/jumbo.html)

Chapter 4

Measuring TCP performance on CEAS system

This chapter presents the simulation studies of the CEAS system under different network dynamics and the corresponding effects on the TCP behavior. The simulations performed in this section are approaches to find out the answers to the research questions listed in Section 1.3

All of our simulation studies in this chapter uses a simple eleven node network as depicted in Figure: 4.1. We choose a node on either side of the network as a source and a destination so that there are two paths between them. Each scenario is simulated for both TCP Reno and TCP Sack and the results are compared. The TCP performance studies are divided into the following sections:

Section 4.2, TCP performance on steady network

The time required to stabilize the pheromone levels along the available paths and the number of ants participating on this stabilization process are measured. As well as the effect of starting TCP connection immediately after the initialization phase is examined.

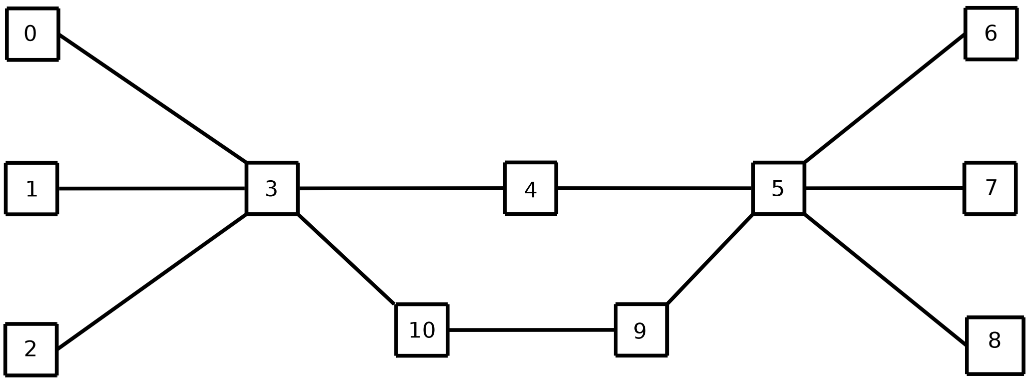

Figure 4.1: A simple 11 node network

Section 4.3, TCP performance during Link Failure

The behavior of the CEAS system during link failure is observed and the time required to update and re-stabilize the pheromone level along the alternative path is measured. In addition to this, the effect of link failure on TCP performance is observed and the time required for the TCP to resume connection and regain its full data rate after the link failure is also measured.

Section 4.4, TCP performance during Link Congestion

The behavior of the CEAS system during link congestion along the best path is observed. Further, how the system diverts the traffic during congestion is examined. The performance of TCP during link congestion is studied and the effect of increasing the packet size is also discussed.

Section 4.5, TCP performance different capacity links

The CEAS system on the network with varying link capacity is studied. The system is examined using the lower capacity link along the best path to examine whether the system selects the alternative path as the best path or continues to use the lower capacity link as the best path. The performance of the TCP under such network is also studied and also the effect of increasing the packet size is examined.

Section 4.4, Multiple TCP connections

The performance of the TCP when there are more than one connections is studied. The study is focused on finding out whether both the TCP connection struggles to use the best path or the system diverts the TCP streams on separate path.

Section 4.4, Previous Hop memory technique

The improvement on the TCP performance during link failure after implementing previous hop memory technique is studied.

4.1 Performance Metrics

The following performance metrics are applied while measuring the performance of TCP on CEAS system:

- **The best path** - The path with the shortest end-to-end delay from the source node to the destination node.
- **Pheromone Stabilization period** - A point in time when the probability of normal ants following the best path is 100 times higher than the other paths in the network.
- **Re-stabilization period** - A point in time when the system updates pheromone level to elect new best path in response to the network events such as link failure, link Establishment or congestion.
- **TCP data rate** - the rate at which TCP source transmits data.
- **TCP Throughput** - The total amount of TCP data transferred after starting a TCP connection over the entire interval of the simulation.

4.2 TCP performance on steady-state network

Before measuring the TCP performance in a network with CEAS system, it is important to observe the time required for the CEAS system to stabilize the pheromone levels along the available paths. Once the pheromone level stabilizes the majority of ants follow through the best path making it the most popular path in the network. The stabilization period depends on the number of the ants in the network. Thus, we use simulations at different ant rates and compare the results. Then, we observed the effect of starting TCP connection immediately after the initialization phase using TCP data rate as the performance metric. We measured the time required for a TCP source to transfer packets at full data rate. TCP connection using both of the TCP variants is studied.

Scenario:

To measure the stabilization period two nodes; *node 0* and *node 8* are selected as a source and destination nodes. In this simulation only ant traffic is generated but not the data traffic. Elite selections are made on all ants including the explorer ants. Further, cyclic paths are not removed at the destination. The scenario is simulated for *600 seconds* with the first *50 seconds* as the *initialization phase*, during which all ants generated are explorer ants searching the possible paths. The stabilization period at different ant rates are compared and graphs are plotted with 95 percent confidence interval. Throughout the simulation, the network remains steady with no link failures.

In second phase of this simulation we start the TCP connection right after the initialization phase i.e. at *50 sec.* and measured the time required for the TCP source to transmit data at the full rate. The scenario is simulated for *1800 seconds*. The TCP receiver uses the *Sack1* TCP sink along with delayed ACKs for TCP Sack whereas simply *TCPSink* with delayed ACK for TCP Reno. The packet size, data rates and other TCP parameters are as in the Table 4.1.

Results and Discussion:

Figure 4.2 shows the Pheromone Stabilization periods at different ant rates. The time required to stabilize pheromone values decreases with the increase in the ant rates.

Parameter	**Values**
Data Rate	75 Mbps
TTL	10
Window Size	80
Packet Size	1500 bytes

Table 4.1: TCP configuration

Ant Rate (Normal + Explorer)	**Mean Time**	**Approx. Number of Ants required** (Normal + Explorer)
1 + 0.1	151.2	≈166
2 + 0.2	73	≈160
3 + 0.3	49.8	≈164
5 + 0.5	30.7	≈169
10 + 1	14.9	≈164
15 + 1.5	9.8	≈162
20 + 2	7.52	≈165
Average		**≈164**

Table 4.2: Pheromone Stabilization Phase with Elite Selection on All Ants

The Table 4.2 summarizes the stabilization time period as well as the number of ants participating in the stabilization process. The number of ants generated during the initialization phase is excluded. The Table shows that the number of ants participating in the stabilization process at all ant rates is almost equal. From these results, we can assume that at an ant rate of 164 ants per second the system would establish the best path in our network immediately after the initialization phase.

Figure 4.3 compares the time taken by TCP Reno and TCP Sack to transmit data at full rate i.e. 75 Mb for our scenario. The results from both of the TCP variants at different ant rates are compared. The figure shows similar results to that of the pheromone stabilization process; at higher ant rates the time required by both the variants are shorter. Graphs plotted in the figure also indicate that at all ant rates the TCP Sack takes shorter time to transmit at full data rate than the TCP Reno. However, at higher ant rates the time differences between the variants are smaller.

In these simulations we start the TCP connection immediately after the initialization phase, at this moment the pheromone levels in the network are yet to be stabilized.

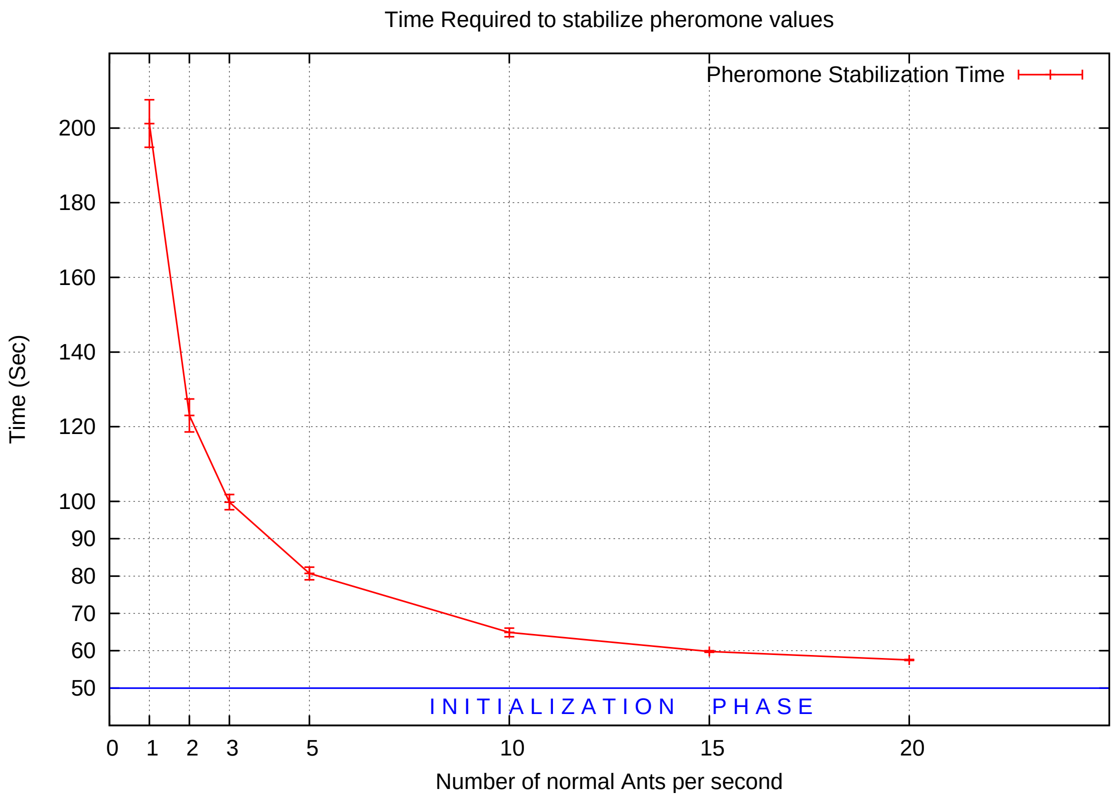

Figure 4.2: Pheromone Stabilization period

Therefore, the system makes use of both the available paths to deliver the TCP packets resulting out-of-order packet delivery.

The negative effect of the out-of-order packet delivery on TCP throughput is discussed above in Section 2.7. Also, during multiple packet losses TCP Reno suffers from cycles of re-entering Fast Transmit and Fast Recovery phase resulting further decrease in the TCP window size. On the other hand, TCP Sack does not leave the Fast Transmit and Fast Recovery phase immediately after retransmitting the first unacknowledged packet in response to the 3 continuous DUPACK received. Further, TCP Sack benefits from SACK enabled ACK which informs the TCP sender about the currently missing packet at the receiver.

Figure 4.4, shows congestion window, *cwnd,* of both TCP Sack and TCP Reno at different ant rates. The figure also shows that the *cwnd* of TCP Reno fluctuates more

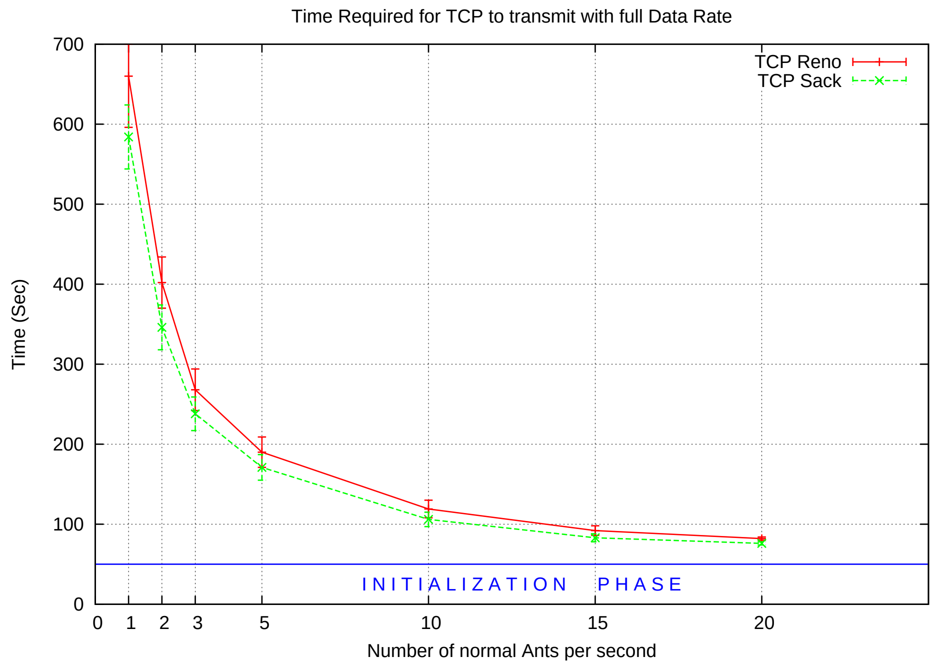

Figure 4.3: Time required for TCP to transmit with Full Data Rate

frequently than that of the TCP Sack due to the cycles of re-entering the Fast Transmit and Fast Recovery Phase.

In brief, until the system stabilizes the probability of out-of-order packet delivery is higher resulting poor TCP performance. But, as the system stabilizes, the *cwnd* for both the variants increases. Figure 4.5 provides an image of out-of-order packet delivery rate at an ant rate of 5 ants per second for both the TCP variants. The graphs are plotted using a result from an instance of the simulations. Initially the rate of out-of-order delivery is larger but the rate decreases as the pheromone level stabilizes. Therefore, if we start the TCP connection sometime after the pheromone stabilization phase, the data transmission would start at full rate.

However, this is not the case in LSRP and DVR; the initialization phase in these protocols takes very short time, less than a second for a small network like ours. Further, the data packets are always forwarded through the same path which prevents reordering of packets resulting stable TCP throughput.

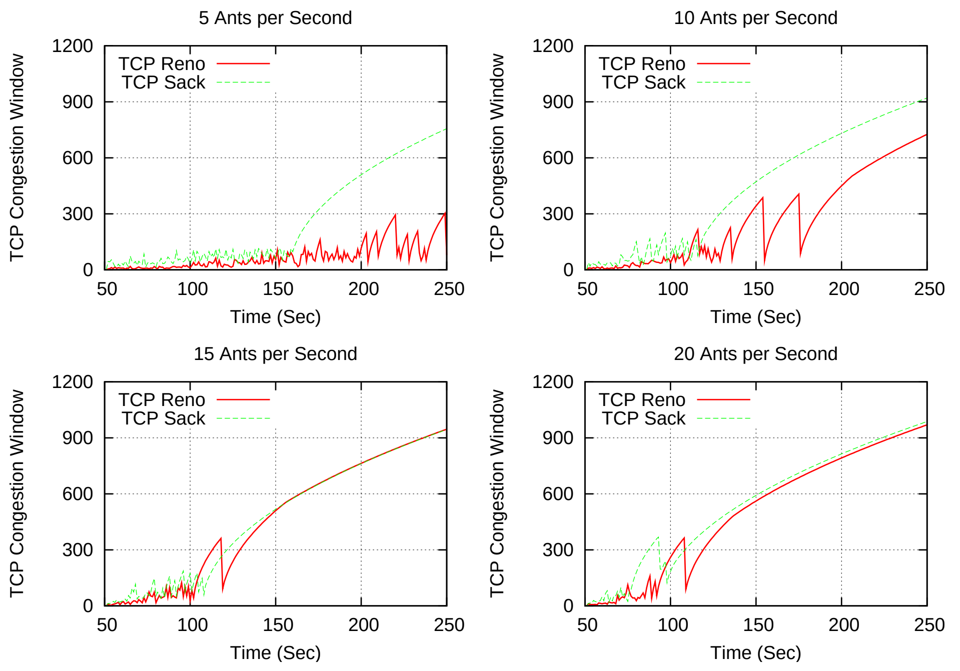

Figure 4.4: cwnd at different ant rate

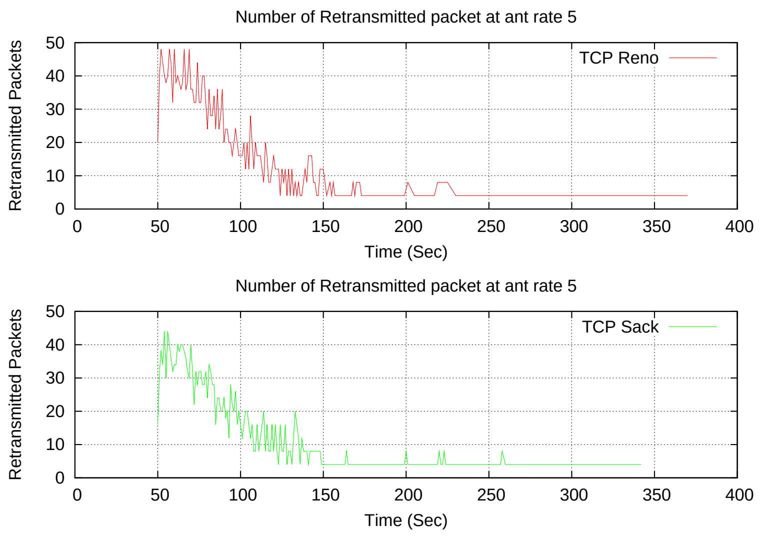

Figure 4.5: Out-of-order packet received at Ant rate 5 (an instance of simulation)

4.3 TCP performance during Link Failure

The next step in our study is to understand the behavior of the CEAS system under the influence of network dynamics such as link failures and re-establishments. In this study, we measure the time required for the system to update and re-stabilize the pheromone values in order to divert the traffic through alternative path after the link failure. Similar to the simulation carried out in Section 4.2, we measure the system at various ant rates. We also measure the time taken by the TCP source to resume data transmission, via the immediate best path, after the link failure. The effect of the link re-establishment is also studied.

