# Using Congestion Exposure for Traffic Engineering (May be misleading)

*Imad GRANDI*

A dissertation submitted in partial fulfillment
of the requirements for the degree of
**Master of Research**
of the
**University of London**.

Department of Electronic Electrical Engineering
University College London

August 31, 2010

**To whomever**

# Abstract

If I knew my thesis, I would know my abstract and guess what? It would be HERE.

# Acknowledgements

Words comprising the acknowledgements.

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

This is adraft that resumes the ideas that i want to speak about in the introduction

Internet is growing and also expectaion bla bla . One of engineering solution is path diversity, has multiple advantages robustesness bla bla, but the network is not efficielty benefitting from the path diversity, and the problem is in the internet design itself. The small problem, is that pooling has being always seen as a network thing, rising from transport guys says that they could do better since they have an end-to-end visbility. Actually they are right about the last point. This is one of the problems with the Internet design. The hourglass concept implies that the network should be kept dumb and all the intelligence should be kept at the edge. The reason behing this dogmm are diminishing, the equipement caapcity is increasing exponentially and the network providers are alreasy breaching it for diverse reasons. This problem exist also in congestion control. Network lack information to identify the responsibles for the congestions and are not doing that good in tehir attempts for dividing the capacity among their customers. Congestion exposure is one of the solutions, that allow to reveal to the network the congestion that a host is experiencing and though his participation for it.

But, this information could be also used for exploring path diversity. PREFLEX, is an architecture that uses this concepts and try to use it for balancing the traffic. The aim of this project was to evaluate how balancing could be done using this new architecture and then compare its performance with a tradditional traffic engineering approach for path diversity which is TEXCP that uses load instead of congestion.

## 1.1 Overview

## 1.2 Objectives

## 1.3 Document Structure

# Chapter 2

# Background and state-of-the-art

## 2.1 Traffic Engineering

Network traffic engineering is an old discipline in the telecommunication industry and has evolved with certain specificities for different networks types. This overview focus on the issues and techniques related to Internet and IP networks TE, though many of the discussed concepts are valid for other network contexts.

### 2.1.1 Definition and problem description

Traffic engineering is the discipline concerned with the performance evaluation and optimization of networks. This aspect of