Scenario:

Similar to our previous scenario in Section 4.2, we use *node 0* and *node 8* as our source and destination nodes. Similarly, the packet size, Data rates and other parameters are set identical. Further, the scenario is simulated for both the TCP variants at different ant rates. However, we start the data traffic *100 seconds* after the stabilization phase to avoid out-of-order packet delivery. As a consequence, the TCP source transmits data at the full rate.

The TCP traffic generation start time varies according to the stabilization time. Similarly, the simulation time varies according to the ant rates because the system with the lower ant rates takes longer time to update the pheromone values. The link failure event is scheduled after *100 seconds* of the TCP transmission starts. At this time, we break the link connection *node 4* and *node 5* in order to have a link failure on the best path. Finally, after *100 seconds* of successful re-route at full data rate we re-establish the failure link. The results from both the TCP variants are compared and graphs are plotted with 95 percent confidence interval.

Results and discussion

Figure 4.6 compares the time required by the CEAS system to update and re-

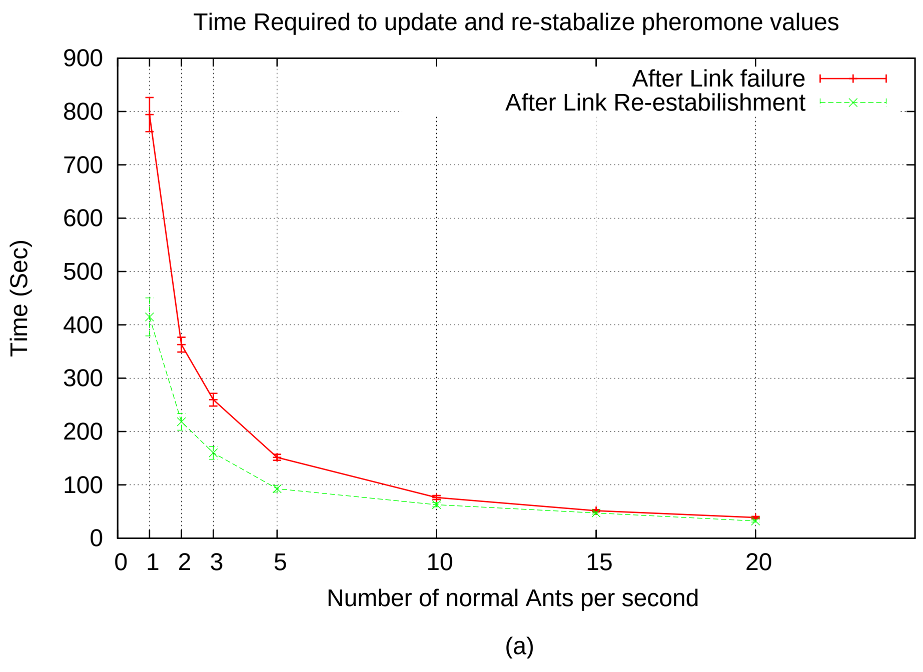

Figure 4.6: Re-stabilization phase

stabilize the pheromone values at different ant rates in order to divert the traffic after the link failure and after the link re-establishment. The results show that during the pheromone re-stabilization process, it requires more number of ants than during the initial stabilization process. During the system initialization, the pheromone level along both the paths are at their minimum levels but the stabilization process increases the pheromone level much higher along the best path compared to the alternative path. In our scenario, the link failure event occurs after the stabilization phase and also along the best path. Therefore, the system requires more number of ants to reduce the pheromone level along the failure path and raise the level along the alternative path.

The results also show that during the update process, more number of ants is required at the higher ant rates than at the lower rates. The reason is that we start the TCP traffic 100 seconds after the stabilization phase and also the link failure is scheduled 100 seconds after the TCP connection. During this time, the number of ants updating pheromone values is larger at the higher ant rates which further reduce the pheromone level along the alternative path compared to the pheromone level at the lower rates. Therefore, proportional numbers of ants are required in the re-stabilization process. Further, the slower pheromone update process at lower ant rates is also benefited by the nature of the control variable i.e. temperature which decreases over time. The control variable, temperature is a main parameter used in pheromone update method (see [16] for detail).

The results from the link re-establishment, depicted in Figure 4.6, shows the similar results to that of link failure process. However, the time period for re-establishing the best path is shorter than the re-route time during link failure.

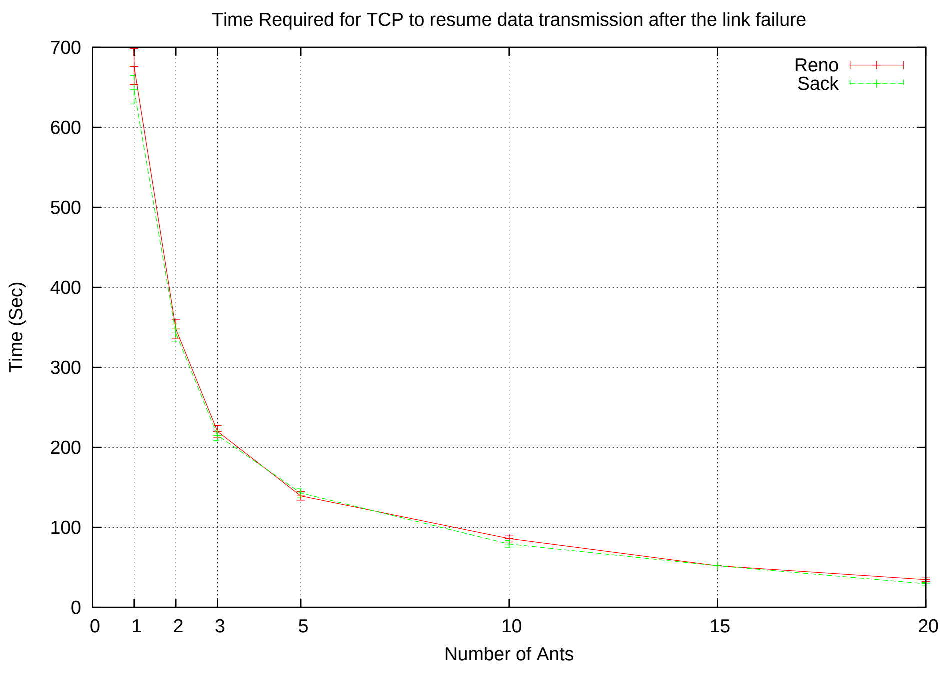

Figure 4.7: Time required for TCP to resume transmission after Link Failure

Figure 4.7 compares the time taken by TCP Reno and TCP Sack to resume data transmission after the link failure where as Figure 4.8 compares the time taken by both the TCP variants to regain the full data rate after the data transmission is resumed. The time taken by the TCP to resume data transmission depends on the pheromone update process. During which, the increase in the pheromone level along the alternative

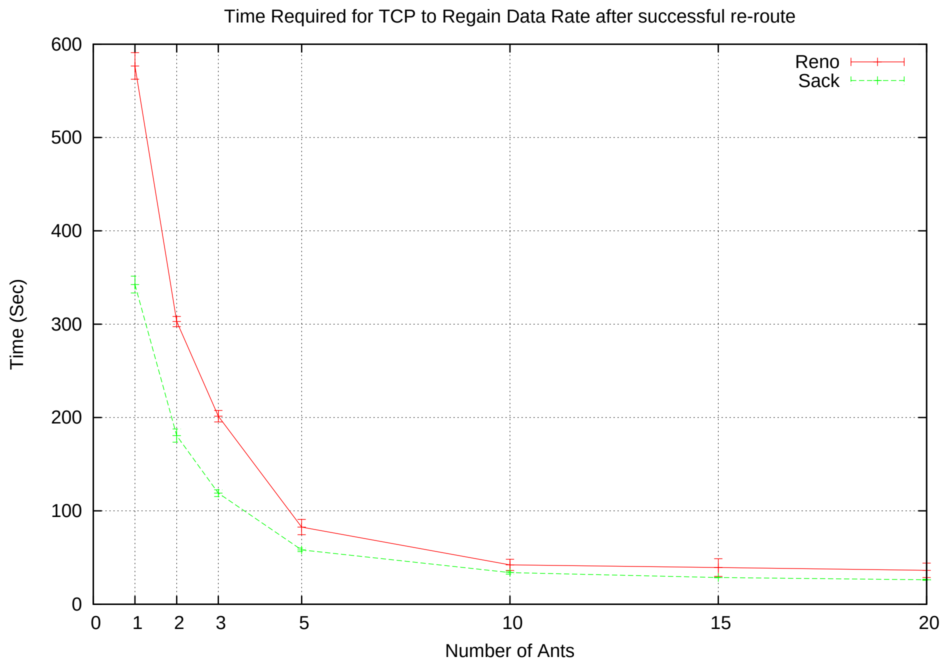

Figure 4.8: Time required for TCP to regain full data rate after transmission is resumed

path increases the probability of selecting that path. Once the probability of selecting the alternative path is significant enough the TCP data transmission resumes. Thus, the results from both the TCP variants are similar. However, it takes significant amount of time before TCP regains full data rate because of the random packet loss.

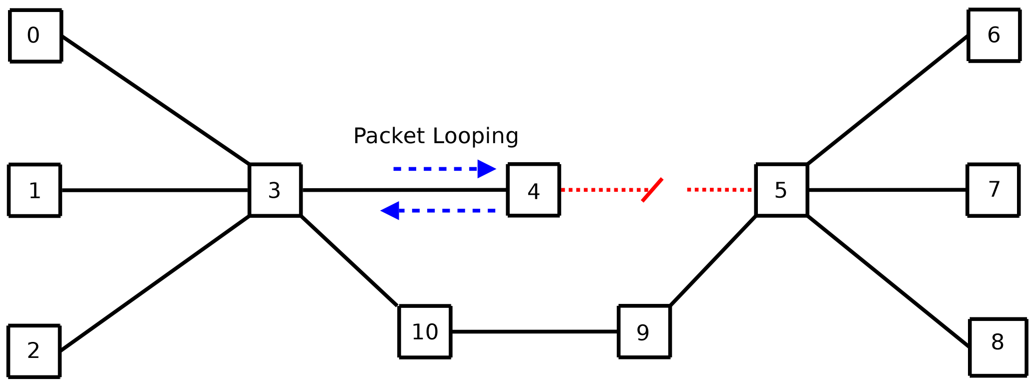

Figure 4.9: Micro-Loops after Link Failure

At *node 3* as shown in Figure 4.9, though there remains only one available path towards the destination, the probability of selecting *node 4* over *node 10* as the next node still remains higher. Further at *node 4*, the failure of only available route towards the destination other than the route back to the *node 3* worsen the situation. This causes a large number of TCP packets to loop around *node 3* and *node 4* until their TTL value expires and finally dropped by the routers. The number of packets looping around *node 3* and *node 4* decreases as the pheromone level re-stabilizes resulting regain in TCP throughput.

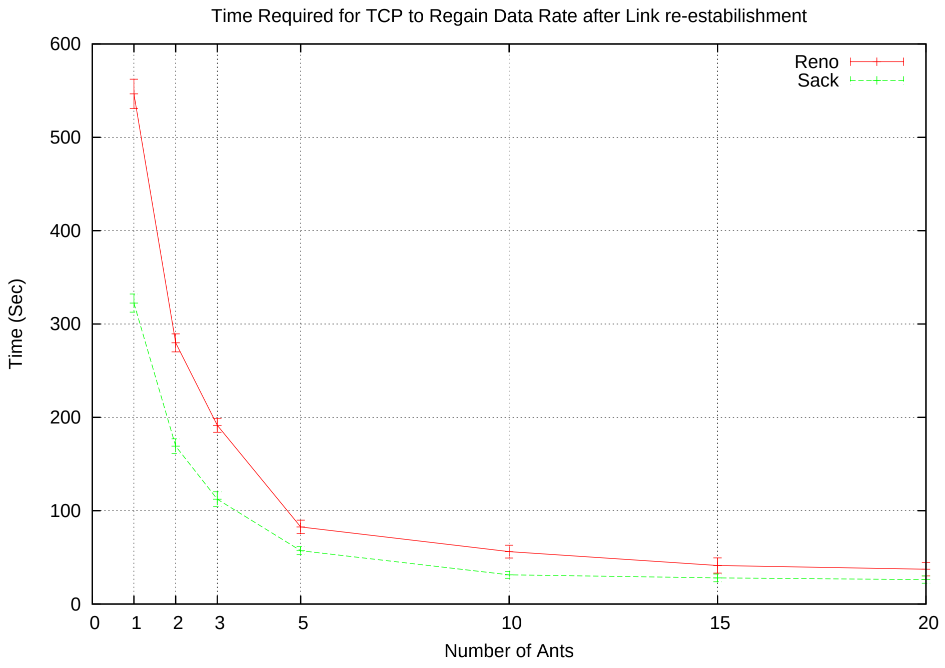

Figure 4.10: Time required for TCP to regain full data rate after link re-establishment

Similarly Figure 4.10 compares the time taken by both the TCP variants to regain full data rate after the link re-establishment. During pheromone update process, the system routes traffic through both the path causing out-of-order packet delivery. However, as the pheromone stabilizes the rate of out-of-order delivery reduces resulting gain in TCP throughput. Thus, considerable amount of time is required depending on the ant rates before the TCP regains the data rate.

Figure 4.8, Figure 4.10 and Figure 4.3 shows that depending on the ant rates, the performance of TCP during pheromone stabilization periods are similar. The results from all these three graphs also shows that the TCP Sack regains faster compared to the TCP Reno.

4.4 TCP performance during Link Congestion

As a stochastic routing, CEAS uses multi-path routing. CEAS system continuously evaluates all available paths and selects a best path based on the quality of the path. Depending on the traffic loads the best path between two nodes changes over time. When there are no traffic loads on the network the CEAS selects shortest path as the best path. In this section we explicitly introduce heavy traffic loads along the shortest path forcing the system to re-select the best path. In this section we study the behavior of CEAS system during path congestion and analyze the effect on the TCP performance. We also compare the results with the LSRP and the DVR under similar configurations.

Scenario:

Like in our previous scenarios, we use similar parameters and configurations. We load the link connecting *node 4* and *node 5* with a heavy traffic. This link is selected in order to introduce congestion along the best path. We load the link with *97* percent of the link capacity as the background traffic. For this we generate CBR UDP connection at data rate of *97 Mb* between the *nodes 4* and *5*. We use TTL value of *1* for UDP packet in order to load traffic only on that link. Similarly, we start the link congestion *100 seconds* after the stable TCP connection. For this study, we use TCP throughput as the performance metric. Further we simulate the system only at the higher ant rates i.e. above *5* ants per seconds. The results from both the TCP variants at different ant rates are plotted with the 95 percent confidence interval. We compare the results against the results from LSRP and DVR simulated under similar configurations.

Results and Discussions:

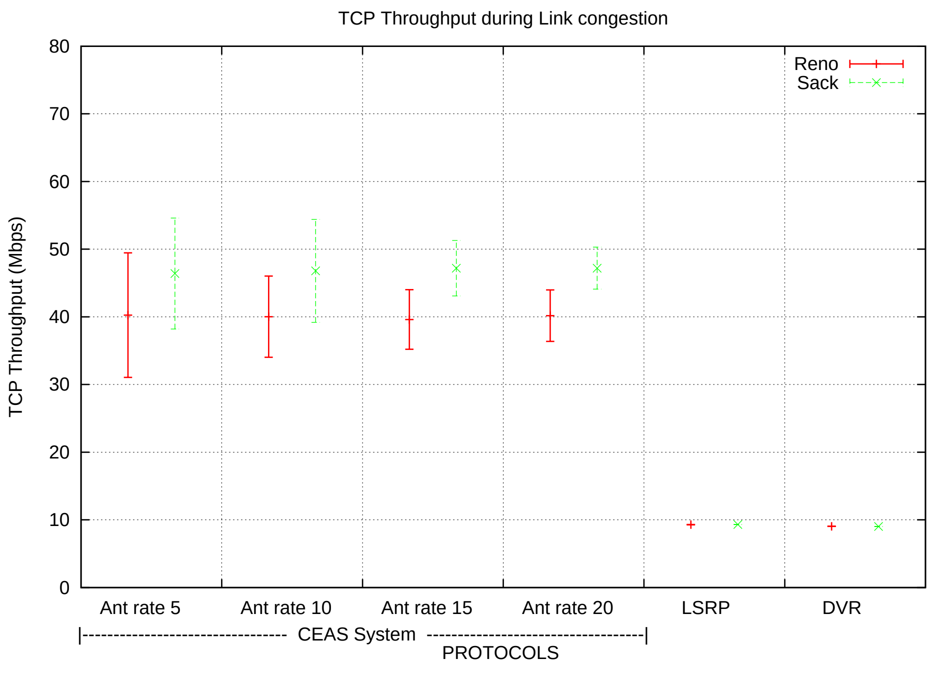

Figure 4.11: TCP Throughput during congestion

During all simulations we observe that after starting the link congestion, the TCP data transfer rate never reaches its full rate. At the receiver node we observe continuous out-of-order packet delivery. While visually examining the simulations we find that the system uses both the available paths to some extent. We find that the use of the shortest path is more frequent than the non-congestion path. The system does not completely divert the traffic along the non-congested route. We also find that the overall TCP throughput at different levels of ant rate are almost similar.

There are mainly two reason for this; TCP congestion control algorithm which reduces the data rate to avoid the congestion and the large difference in the packet size between the ant packets and the data packets. The ant packet size is comparatively smaller than the data packet thus, ant packets experiences less transition delay than the data packets. During congestion both the ant packets as well as data packets experience delay due to increase in the queue length on the nodes. In response to this, TCP reduces its data rate meanwhile the system updates the pheromone level along the non-congested path increasing the probability of its use. As the result of reduced data traffic and some of its traffic following the alternative path, the previously congested path gets fewer traffic loads other than the background traffic. In the mean time, the ants traveling through the previously congested path do not experience large queuing delay as before. While reaching the destination the ants re-marks this path as the best path and updates the system accordingly. This causes periodic fluctuation on the selection of the path. In addition to this, such fluctuation on path selection causes fluctuation in the TCP data rate as well as reordering of the packets. As a consequence, TCP never regains its full data rate.

On the other hand, LSRP and DVR never use alternative path unless there is a link failure. Thus, they always experience network congestion and reduce the TCP throughput. The overall TCP throughput on all three systems are compared in the Figure 4.11. The graph shows that the TCP throughput on the CEAS system is not much affected by the ant rates. Which further indicates that the system does not make use of the alternative path in response to the congestion. However, compared to the TCP throughput on LSRP and DVR, the TCP throughput on the CEAS system is *4* times higher in our network. Similar to our previous results the performance of TCP Sack is higher than that of TCP Reno. But, the difference between the performance of both the TCP variants in case of LSRP and DVR is insignificant.

The effect of increasing packet size.

Increasing the packet size increases the transition time of the packet and also decreases the number of packets required to transmit the same amount of data. As explained in Section 2.7, increasing the packet transition delays increases the time interval of sending 3 continuous DUPACKs for missing packets. Consequently, this increases the average delay that the missing packet can spend on the longer path before triggering the *Fast retransmission* and *Fast recovery* mechanism. Additionally, reduce in the packets quantity reduces the number of out-of-order packets proportionally.

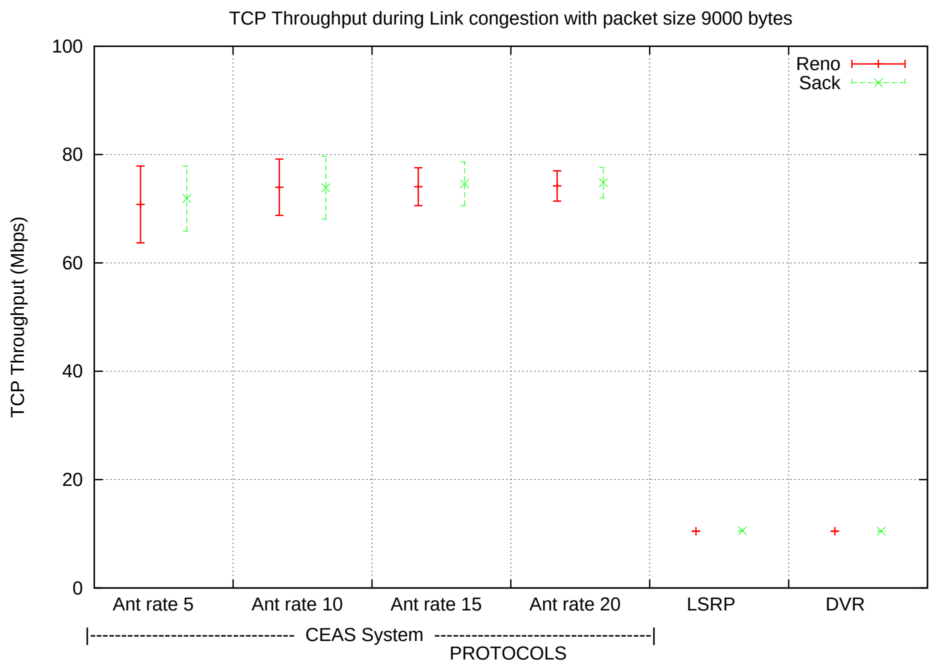

Figure 4.12: TCP Throughput on different system with Large packet size

Therefore, TCP transmits data with higher throughput.

In addition to this, longer time spent by data packets on a link causes ant packets to wait longer time in queue. This adds considerable amount of delay on ant packets when the network is loaded. Taking our previous scenario as an example; when there is congestion along the best path, the ants traveling through the path experience longer queuing delay. In the mean time, ants traveling through the alternative path do not experience such delay. As a consequence, the system diverts data traffic along the alternative path. But this time, the ants traveling through the shortest path still experience queuing delay as some of the data traffic still follow the best path. Further, the UDP background traffic does not reduce its transmission rate in response to the congestion. Besides that, the background traffic load on this link is higher compared to the data traffic along the alternative path. This further increases the queuing delay along the shortest bath. As a result, the system gradually diverts large amount of data through the alternative path which ultimately increases the TCP performance.

The results obtained by increasing the packet size to 9000 bytes are plotted in Figure 4.12. The graph shows significant increase in TCP performance on CEAS system but, the TCP performance on LSRP and DVR show no improvement because they continue to use shortest path and experience congestion.

These results indicate that forcing the ants to experience considerable amount of delay corresponding to the traffic load enhances the TCP performance. This can be achieved easily by setting *packet-priority* on the data packets. The Packet-priority allows a packet with a higher priority to be served before a packet with a lower priority. Thus, at each router the data packets would get higher preference forcing the ant packets to remain longer time in queue. Another possible solution could be piggybacking ant information on the data packets. With such piggybacking the ants would experience the same amount of delay as the data packets.

4.5 The effect of different capacity links on TCP.

In a real network capacity of a link varies according to the network requirements and the capacity planning. In LSRP, the capacity of the links is the major factor influencing the link cost. The best path between two nodes is calculated on the basis of total end-to-end link costs among the available paths. Unlike LSRP, the CEAS system calculates path cost on the basis of the end-to-end delay. Therefore, when the network is not loaded with data traffic, the end-to-end delay depends on the average link transition delay. Since the packet size of the ants is very small, the transition delay on ant packets is negligible compared to data packets. However, the end-to-end delay varies according to the traffic loads. In this section we study how the CEAS system treats the link with unequal capacity during path selection and its effect on TCP performance.

Scenario:

Similar to our previous scenarios, we select link connecting *node 4* and *node 5* as our victim and lower its link capacity by one tenth. Similarly, we simulate the scenario for 1200 seconds at the higher ant rates. Again, we use TCP throughput as the performance metric. The results from both the TCP variants at different ant rates are compared with 95 percent confidence interval.

Results:

The results from these simulations are very similar to the results that we obtain in Section 4.4. During entire period of simulations we observe that the TCP never gain its full rate. (The graphs from the results are plotted in Appendix C.) We also find that the TCP data rate fluctuates continuously as well as the system continues using the lower capacity path more frequently than the alternative path. The reason behind such fluctuation is same as discussed above in Section 4.4; TCP congestion control algorithm and the large difference in the packet size between ant packets and data packets. Similarly this problem could be solved by setting higher Packet-priority on data packets allowing them to be served before ant packets. Also piggybacking ant information on the data packets could be an option.

4.6 Multiple TCP connections.

All of our studies above only deals with a single TCP connection in the network. In this section, we study the behavior of the CEAS system under influence of two TCP connections. Here, we examine how the system handles both the connections to find out:

- Whether the system handles both the TCP connections equally. In other words, we investigate whether the overall TCP throughput of both the TCP connections are similar or one of them dominates the other, and
- Whether both the TCP connections struggle to use the best path or they are diverted along different paths.

Scenario:

To understand the behavior of the system under TCP connection we use select two TCP connections in such a way that they both share the best path along the same direction. For which we establish first TCP connection between *node 0* and *node 8* and the second TCP connection between *node 2* and *node 6*, where *node 0* and *node 2* are the source nodes. We start both the TCP connections at the same time i.e. *100 seconds* after the initialization phase. This time we only simulated the system at an ant rate of *20*. We also simulate the system using larger packet size i.e. *9000 bytes*.

Results and Discussion:

The results obtained from the simulations are listed in Table 4.3. During simulations we observe that non of the TCP connection transmit data at their full rate. The results show that the average TCP throughputs for both the TCP connections on the CEAS systems are almost equal. This means that the system handles both the TCP connection independently with equal priority1. This also means that both the TCP connection experiences congestion in a similar manner.

Parameter	Throughput (TCP Reno)		Throughput (TCP Sack)	
	TCP 1 (standard error)	TCP 2 (standard error)	TCP 1 (standard error)	TCP 2 (standard error)
CEAS (ant rate 20)	40.65 (11.82)	38.81 (12.97)	44.46 (11.23)	40.58 (9.81)
LSRP	44.45	55.01	50.12	47.07
DVR	42.01	55.01	50.12	47.07

Table 4.3: TCP throughput during multiple connections

Figure 4.13, depicts the pheromone distribution at *node 3* towards the *node 4* and *node 10* (*i.e. along the best path and the alternative path*). The graph is plotted using the results from an instance of the simulations. Looking at the pheromone distribution we find that the probability of using the shortest path by both the systems is much higher than the alternative path. While visually examining the simulations we find that

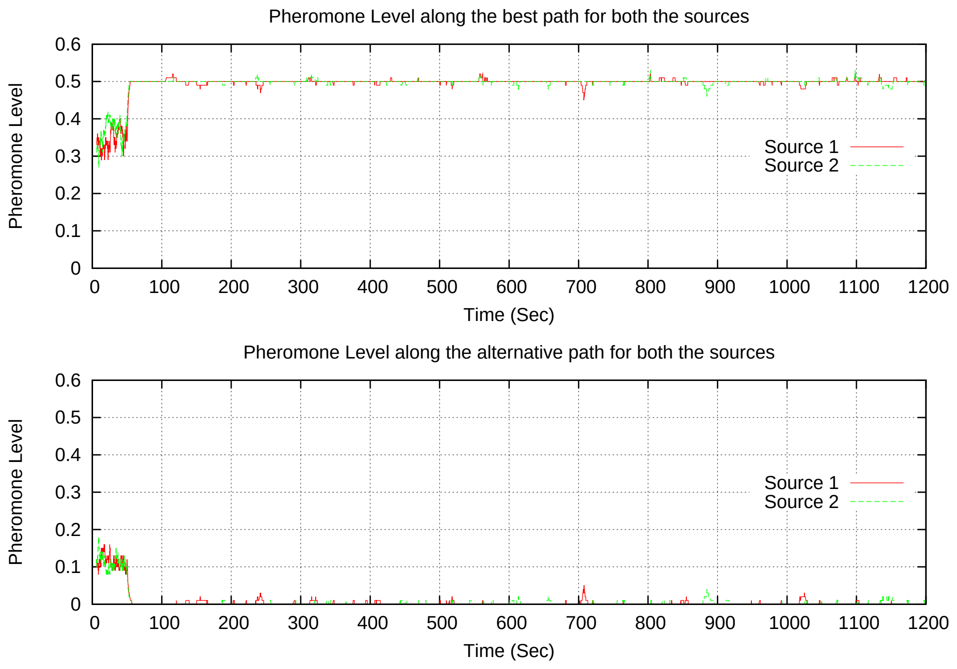

Figure 4.13: Pheromone Distribution at node 3

both the TCP traffic strives to use only the shortest path. During such competition the shortest path gets congested. As a consequence, both the TCP connections experience congestion at the same time and reduce their data rate. The reduction of the data rate on both the connections removes the congestion however, both the connection then again increase their data rate. This causes periodic congestion along the shortest path. Thus the behavior of the system is similar to the one that we observed in Section 4.4. And, the TCP connections neither gain full data rate nor they are diverted separately.

Unlike CEAS, LSRP and DVR both use only the shortest path for both the connections. The TCP connections using TCP Reno in these systems get periodic preference along the best path allowing one TCP stream to transmit at higher rate than the other alternatively (the graphs showing such alternating data rates are plotted in Appendix D). This is the reason why we have the differences in the throughputs of the two TCP connections in case of LSRP and DVR. At the end of our simulation one of the TCP connections is transmitting at higher data rate than the other resulting the difference in overall TCP throughputs.

Parameter	Throughput (TCP Reno)		Throughput (TCP Sack)	
	TCP 1 (standard error)	TCP 2 (standard error)	TCP 1 (standard error)	TCP 2 (standard error)
CEAS (ant rate 20)	74.94 (3.13)	75.03 (2.31)	74.9 (1.3)	75.23 (0.81)
LSRP	50.21	49.12	52.45	47.21
DVR	58.63	40.75	59.66	38.28

Table 4.4: TCP Throughput during multiple connection at packet size of 9000 bytes

The results obtained after increasing the packet size are listed in Table 4.4. Similar to our previous results in Section 4.4, the TCP performance on CEAS system shows significant improvement while the improvements on LSRP and DVR are very small. Looking at the pheromone distribution at *node 3*, depicted in Figure 4.14, we can say that the TCP streams make use of both the paths more frequently. The graph also show that the probability of using the shortest path by one TCP stream is higher than other alternatively. This indicates that the system diverts TCP streams separately. While visually examining the simulations we find that most of the time both the TCP traffic follows separate paths yet the TCP streams compete to use the best path.

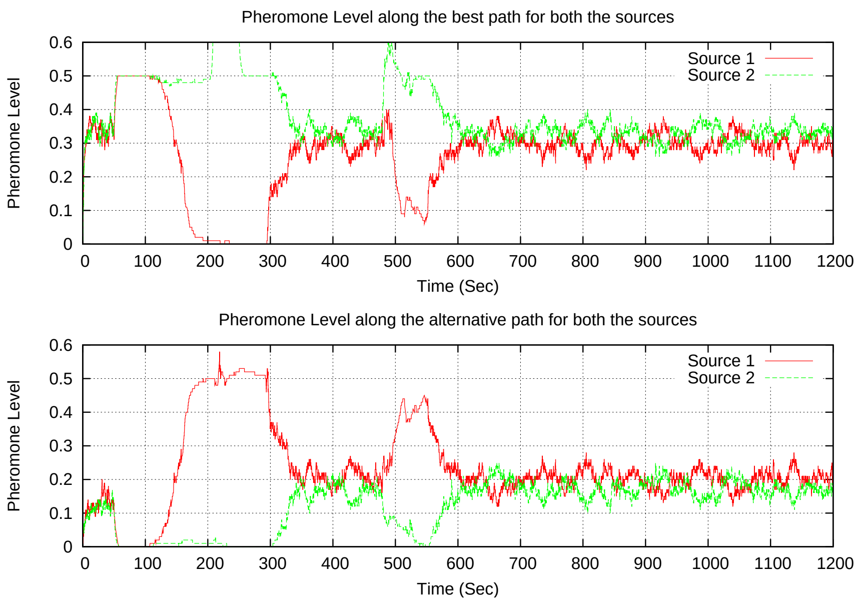

Figure 4.14: Pheromone Distribution at node 3 using packet size of 9000 bytes

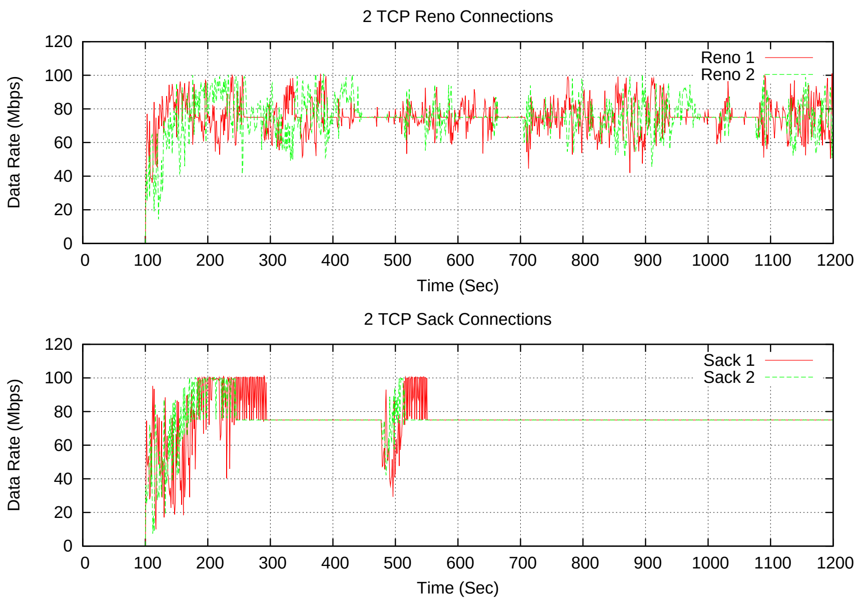

Figure 4.15: Comparison of TCP data rate between TCP Reno and TCP Sack (results from an instance of a simulation)

Figure 4.15 compares the TCP data rate of both the connection using TCP Reno and TCP sack during an entire period of simulations. The graph is plotted using the results from an instance of the simulation after increasing the packet size to 9000 bytes. The results show that the performance of TCP Sack is much smoother than TCP Reno.

4.7 Previous Hop memory

During link failure, we observe that large number of packets continuously loop around *node 3* and *node 4*. Although there is an alternative path available towards destination, the pheromone level toward the failure path still remains higher than the alternative path. Thus at *node 3*, packets are forwarded to *node 4* instead of *node 10*. But at *node 4*, the only available link towards the destination other than route back is broken so the packets are forwarded back to the *node 3*. As a result packets continuously loop around the link and reduces the TCP performance.

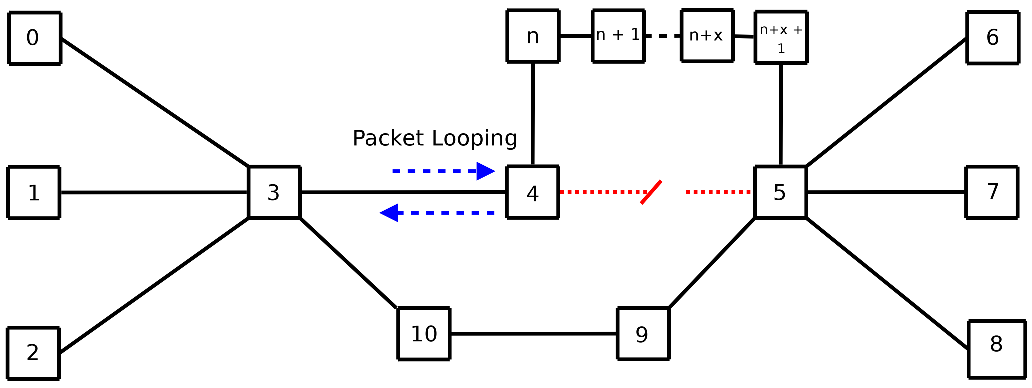

Figure 4.16: Micro-looping during Link failure

Let's assume a topology as shown in Figure 4.16, where we have another path connecting *node 4* and *node 5* via a series of nodes $n + x + 1$. Suppose the link connecting *node 4* and *node 5* is broken as in our previous scenario. At this point, the path cost towards *node 5* via $n + x + 1$ nodes is much higher than the path cost along $[4, 3, 10, 9, 5]$. Thus, packets reaching *node 4* is forwarded back to *node 3* instead of node n. However at *node 3*, the probability of selecting *node 4* over *node 10* still remains higher. As a result the TCP packets keep looping around *node 3* and *node 4* until the probability of selecting *node 10* becomes higher.

To avoid such micro looping we implement a previous hop memory technique. This technique prohibits forwarding packets back to the node from which they are recently forwarded. As in our example from Figure 4.16, once the packet reaches *node 4*, the previous hop memory technique restricts the forwarding of the packet back to the *node 3*. Thus, the packet is forwarded to node n which would then ultimately reach *node 5*. Further as in our scenario where we have only one available path i.e. route back to the *node 3*, if we prohibit forwarding the packet back then the packet would have to be dropped. In such cases, where there is no available path other than route back we allow forwarding back the packet. But as the packet reaches *node 3,* there are other paths available so the packet would not be forwarded again to *node 4*. This ultimately removes the continuous looping.

Results and Discussion:

After implementing a previous hop memory technique, we find that at all ant rates the TCP transmission rate is unaffected by the link failure. TCP resumes data transmission immediately after the failure and continues transmitting data at the same rate. The data packets that are forwarded to *node 4* loop back to *node 3* and continue their journey through the alternative path. However, during pheromone update and re-stabilization process, the system slowly diverts more TCP traffic along the alternative path. As a result TCP data packets follow both the paths; the alternative path and the one with a single loop. The use of multi-path causes reordering of the packets and reduces TCP performance. However, TCP regains its data rate once the re-stabilization phase ends.

The overall percentage gain in TCP throughput after implementing a previous hop memory technique is plotted in Figure 4.17. The graph shows that the system at the lower ant rates benefits more with this technique than the system at higher ant rates. The reason behind such benefit is due to slower pheromone update and re-stabilization

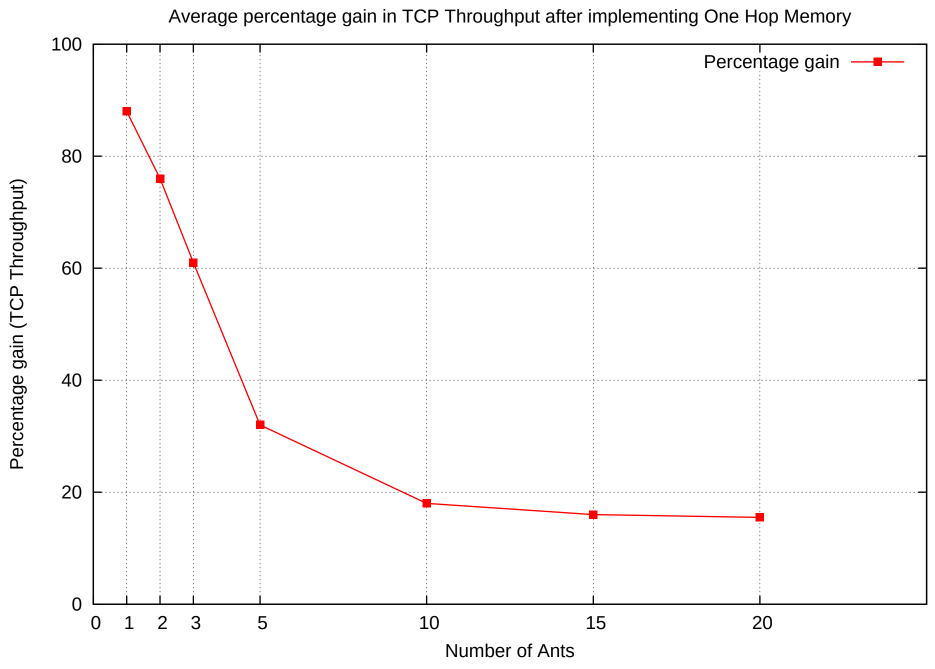

Figure 4.17: average percentage gain in TCP Throughput

process at the lower ant rates. Without the previous hop memory technique large number of TCP packets suffers looping at lower ant rates than at higher ant rates. Thus the overall gain in TCP performance is more at lower ant rates than at higher ant rates.

Conclusion:

From the above results, we find that during pheromone update periods, the use of multiple paths reduces the TCP performance. During link failure along the best path, the system takes certain time to update the pheromone values depending on the ant rates. Further, TCP data transmission is interrupted until the pheromone level along the alternative path rises increasing the probability of its selection. However, the TCP does not regain its data transmission rate immediately. Until the probability of selecting the alternative path over the failure path becomes significantly high enough the TCP continuously experiences packet loss due to micro-looping. Increasing the ant rate reduces such transmission interrupt time yet the micro looping of the packets reduces the TCP performance to some extent.

We find that the use of previous hop memory technique not only removes such micro looping but also helps TCP source to resume data transmission immediately after the link failure. However, the TCP performance reduces again during pheromone re-stabilization period. In addition to this the overall performance gain achieved after implementing previous hop memory technique is higher at the lower ant rates compared at the higher ant rates.

We also find that the CEAS utilizes its available resources more efficiently than the LSRP and DVR according to the network load. However, due to the smaller transition delay of ant packets compared to the data packets we find that the TCP experiences periodic congestion causing its transmission rate to fluctuate. Although the system make use of the alternative path in response to the congestion, the periodic fluctuation of the TCP and the smaller transition delay of ant packets causes the system to continuously use the shortest path. We also find that forcing the ant packets to experience considerable amount of queuing delay according to the traffic load helps ants to better update the system more accurately.

Unlike LSRP and DVR, we find that the CEAS system handles multiple TCP traffic independently with equal priority. However, the TCP streams strive to use the shortest path causing periodic congestion along the shortest. This significantly reduces the performance of the TCP streams. Similarly, we find that forcing the ant packets to experience queuing delay, the system handles multiple TCP streams more efficiently and diverts them along separate paths. However, the TCP streams strive to use the shortest path as far as possible but one TCP stream at a time.

Finally, we find that the performance of TCP Sack is better than the TCP Reno due to its congestion control strategy. Thus, the TCP Sack would be better choice to use in multipath routing such as CEAS.

Chapter 5

Case Study

This chapter presents simulation studies of the TCP performance on the CEAS system under the influence of combination of network events. The results are compared against LSRP and DVR under similar configurations. The studies in this chapter mainly focus on observing the behavior of the TCP in complex networks. The simulations performed in this case study is to understand whether the CEAS behavior under influence of the various network events on complex network produce similar effects on the TCP performance as in the simple network or the results are topology specific.

The case studies are divided on the basis of the topology; Figure 5.1 and Figure 5.12.

5.1 Performance Metrics

- **TCP Connect Time -** The amount of time required to establish a TCP connection from a source to a destination. In other words this is the total time required to deliver first TCP data packet after completing TCP handshaking.

- **TCP Throughput -** The total amount of TCP data transferred after starting a TCP connection over the entire interval of the simulation

- **TCP Goodput -** The total amount of data delivered over the interval of the simulation.

5.2 TCP performance using 12 node network

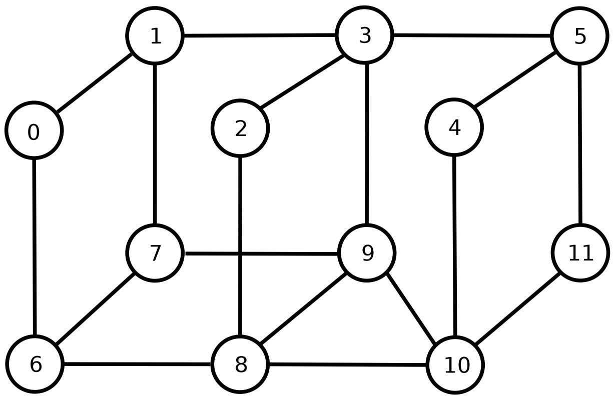

Figure 5.1: 12 nodes network.

This topology is selected to examine the TCP performance on more complex network where we have more than two paths between connection pairs. However, we maintain a regular structure in the topology which would help us understand the system easily and also enable us to visually examine the simulations to gain more insight.

Configurations:

In this study, we use lower capacity links, each of *1 Mb* bandwidth, to connect nodes. In addition to this, we use longer transition delay of *10 milliseconds* on each links. We simulate the CEAS system at ant rate of *10 normal ants per second* and *1 explorer ant per second* with *beta* value of *0.98*. Rest of the CEAS configurations remains similar to our above studies in Chapter 4.

Here, we observe the behavior of the CEAS system under influence of multiple link failures, link re-establishment and frequent transient links in presence of various levels background loads. We study the TCP performance of both TCP Reno and TCP Sack. The TCP performance results are compared with the results obtained from using LSRP and DVR under similar network configurations. The TCP configuration parameters are listed in Table 5.1 and the Table 5.2 lists the configuration parameters for background traffic generators.

Parameter	**Values**
TCP variants	TCP Reno, TCP Sack
TTL	10
Window Size	200
Packet Size	512 bytes
Data Rate	800 Kbps

Table 5.1: TCP configuration

Parameter	**Values**
Traffic	UDP
Source	CBR generator
Packet Size	512 bytes
Data Rates	0 - 5000 Kbps
TTL	10
Number of Connections	5

Table 5.2: Background traffic parameters

5.2.1 TCP performance on steady network

In this section, we measure and compare the TCP performance of all three systems on our network when there are no link failures. The TCP performance is measured on the basis of TCP connect time, TCP Throughput and TCP Goodput. Additionally, we measure the TCP performance at various levels of background traffic. The TCP performance using both TCP Reno and TCP Sack are compared.

Scenario:

We select *node 0* and *node 10* as our TCP source and destination nodes. The source and the destination nodes are chosen in order to have multiple paths between them. To avoid the stabilization phase we start TCP traffic *50 seconds* after the initialization phase. The scenarios are simulated for *700 seconds* with first *50 seconds* as the initialization phase.

We introduce background traffic loads using 5 different UDP traffic generators immediately after the initialization phase. The UDP traffic source and destination nodes are selected in such a way that at least one of the background connections shares a common link with the TCP connection. The combined traffic generated by these UDP connections provides different levels of background traffic loads on the network.

Results and Discussions:

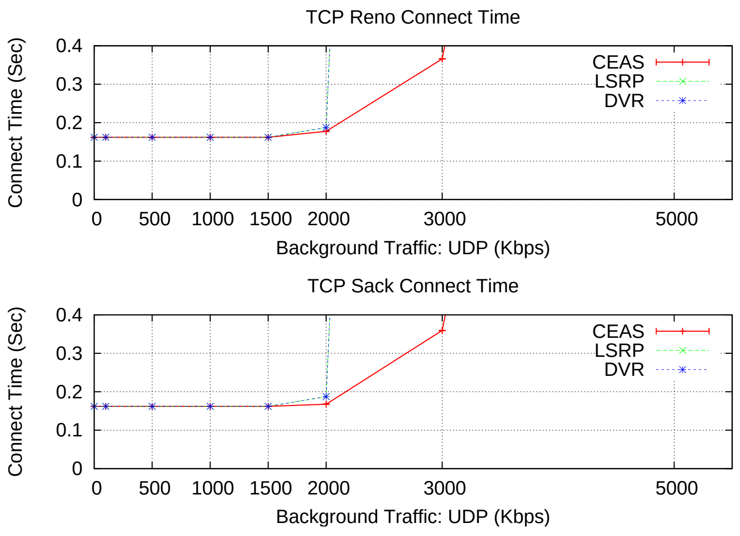

Figure 5.2: TCP Connect Time.

Figure 5.2, compares the connect time on all three systems at various levels of background traffic. The graphs show that the connect time for both the TCP variants are almost similar. Up to the background loads of 1500 Kbps, the connect time remains unchanged on all three systems. Further increasing the background load delays the connect time. However, at 3000 Kbps the connect time on CEAS system takes much less time compared to the other systems. The reason for this is, the background loads on the shortest path is higher compared to the other available paths and the TCP traffic on LSRP and DVR only follows the shortest path while the TCP traffic are forwarded using multiple paths on the CEAS system.

Figure 5.3 and 5.4, compare the TCP throughput and goodput on all three systems

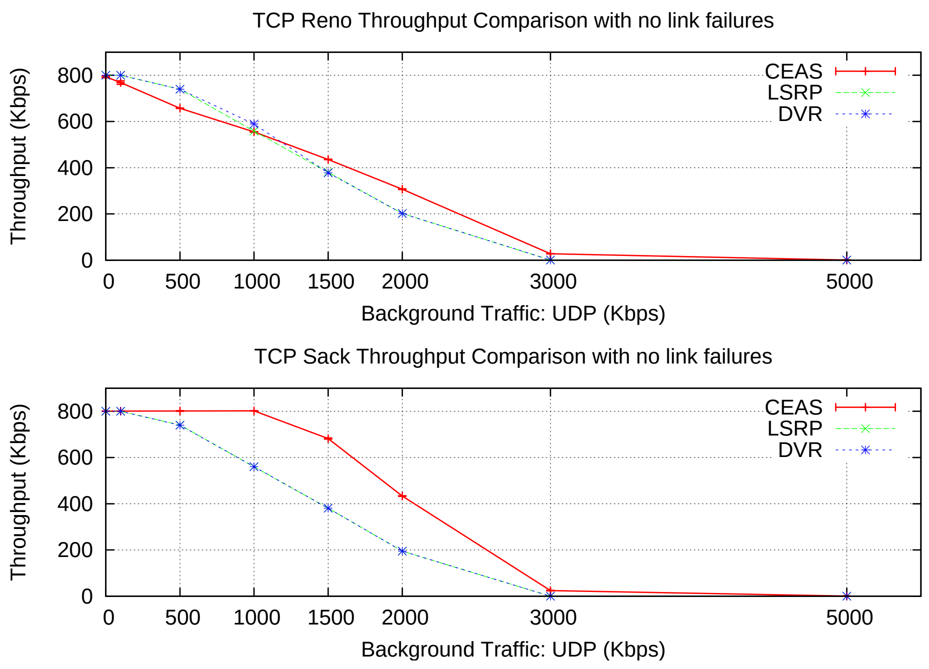

Figure 5.3: TCP Throughput at various levels of background traffic loads

at various levels of background traffic. Both the graphs show that the performance of TCP Sack is better than TCP Reno on CEAS system while the TCP performance of both the variants on the LSRP and DVR remains unchanged. The results show that TCP Sack on CEAS system maintains highest level of goodput up to 1000 Kbps of background loads on the network while the performance of TCP Reno decreases with the increase in the background loads. The results also show that there are very little differences on the TCP throughput and TCP goodput values. This infers that the data retransmission rates on all three systems are very low.

Due to the fact that the background loads on the shortest path is higher compared to the other paths the TCP performance on LSRP and DVR is much lower than on the CEAS system at higher background loads. However, beyond 3000 Kbps of background loads the TCP throughputs on all three system is very negligible. All these results indicates that in presence of lower traffic loads the TCP performance on all three systems are similar. Further increase in the traffic load reduces the TCP performances

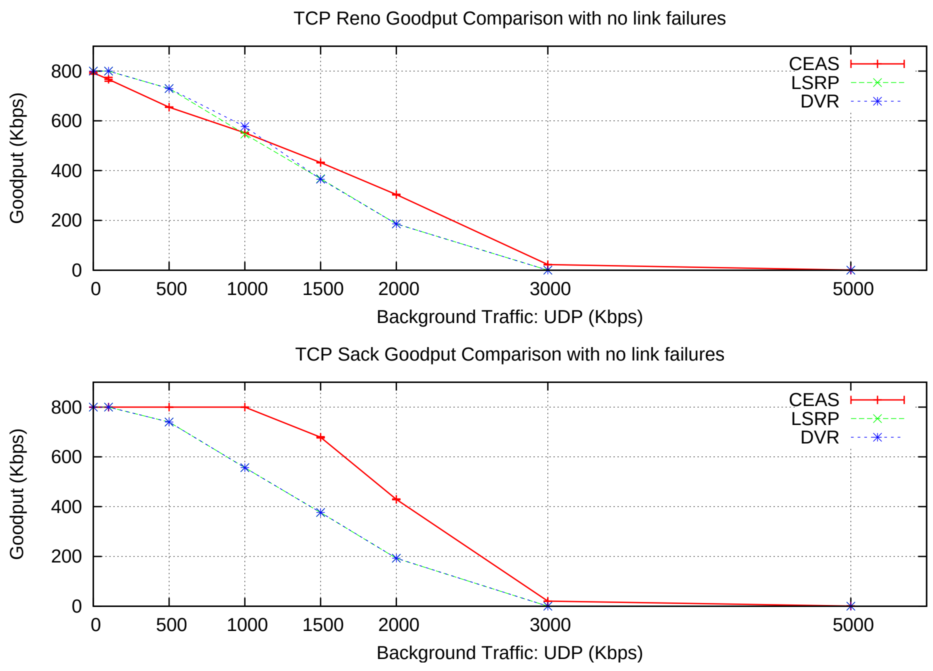

Figure 5.4: TCP Goodput at various levels of background traffic loads

yet CEAS maintains higher TCP performance than others. This indicates that the CEAS system makes better use of its available resources than LSRP and DVR.

5.2.2 TCP performance during multiple link failures

In this section, we measure the TCP performance of all three systems during multiple link failures. Similar to our above study we use TCP Throughput and TCP Goodput as our performance metrics. Similarly, we measure the TCP performance during link failures at various levels of background traffic. The performances of all three systems are compared using both the TCP variants.

Scenario:

Similar to our scenario in 5.2.1, we select *node 0* and *node 10* as our TCP source and destination nodes. The simulation parameters, TCP parameters and UDP background traffic loads are configured identically. We schedule first link failure 100 seconds after

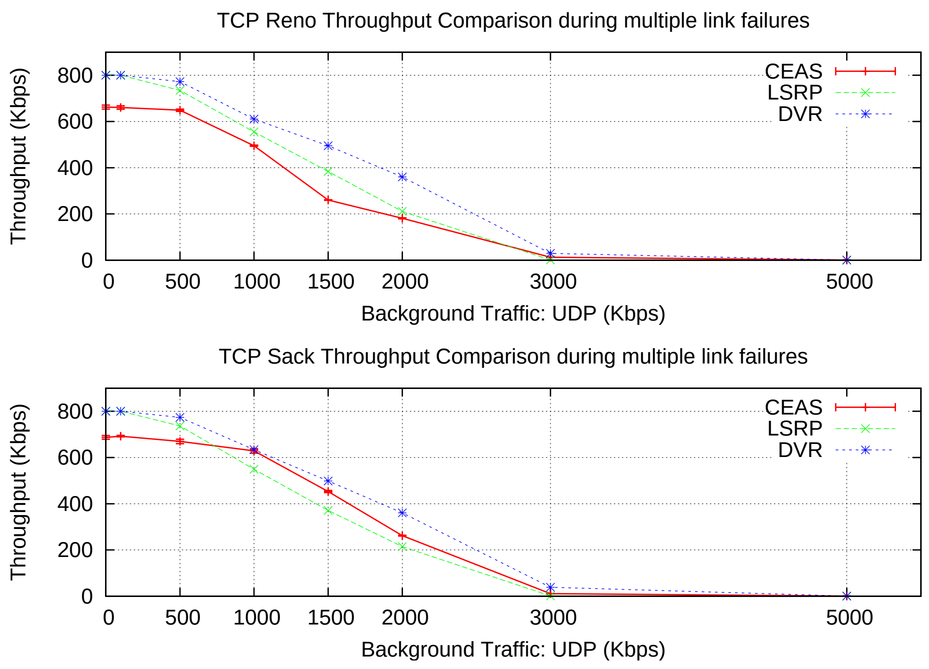

Figure 5.5: TCP Throughput during multiple links failure at various levels of background traffic loads

starting TCP traffic. The first failure is scheduled along the shortest path between the TCP connection pair so that the failure affects all three systems. The next failure is scheduled along the next shortest path and further failures are selected among the equal cost paths. However, we re-establish the failure links along the second shortest path and the shortest path after some time. All the events are at least 50 seconds apart where as the time period between a particular link failure and its re-establishment is scheduled 350 seconds apart. The scenario is simulated for both the TCP variants at various levels of background traffic loads.

Results and Discussions:

Figure 5.5 and 5.6, compare the TCP throughput and TCP goodput on all three systems at various levels of background traffic when there are multiple link failures on the network. Similar to the previous results, the graphs show that the performance of TCP Sack is better than TCP Reno on CEAS system while the difference between the

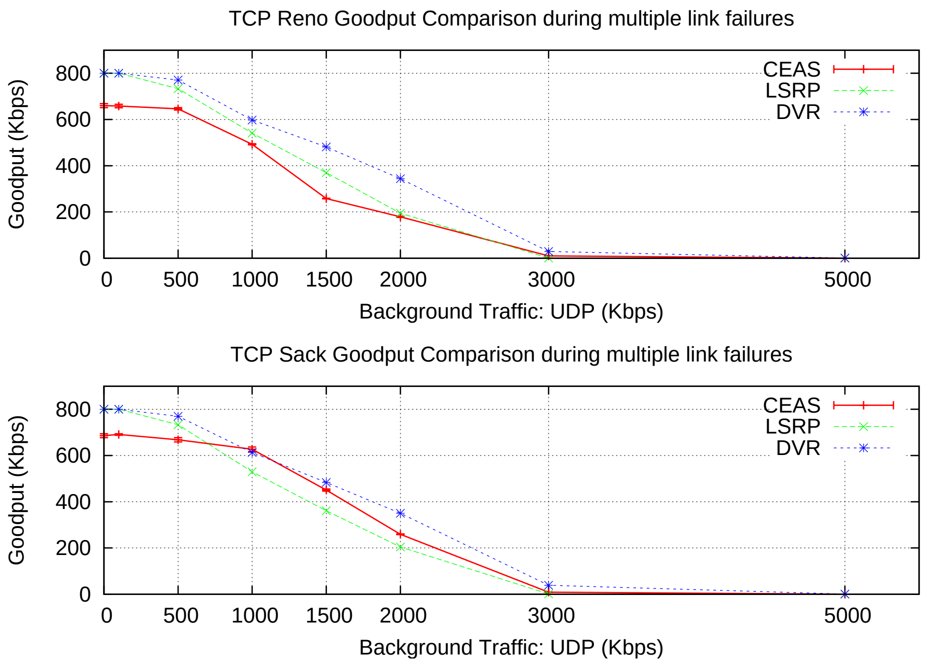

Figure 5.6: TCP Goodput during multiple links failure at various levels of background traffic loads

TCP performance of both the TCP variants on other systems are insignificant.

The lower performance of TCP on CEAS system compared to other system is due to the time it takes to update the pheromone level and divert the data traffic along other available path. Unlike LSRP and DVR the selection of the next shortest path on CEAS takes significant amount of time depending on the ant rate. In the mean time, large amount of packets are delivered out-of-order reducing the TCP performance. This has been discussed in *Section 4.3.*

Figure 5.7 gives an overview of data re-transfer in all three systems during multiple link failures at *1000 Kbps* of background traffic. (*The graph is plotted using a result from an instance of the simulation at 1000 Kbps of background traffic.*) The figure shows that the data retransmission on the CEAS system is more frequent than LSRP and DVR. This indicates that the large number of packets are delivered out-of-order on CEAS system. While visually examining the simulations we find that the CEAS system diverts the traffic along multiple paths after the link failure. In addition to this,

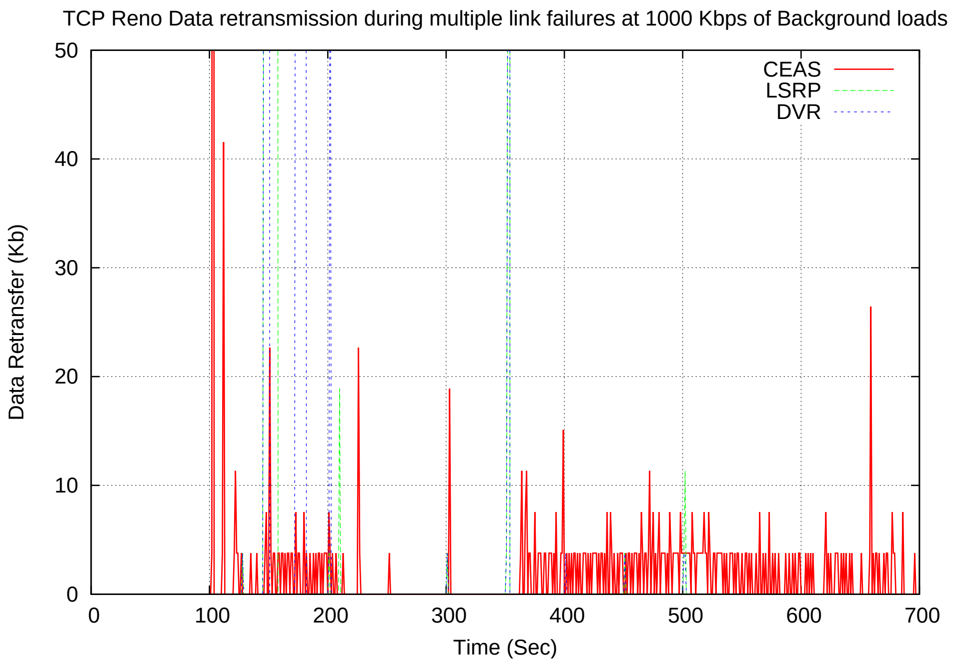

Figure 5.7: TCP data retransmission during multiple links failure at 1000 Kbps of background traffic loads

there are multiple packet losses due to micro-looping. *(see Section 4.3 for detail.)*

Figure 5.8 compares the improvement achieved on TCP Goodput after implementing previous hop memory technique. The graphs show slight improvement on TCP goodput at all levels of background traffic. However, the difference in performance would have been larger at lower ant rates.

5.2.3 TCP performance during frequent transient links

In this section, we measure the TCP performance of all three systems on the network with unstable links (frequent transient links). The TCP Throughput and TCP Goodput of all three systems are measured and compared at various levels of background traffic.

Scenario:

To simulate an unstable behavior of links, we flap links up and down continuously

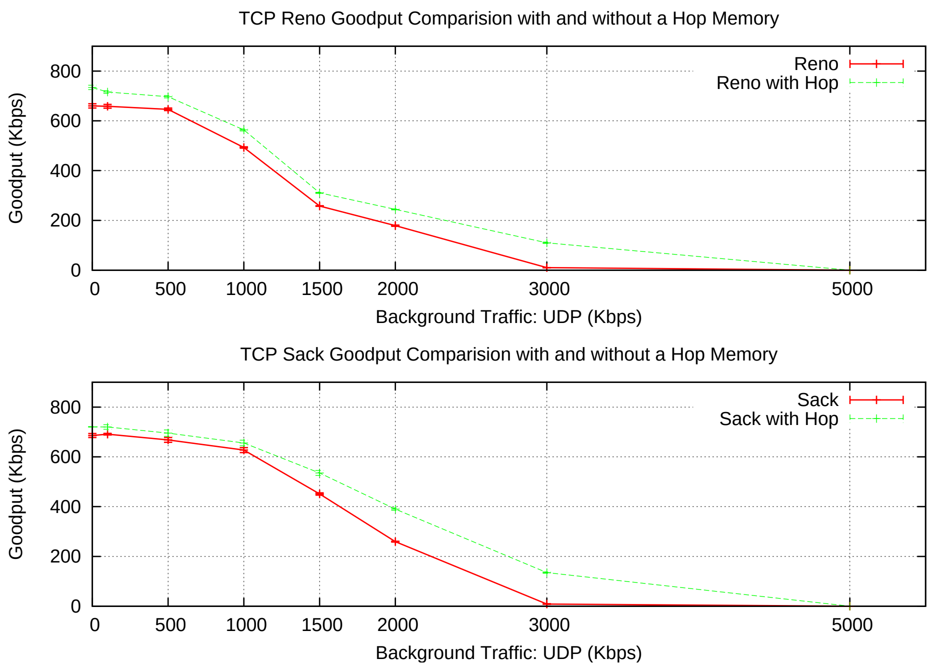

Figure 5.8: TCP performance after using one hop memory technique.

at the frequency of less than a second. Each flap period last for 2 seconds and the next flap period begins 2 seconds after the end of previous period. For this scenario, *node 7* and *node 10* are selected as our TCP source and destination nodes. The simulation parameters, TCP parameters and the background traffic loads are configured identical to above scenarios. The cyclic link flapping continues throughout the simulation period. A link along the shortest path and a link along the equal cost path are selected as our unstable links. The equal cost paths are among the next shortest path. The two links flap up and down at frequency of 0.3 second and 0.5 second.

Results and discussions:

Figure 5.9 and 5.10, compares the TCP throughput and TCP goodput on all three systems at various levels of background loads in presence of unstable links on the network. The results show that the TCP performance on the CEAS system is significantly better than the LSRP and DVR. The LSRP and DVR both try to route traffic through

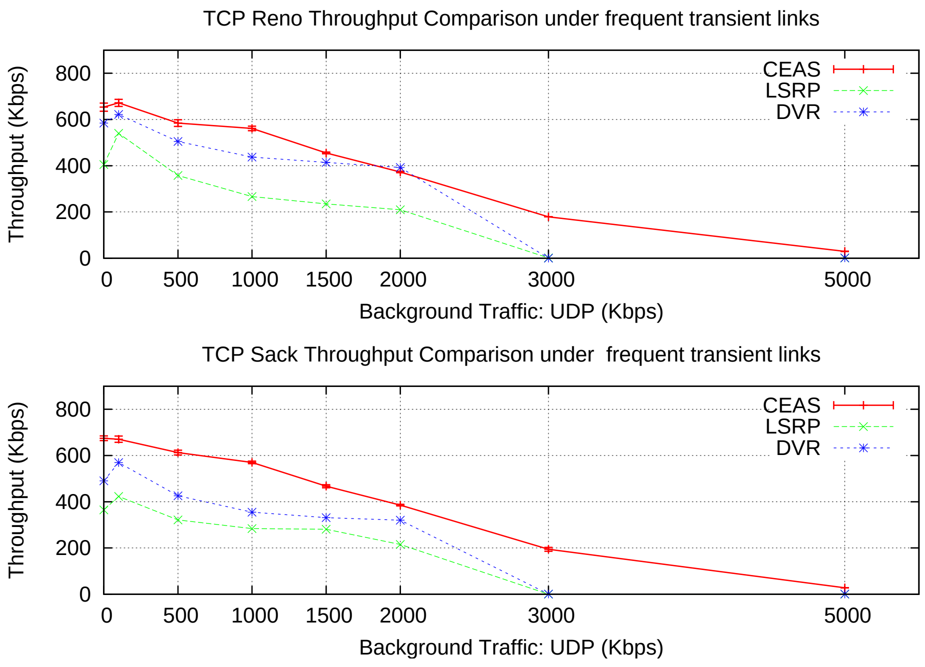

Figure 5.9: TCP Throughput under frequent transient links at various levels of background traffic loads

the shortest path thus during flapping large amount of data packets are dropped. Further the situation is worsen by the flapping behavior of both the shortest path and the second shortest path at different time interval. The LSRP and DVR continuously struggle to re-route the data traffic along the shortest path which causes huge amount of packet loss.

On the other hand, the higher performance of TCP on CEAS is due to the use of stable paths more frequent than the shortest path. While visually examining the simulations we find that the CEAS system gradually avoids the unstable links and diverts most of the traffic via reliable links. We find that the LSRP suffers the most among the three systems. During link flapping large amount of LSA messages are generated to reflect the change in the topology which requires extra over head of updating the routing table and re-calculating the shortest path. But, the DVR only updates the information along the shortest path available which reduces the overhead compared to the LSRP. In addition to this, the 2 second period update of routing tables in DVR

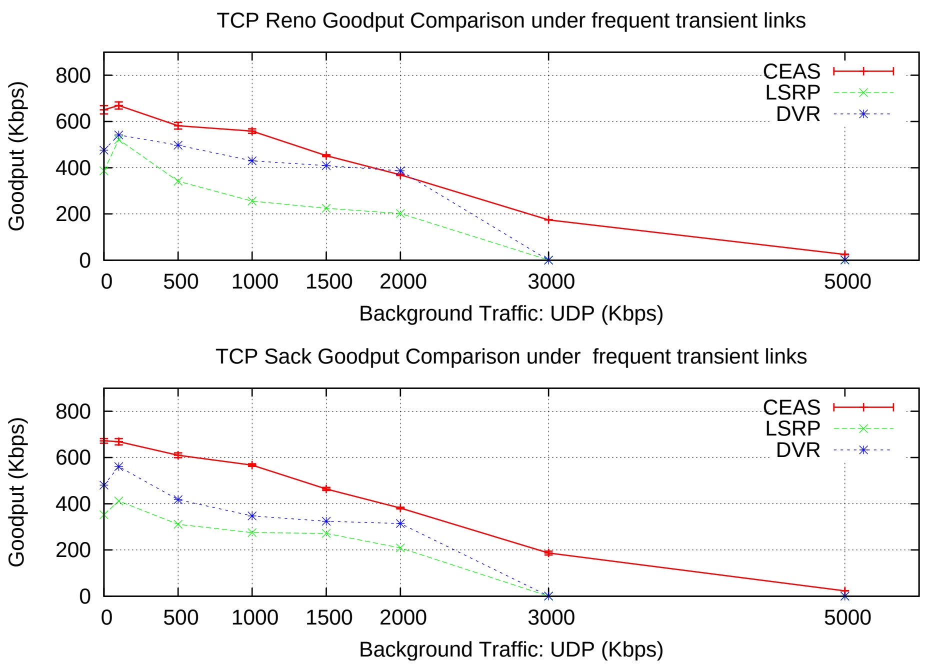

Figure 5.10: TCP Goodput under frequent transient links at various levels of background traffic loads

further helps maintain the system more accurately than LSRP. However, if the links on the network were of different link cost LSRP would have performed better than DVR as DVR only uses hop count to calculate shortest path.

Figure 5.11 shows the slight improvement achieved on TCP Goodput after implementing previous hop memory technique.

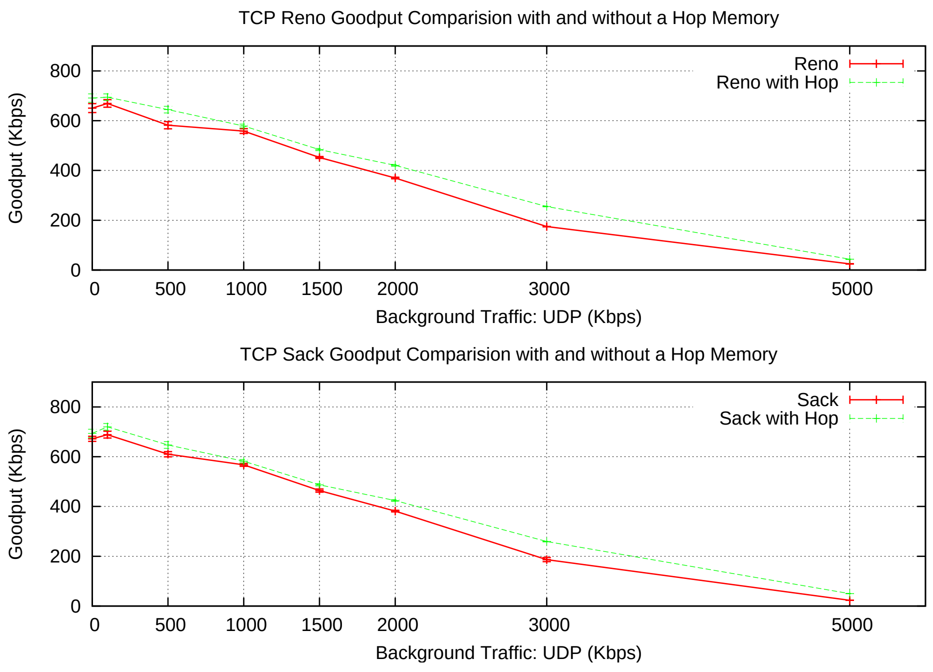

Figure 5.11: TCP performance after using one hop memory technique.

5.3 TCP Performance on large Topology

In this section we carry out simulation study based on a 58 node network topology extracted from Uninett[1] network. The network consists of 12 routers connected using fifteen 2.5 Gbps links. The remaining routers are connected using lower capacity links ranging from 10 Mbps to 1 Gbps. The topology is illustrated in Figure 5.12. This topology is chosen to provide realistic settings for our study. The studies in this section are mainly focused on observing the performance of TCP during different network events and verify the behavior of TCP with the results from the studies above. (The geographic location of the nodes are listed in Appendix E)

Scenario:

In a real world the bulk TCP data transfer is not very common. Further, the link failure events are also very rare except during the scheduled maintenance period. However, a study by Markopoulou et al. [20] shows that the link failures other than scheduled maintenance occur often on a network. The study shows that most of the failures occurs simultaneously due to router related problems or link related problems. In addition to this, there may be other individual link failures. Some of the individual link failures last for longer period of time while the other may last for very short interval but the frequency of such short link failures may be quite high.

Assuming a scenario of scheduled data backup which involves large TCP data transfer we selected 4 TCP connections on different parts of the network. We select a common destination on all four connections assuming it to be the central server. Also, assuming the worst case scenario we created some link long term link failures on the network as well as high frequency short term link failures. We schedule these high frequency short term link failures in such a way that the link remains disconnected for 3 to 7 seconds and the next failure occurs 1 to 3 seconds after the re-establishment. We also introduce some background traffic around one TCP source. All the network events

[1]www.uninett.no

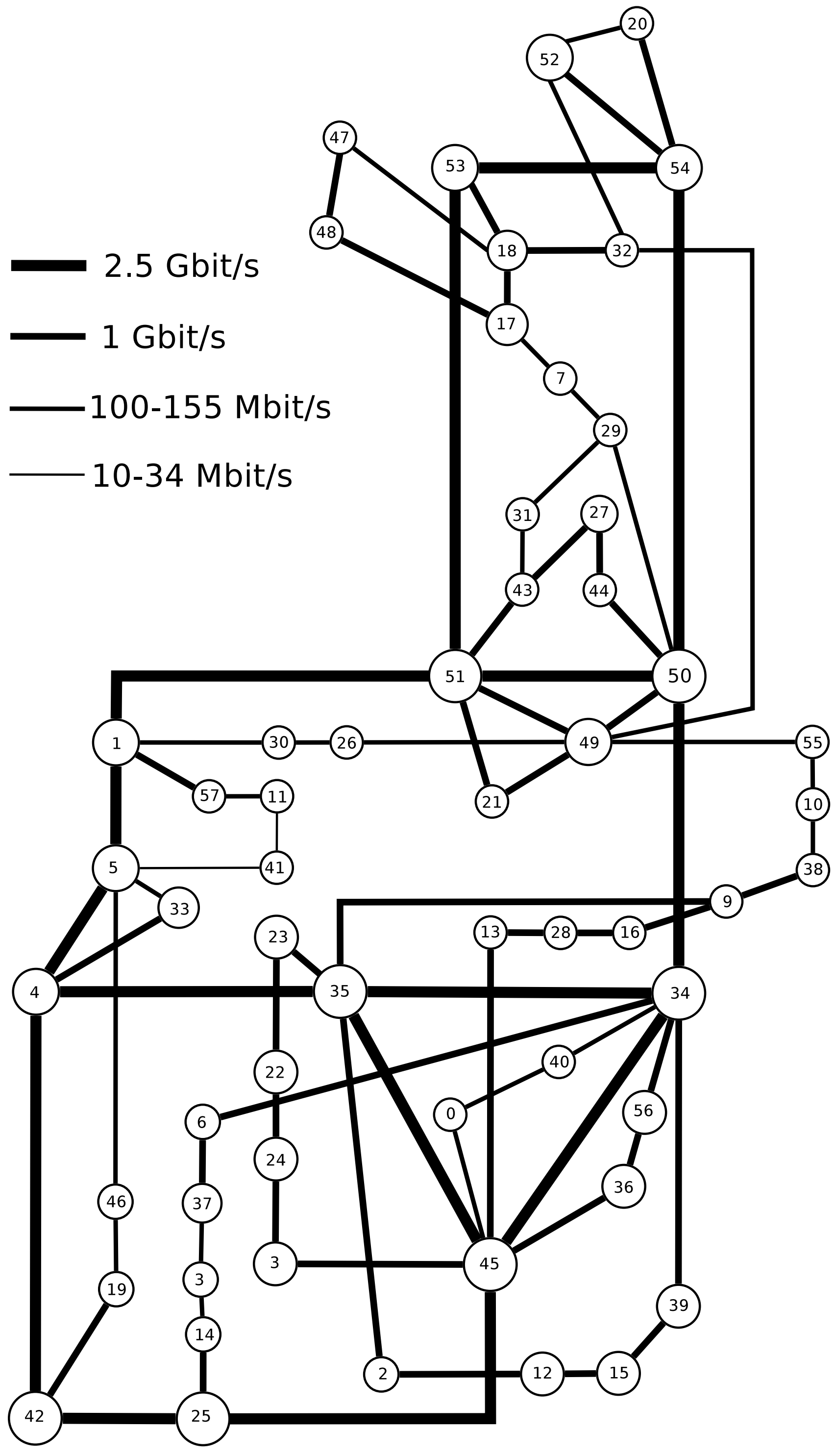

Figure 5.12: Uninett Topology Oct 2007.

are selected in such a way that each of the event effects at least one TCP connection.

We select *node 51*, connected with high capacity links to its neighboring nodes, as our central server. We start all the TCP connections at the same time i.e. 50 seconds after the initialization phase. We simulate the scenario for *1800 seconds* (30 minutes) with first *50 seconds* as the initialization phase. We use ant rate of *20 normal ants per second* and *2 explorer ants per second* with *beta* value of *0.98* for all TCP connections. The packet size used is *1500 bytes* with TTL value of *30*. For this study all the TCP connections use only TCP Sack with delayed ACK.

First TCP connection pair:

We select *node 4* as our first TCP source. *Node 4* has multiple high capacity paths towards the destination. We introduce frequent transient link failures along the shortest path (*link connecting node 5 and node 1*) and a long term link failure along the second shortest path (*link connecting node 34 and node 35*). These events are scheduled *600 seconds* apart. For this connection we use data rate of *2 Gbps* as most of its paths consist of high capacity links.

Second TCP connection pair:

We select *node 12* as our second TCP source. *Node 12* is connected to its neighboring nodes with *1 Gbps* links. We introduce background CBR UDP traffic of *800 Mbps* along its shortest path (*i.e. between node 12 and node 35*). We use TTL value of *2* so that the TCP experiences congestion only along that particular path. The UDP background traffic uses packet size of *1500 bytes*. The TCP data is generated at data rate of *800 Mbps*.

Third TCP Connection pair:

We select *node 41* as our third TCP source. *Node 41* is connected to its neighboring nodes with low capacity links (*32 Mbps and 34 Mbps*). This connection shares its shortest path with the first TCP connection. Thus, the flapping of link connecting *node 5* and *node 1* affects both of the TCP connections. The TCP source generates data at the rate of *30 Mbps*.

Fourth TCP Connection pair:

We select *node 47* as our fourth TCP source. Although *node 47* is connected with *1 Gbps* links to its neighbors, the composite link capacities along the paths varies. For instance; paths *[47,17,18,53,51]* and *[47,48,18,53,51]* consist of equal numbers of intermediate nodes but all the nodes along the first path are connected by *1 Gbps* links where as the link connecting *node 48* and *node 18* along the second path is only of *155 Mbps* bandwidth. The TCP source generates data at the rate of *200 Mbps*.

Results and Discussions:

First TCP Connection:

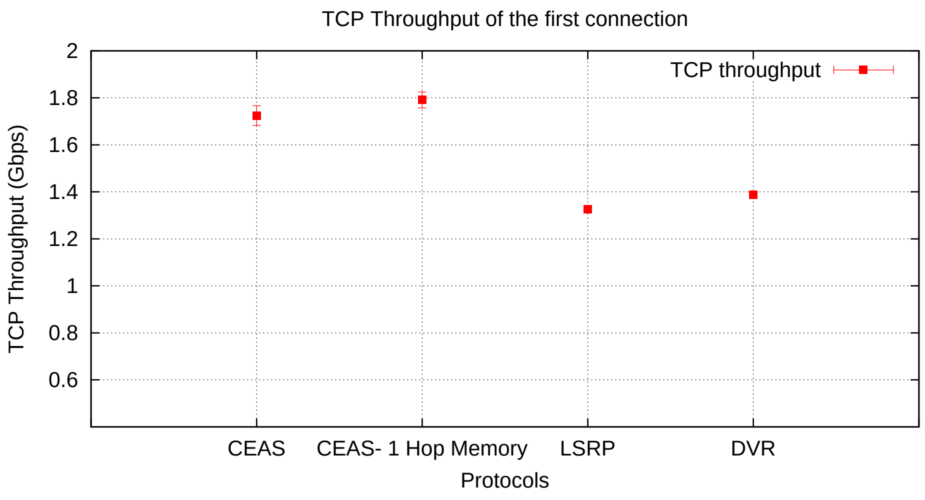

Figure 5.13: TCP Throughput of first TCP connection

Figure 5.13 compares the TCP throughput of the first TCP connection on all three systems. *(The figure also compares the TCP throughput on the CEAS system after implementing one hop memory technique.)* The results show that the LSRP and DVR are more severely affected by the frequent transient behavior of the link. Looking at the data transmission graph, depicted in Figure 5.14, we find that TCP gradually increases its transmission rate some time after the transient behavior starts. This indicates that the CEAS system makes use of alternative path to divert the TCP traffic. However, the fluctuation in the graph also indicates that the CEAS do not completely avoid the unstable link. The gradual increase in the data rate suggests that the considerable amounts of data are being diverted along the alternative path. Over time, the range of the fluctuation decreases which suggested that the use of alternative path is more often.

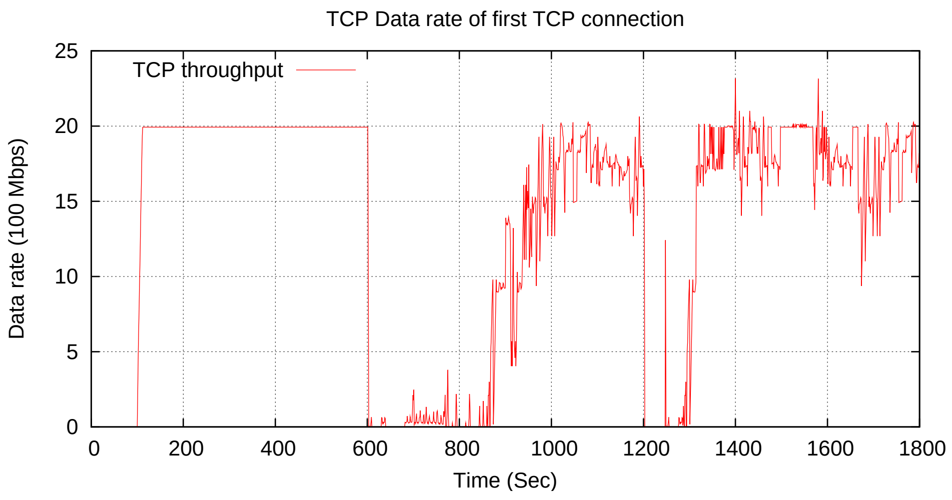

Figure 5.14: TCP data transmission of first TCP connection

The failure of the link along the second shortest path, at time 1200 sec, caused some interrupt in the data transmission. However, the TCP connection resumes after some time yet the continuous fluctuation indicates that the system continuously uses multiple paths.

The TCP performance does not show much improvement after implementing previous hop memory technique. The reason is due to the higher ant rates. At higher ant rates the system updates quickly. (see Section 4.7 for detail)

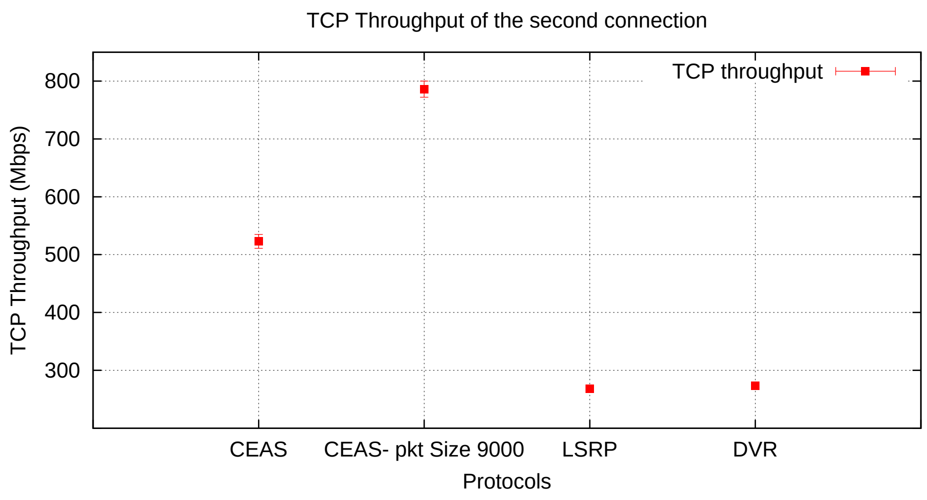

Figure 5.15: TCP Throughput of second TCP connection

Second TCP connection:

Figure 5.15 compares the TCP throughput of the second TCP connection on all three systems. The figure also shows the results obtained after increasing the packet size. The results show that all three systems are affected by the congestion along the shortest path. However, the higher TCP throughput on the CEAS system compared to the LSRP and DVR suggests that the CEAS system uses alternative path to divert some of the traffic during congestion.

The overall TCP throughput is more than half of its full data rate. This indicates that the also in this network the TCP connection experiences periodic congestion and continuously diverts the data traffic along multiple paths. Similarly we find that the TCP performance on CEAS increases significantly after increasing the packet size to 9000 bytes. This shows that the large difference in the packet size of ant packet and data packet is crucial factor affecting the TCP performance during congestion. (*see Section 4.4 for detail.*)

Third TCP Connection:

Figure 5.16 compares the TCP throughput of the third TCP connection on all

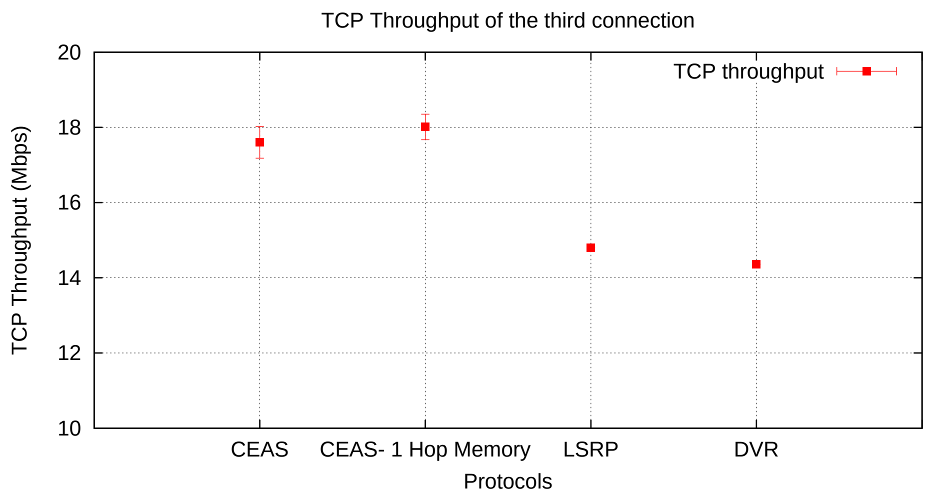

Figure 5.16: TCP Throughput of third TCP connection

three systems. Similar to the results from first TCP connection, the results show that the frequent transient behavior of the link along the shortest path affects all three systems. And the one hop memory technique does not improve the TCP performance significantly.

Visually examining the simulations we find that some of the packets suffer looping around cyclic path during link flapping. Although previous hop memory technique prevents looping around a link it is not adequate enough to prevent looping around a cyclic path. During link flapping, we find that some amount of packets loops around *node 5*, *node 4* and *node 33*. At *node 4*, the previous hop memory technique prevents forwarding packets back to the *node 5*. *Node 4* then selects *node 35* as the next node however, there is the probability of selecting *node 33* as the next node which causes such looping.

These loopings could be avoided by using Route Record option of IP packet header that allows to record IP addresses of up to 9 first visited routers. However, in larger network such looping may occur after the first 9 nodes. Another solution could be to include a list in each data packet that records all the addresses of the nodes visited by the packets. However, processing such list could increase the overhead at each node.

Further, at higher ant rates we find only slight improvement on the TCP performance after implementing previous hop memory. Thus, implementing such list could be more costly.

Fourth TCP connection:

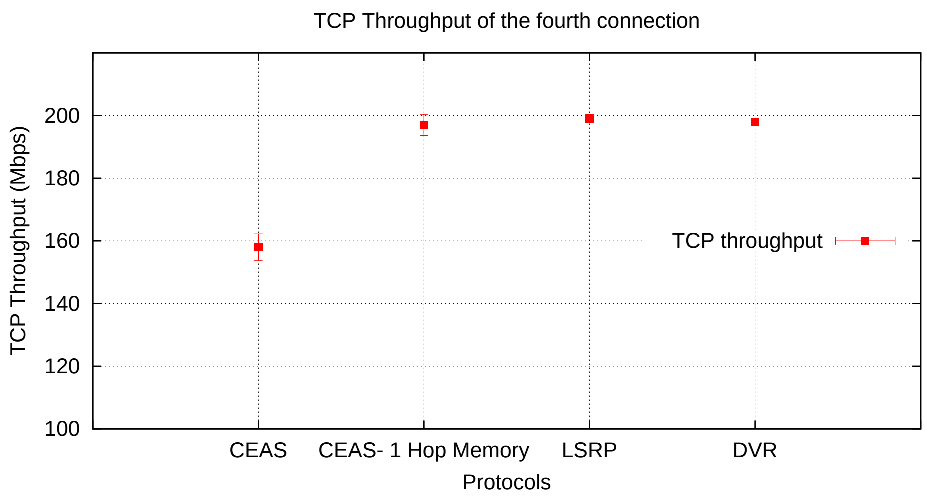

Figure 5.17: TCP Throughput of fourth TCP connection

Figure 5.17 compares the TCP throughput of the fourth TCP connection on all three systems. The unequal link capacities along the paths caused the TCP to reduce its throughput. For this TCP connection there are two paths [47,17,18,53,51] and [47,48,18,53,51] with equal number of intermediate nodes. The link capacity along the second path is much lower compared to the first. Due to the smaller packet size of the ant packets the difference in the end-to-end delay experience by the ants along the paths are insignificant. Thus, the CEAS system uses both the paths and experiences periodic congestion along the second path.

The results also show that the LSRP and the DVR are not affected by such differences in link capacities. The link cost metric associated with each link on the network drives TCP data packet along the best path rather than the shortest path in case of the LSRP. However, in case of DVR, both the paths consist of equal number of hops towards the destination. The results show that the TCP source router chooses the next router along the first path rather than the second path. This might not always be the case.

The result after increasing the packet size to 9000 bytes shows significant improvement on TCP performance. This again shows that the smaller transition delays experienced by ant packets are the major factor affecting the TCP performance.

Conclusion

The behavior of the TCP during different network events in these case studies shows similarity with the behavior that we observe in our basic studies. We find that TCP suffers from micro looping during link failures in all 3 topologies. Although such loopings are removed by the previous hop memory technique, it is not adequate enough to prevent looping on circular paths. Further, TCP suffers from out-of-order packet delivery during pheromone re-stabilization period. In addition to this, the overall performance gain with this technique is very little at higher ant rates.

Although TCP performance during network congestion is higher in CEAS system than on LSRP and DVR, CEAS do not completely divert the TCP traffic along the alternative path. The major cause is the smaller delay experienced by the ant packets compared to the data packets. Therefore, forcing the ant packets to experience delay according to the network load updates the system accurately and increases the utilization of the network resources. As a result TCP performance is enhanced.

Chapter 6

Conclusion and Future Works

6.1 Concluding remarks

In this work, we study the TCP friendliness of the Cross Entropy Ants System in comparison with the standard Link State Routing protocol and the Distance Vector Routing. The CEAS is an adaptive, robust and distributed routing and management system based on the swarm intelligence. This work is more focused on analyzing the behavior of CEAS during different types of network events and investigating the corresponding effects on the TCP performance.

We study the TCP performance using two TCP variants; TCP Reno and TCP Sack on the CEAS network. The study is carried out in two phases: First using a simple network we try to find out how does the TCP behavior varies during different events. In the second phase, we use complex networks to measure the TCP performance during combination of the events. The results from the first phase are used as references to reason out the behavior of the TCP in complex network. The TCP performance are compared with the results obtained from the LSRP and the DVR under similar configurations. All the studies are performed using simulations. Each simulation is replicated 15 times with different seed and the results are synchronized to calculate 95 percent confidence interval.

We find that the CEAS takes considerable amount of time to establish the best path depending on the ant rate. Unlike CEAS, the LSRP and the DVR establish their shortest path immediately after their initialization phase. Thus, LSRP and DVR are immediately ready to transmit TCP data at full rate while CEAS takes some time before transmitting TCP data at full rate.

Similarly, the update process in response to change in network topology is slower in CEAS compared to the others. Such longer update process interrupts the TCP transmission as well as reduces the TCP performance. Unlike CEAS, the TCP transmission in LSRP and DVR resumes at full rate immediately after the changes occur. Increasing ant rate decreases the pheromone update and stabilization period however this increases the overhead of processing ants when the network is stable.

During link failures we observe micro-looping of data packets. These looping reduce TCP performance significantly. Such micro-loops are removed by using the previous hop memory technique. This technique not only removes such loops but also helps TCP resume data transfer at full rate immediately after the link failure. However, this technique is not adequate enough to prevent looping around circular paths.

Although the CEAS shows slower response to the network dynamics, it manages network resources far better than the other systems. The higher TCP performance on CEAS compared to the other system during congestion and load sharing clearly indicates that the CEAS updates its resources on the basis of the quality of the network. We also find that the CEAS handles multiple TCP stream independently with equal priority. But the smaller transition delay on ants compared to the data packet reduces the TCP performance however the overall TCP performance is comparatively higher on the CEAS system. Further, forcing the ants to experience longer queuing delay according to the traffic load helps CEAS update and manage resources more accurately resulting higher TCP performance.

Likewise, the results obtained from our case studies shows similarity in the TCP performance and CEAS behavior. This further indicates that the system manages its resource and handles TCP streams on the basis of the quality of the network. We also find that the performance of TCP Sack is better than TCP Reno in all cases.

In brief, The TCP performance on CEAS system during changes in the network dynamics entirely depends on the ant rates. However, the system adjust its resources according to the quality of the network to provide better TCP performance.

6.2 Future works

In this study, we find that considerable amounts of ants are required during pheromone update and re-stabilization periods. Increasing the ant rate decreases the stabilization periods. However, this increases the overhead of processing ants when the network remains stable. In order to adapt the ant rates according to the network state, the *self-tuned refresh rate* strategies have been proposed in [13] which continuously monitors parameters and ant rates to detect state changes and adapt the ant rate according. Thus, the future work include the measurement of TCP performance after implementing such strategies.

We also find that the re-ordering of the TCP packets is due to the use of multiple paths and not due to packet loss. Thus, it would be interesting to study the TCP performance using a different version of TCP known as TCP PR [5] proposed to maintain TCP throughput TCP throughput during packet re-ordering. Unlike other TCP variants, TCP PR does not rely on DUPACKs to detect a packet loss instead it relies solely on retransmission time out. Thus, TCP PR simply ignores packet reordering and continues transmission at full rate.

We observe that forcing the ants to experience considerable amount of delay according to the traffic load helps the system update more accurately. Thus, the future work include study of the CEAS behavior and the TCP performance using packet priority on data packet. The packet priority on data packet would force the ant packets to wait longer time queuing on each router depending on the traffic loads. Another study could include piggybacking the ant information on the data packet. Such piggybacking causes the ants to experience same amount of delay as the data traffic which would help the system update more accurately.

It might also be interesting to study the TCP performance on CEAS using the *subpath [19]* method. Using subpath method, the forward ants not only record the addresses of the nodes it passes by but also record each link cost along the path. Thus reaching at the destination, the reduction of the cyclic path would also reduce the link cost around those cycles which helps system to update pheromone level faster and more accurately during change in the network dynamics.

Appendix A

Multi-* Network NS2 extension

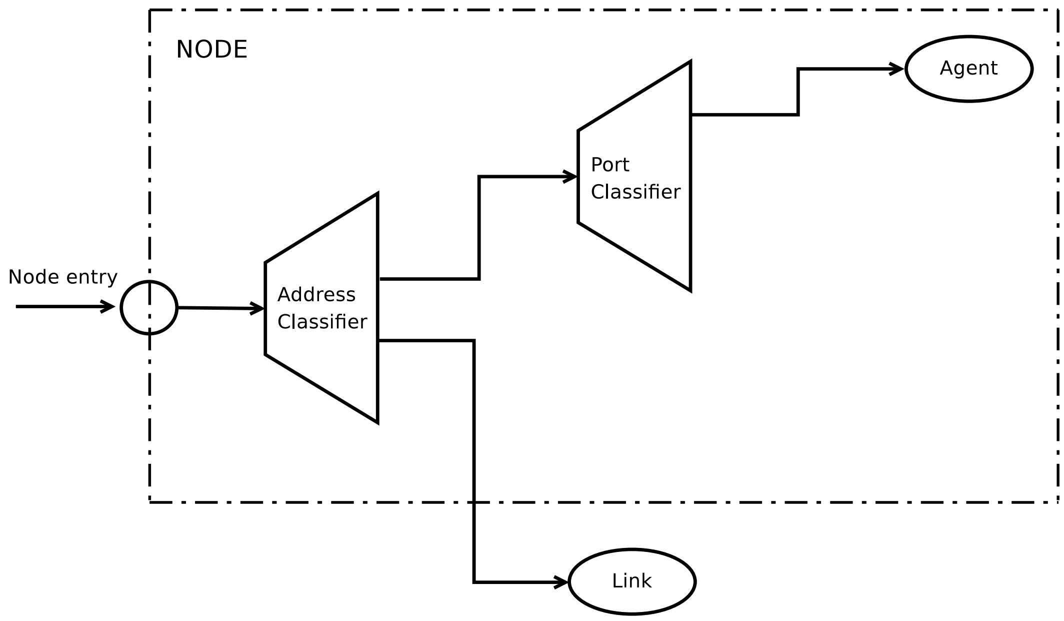

Figure A.1: NS2 Node layout

Typically a node in NS2 consists of two Tcl objects: an address classifier and a port classifier. These classifiers are responsible for distributing incoming packets either to a corresponding agent or to an outgoing link [8]. A typical unicast layout of a Node is shown in Figure A.1. The extension provided by Paquereau et al. [26, 27] adds two new objects; NetworkLayerManager and NetworkLayerUnit, to a node as shown in Figure

A.2. Here, NetworkInterface2 is a generic network interface object that replaces the existing NetworkInterface object of the default NS2. A number of NetworkInterface2 objects can be attached to a node each with a unique identifier [25]. These objects label each packet passing through them with their interface identifier [8].

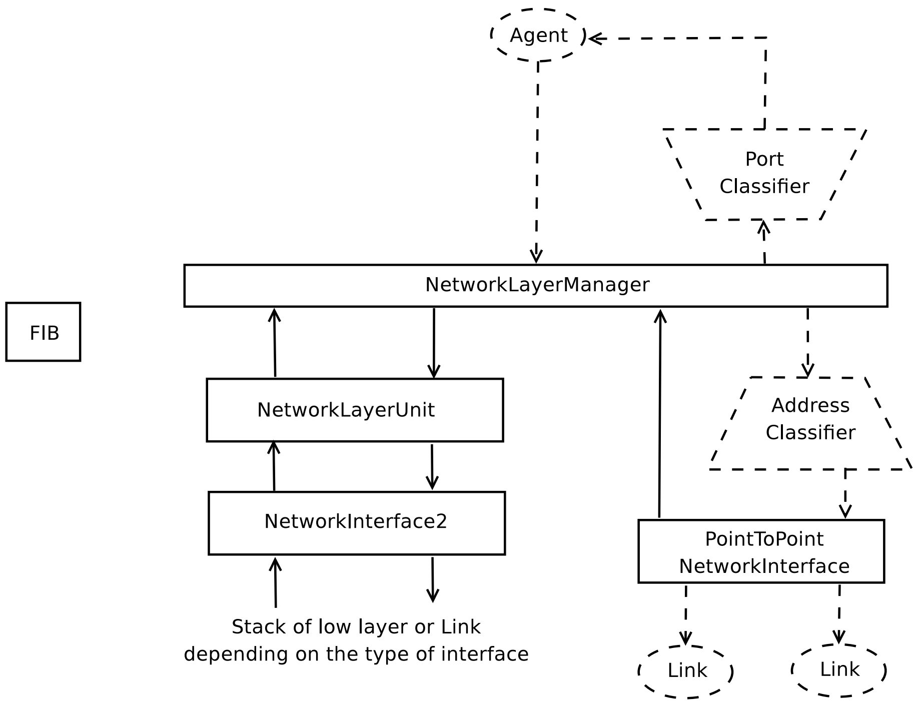

Figure A.2: Node Layout

The primary purpose of NetworkLayerManager is to maintain a list of Network-Interface2 and a list of NetworkLayerUnits as well as to handle and distribute each packet passing through the nodes. NetworkLayerManager decides whether a received packet should be forwarded upwards to the port classifier or forwarded to a neighbor node according to the packet destination. The NetworkLayerUnit consists of a ForwardingUnit and a RoutingUnit which handles the data packets and the routing packets respectively. The detail structure of a NetworkLayerUnit is shown in Figure A.3.

The new node layout also consists of a PointToPointInterface and a Forwarding Information Base (FIB). The PointToPointInterface is an extra layer added between a

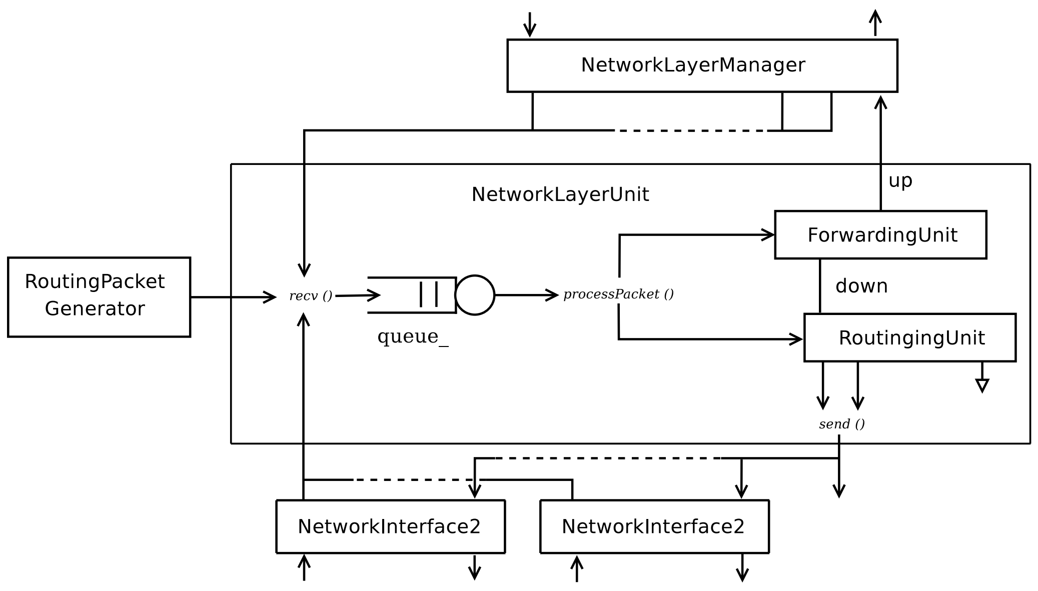

Figure A.3: Architecture of NetworkLayerUnit

Node and a Link. Operations such as *up* or *down* on a PointToPointInterface causes the link attached to connect or disconnect respectively. The FIB stores and maintains all the routing information. Further, information regarding neighboring nodes is stored in a NeighborInformationBase. For more detail refer [25].

Appendix B

CEAS extension modification

All the source codes related to the CEAS module are submitted along with this work. This also includes the TCL scripts and topology files. The multi-* network extension and the technical manual is availabe at http://www.q2s.ntnu.no/~paquerea/ns.php

B.1 Forwarding Unit

The CEAS ForwardingUnit premises are defined in "common/src/ceas-forwarding-unit.h" the modifications are made in "plain/src/plain-ceas-forwarding-unit.h". Plain_CEASForwa is an inherited class of CEASForwardingUnit which is futher inherited from ForwardingUnit of the multi-* network extension.

The method recv(Packet *p, Handler *h) is expanded. This method handles all the incoming data packets that pass through the CEAS NetworkLayerUnit. The method is expanded as in the following pseudo code.

```
if(ch->direction() == hdr_cmn::UP)
{
    // Packet is at the destination
    // Send the packet up to the associated agent.
}
if(ch->direction() == hdr_cmn::Down)
```

```
{
    // Outgoing packet
    // Create a list of neighbouring nodes associated with the cumulative pher
    if(No_Neighbour_Found)
    {
         // Drop the packet
         return;
    }
    if(isSetPrevHop && neighboringNodes > 1)
     { // previous hop prohibits forwarding back only if the neighoring nodes
        // remove the previous node from the list

    }
    // Select next node using selectRandomNeighbour
    // Forward the packet to the next node
    return;}
```

B.2 Link cost modification

The link cost modifications are made on "plain/src/plain-ceas-forwarding-unit.h". The link cost is caluclated by substracting the current time from the timestamp set during the generation of the ants.

Appendix C

The effect of different capacity links on TCP

The Figure C.1 compares the overall TCP throughput on all three system when the shortest path has lower link capacity. The result is very similar to the one we obtain during link congestion (see Section 4.4). Similarly, the results obtained after increasing the packet size to 9000 bytes shows significant improvement on overall TCP performance. However, the TCP performance on LSRP and DVR shows no improvements. Figure C.2 compares the results after increasing the packet size.

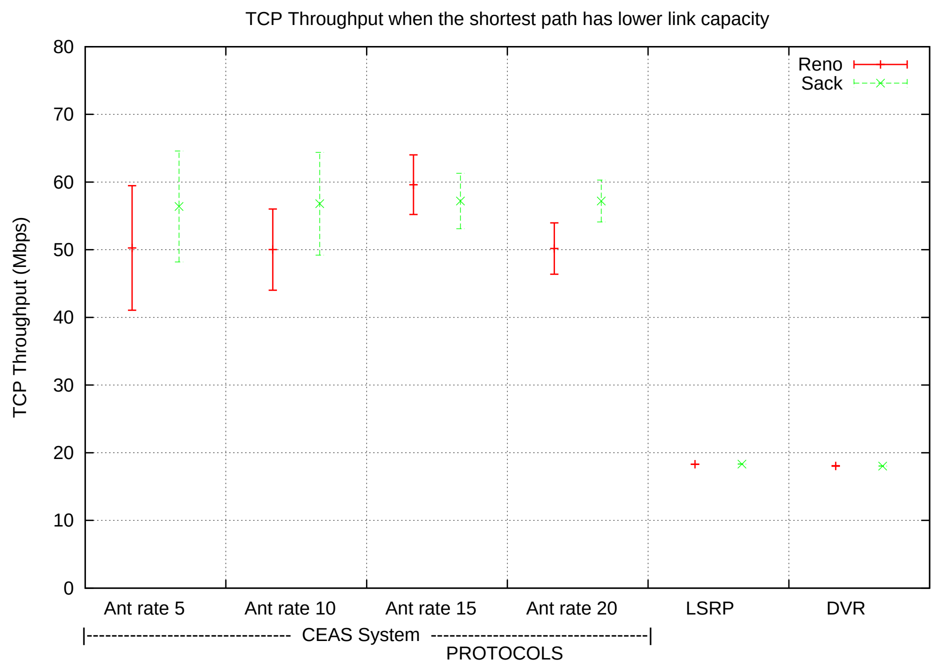

Figure C.1: TCP throughput when the shortest path has lower link capacity

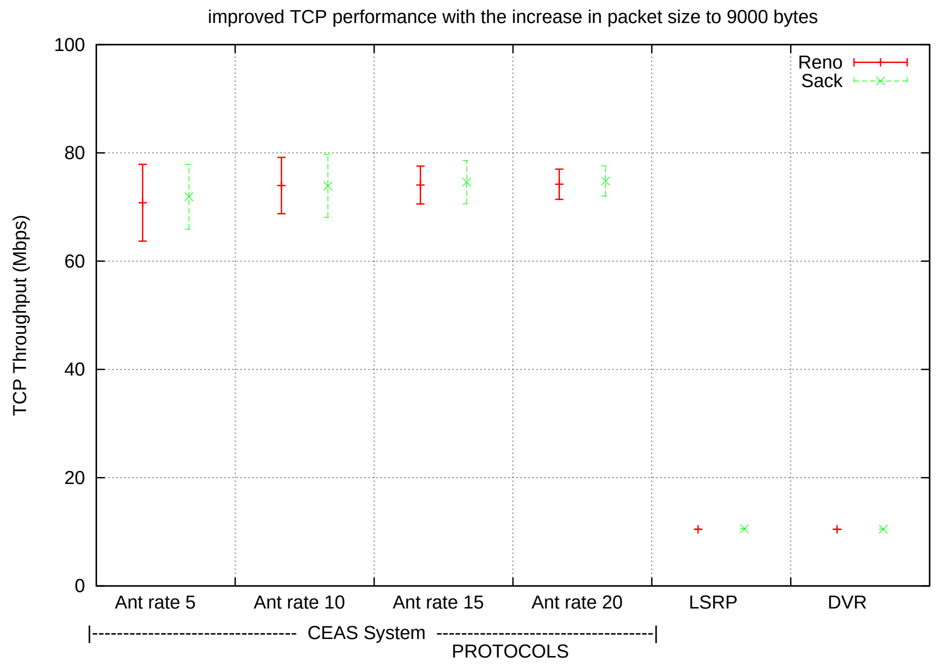

Figure C.2: Improved TCP perfromance after increasing packet size to 9000 bytes.

Appendix D

Multiple TCP stream on LSRP and DVR

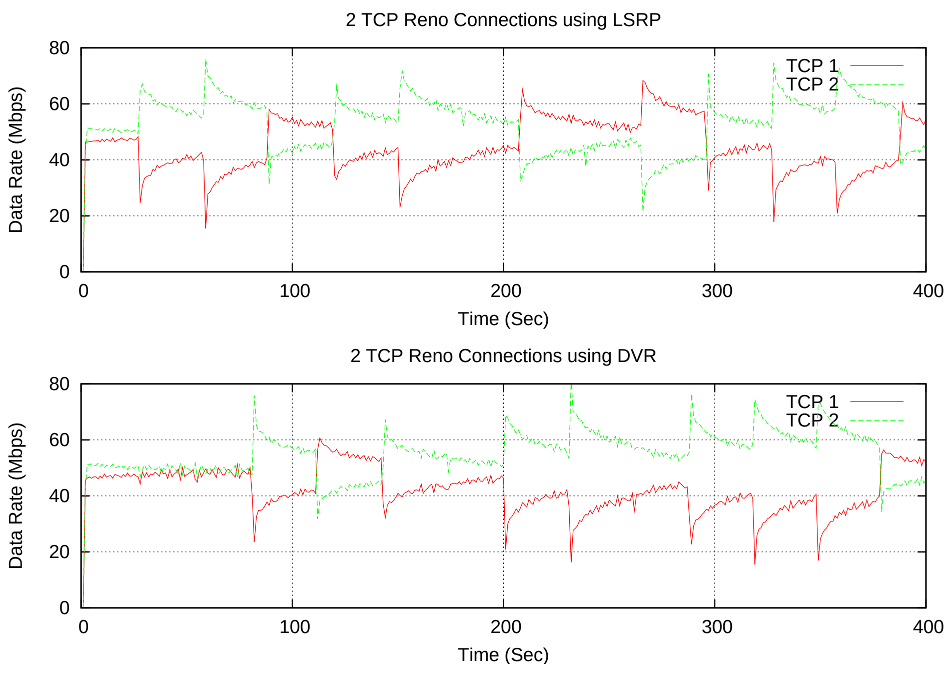

Figure D.1: Data rates of Two TCP stream on LSRP and DVR

The Figure D.1 shows the data rate of two TCP connection on LSRP and DVR over entire period of simulation. The TCP streams on LSRP and DVR always use the shortest path. The figure shows that the LSRP and DVR gives periodic preferences to the TCP streams during load sharing.

Appendix E

Locations of the 58 node Uninett Network

0 Aho	15 Halden	30 Molde	45 Stolavspl
1 Alesund	16 Hamar	31 Namsos	46 Stord
2 As	17 Harstad-gw	32 Narvik	47 Svalbard-gw
3 Arendal	18 Harstad-gw3	33 Nygardsgt	48 Svalbard-gw2
4 Bergen-BT	19 Haugesund	34 Oslo-gw1	49 Teknobyen
5 Bergen-HTS	20 Hitos	35 Oslo-gw2	50 Trd-real
6 Bo	21 Hist	36 Pil52	51 Trd-hvd
7 Bodo	22 Honefoss	37 Porsgrunn	52 Tromso-gw2
8 Drammen	23 Kjeller	38 Rena	53 Tromso-gw3
9 Elverum	24 Kongsberg	39 Sarpsborg	54 Tromso-gw
10 Evenstad	25 Kristiansand	40 Seilduk	55 Tynset

11 Forde	26 Kristiansund	41 Sogndal	56 Veths
12 Fredrikstad	27 Levanger	42 Stavanger	57 Volda
13 Gjovik	28 Lillehammer	43 Steinkjer	
14 Grimstad	29 Mo	44 Stjordal	

For more detail refer http://forskningsnett.uninett.no/forskningsnett/fnett-status.pdf

References

[1] Abouzeid, Alhussein A., Azizoglu, Murat, Roy, and Sumit. Stochastic modeling of tcp/ip over random loss channels. In *HiPC '99: Proceedings of the 6th International Conference on High Performance Computing*, pages 309–314, London, UK, 1999. Springer-Verlag.

[2] M. Allman, V. Pazson, and W. Stevens. TCP Congestion Control. RFC 2581, April 1999. `http://ietf.org/rfc/rfc2581.txt`.

[3] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[4] Blanton, Ethan, Allman, and Mark. On making tcp more robust to packet reordering. *SIGCOMM Comput. Commun. Rev.*, 32(1):20–30, 2002.

[5] S. Bohacek, J.P. Hespanha, Junsoo Lee, Chansook Lim, and K. Obraczka. TCP-PR: TCP for persistent packet reordering. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 222–231, May 2003.

[6] Stephan Bohacek, JoÃ£o P. Hespanha, Katia Obraczka, Junsoo Lee, and Chansook Lim. Enhancing security via stochastic routing, 2002.

[7] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[8] Kevin Fall and Kannan Varadhan. *The ns Manual*, 2008. `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[9] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, August 1999. `http://ietf.org/rfc/rfc2582.txt`.

[10] Malgorzata Gadomska and Andrzej Pacut. Performance of ant routing algorithms when using tcp. 4448/2007:1–10, 2007.

[11] Giovanna, Foukia, Noria, Hassas, Salima, Karageorgos, Anthony, Most a©faoui, Soraya K., Rana, Omer F., Ulieru, Mihaela, Paul Valckenaers, and, Van Aart, and Chris. *Self-Organisation: Paradigms and Applications.* 2004.

[12] Poul Heegaard, Otto Wittner, Victor Nicola, and Bjarne Helvik. Distributed asynchronous algorithm for cross-entropy-based combinatorial optimization. *In Rare Event Simuation and Combinatiorial Optimization*, September 2004.

[13] Poul E. Heegaard and Otto Wittner. Self-tuned Refresh Rate in a Swarm Intelligence Path Management System. In *IWSOS/EuroNGI*, pages 148–162, 2006.

[14] Poul E Heegaard and Otto J Wittner. Overhead reduction in distributed path management system. *Submitted to Computer Networks*, 2008.

[15] Poul E. Heegard, Bjarne E. Helvik, and Otto Wittner. The cross entropy ant system for network path management. *Telektronikk*, 1:19–40, 2008.

[16] Bjarne 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*, pages 14–16. Springer Verlag, 2001.

[17] V. Jacobson. Congestion avoidance and control. *SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, 1988.

[18] J. Kennedy and R. Eberhart. *Swarm Intellignece.* Morgan Kaufmann, 1st edition, 2001.

[19] Vebjørn Kjeldsen. Cooperation through pheromone sharing in swarm routing. Master's thesis, June 2007.

[20] Markopoulou, Athina, Iannaccone, Gianluca, Bhattacharyya, Supratik, Chuah, Chen-Nee, Ganjali, Yashar, Diot, and Christophe. Characterization of failures in an operational ip backbone network. *IEEE/ACM Trans. Netw.*, 16(4):749–762, 2008.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, October 1996. `http://ietf.org/rfc/rfc2018.txt`.

[22] Nicolas Meuleau and Marco Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 8:103–121, 2002.

[23] J. Moy. OSPF Version 2. RFC 2328, Ascend Communications, Inc., April 1998. `http://ietf.org/rfc/rfc2328.txt`.

[24] Teunis J. Ott, J. H. B. Kemperman, and Matt Mathis. The stationary behavior of ideal tcp congestion avoidance, 1996.

[25] Laurent Paquereau. *Extension to ns-2*, October 2008. `http://people.item.ntnu.no/~paquerea/ns/q2s_doc.pdf`.

[26] Laurent Paquereau and Bjarne E Helvik. A module-based wireless node for ns-2. In *Proceedings of first workshop on NS2 (WNS2), Pisa, Italy*, 2006.

[27] Laurent Paquereau and Bjarne E Helvik. Simulation of wireless multi - * networks in ns-2. In *Proceedings of second workshop on ns-2 (WNS2), Athens, Greece*, October 2008.

[28] Le Quoc, C., Bellot, P., and A. Demaille. Stochastic routing in large grid-shaped quantum networks, March 2007.

[29] Transmission Control Protocol. RFC 793, Defense Advanced Research Projects Agency, September 1981. `http://ietf.org/rfc/rfc793.txt`.

[30] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1:127–190, 1999.

[31] S. Savari and E. Telatar. The behavior of certain stochastic processes arising in window protocols. In *in Proc. IEEE GLOBECOM*, volume 18, pages 791–795, Dec 1999.

[32] Ruud Schoonderwoerd, Owen Holl, Janet Bruten, and Leon Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5:169–207, 1996.

[33] Sandeep Tamrakar. Comparison of TCP performance in a network applying CEAS based stochastic routing and Link State Routing OSPF. Master's project, NTNU, December 2008.

[34] Andrew S. Tanenbaum. *Computer Network*. Prentice Hall, 4th edition, 2003.

[35] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, and Deborah Estrin. The impact of routing policy on internet paths. In *in Proc. 20th IEEE INFOCOM*, pages 736–742, 2001.

[36] Lu Yong, Zhao Guang-zhou, and Su Fan-jun. Adaptive swarm-based routing in communication networks. *Journal of Zhejiang Univ. Science*, 5:867–872, 2004.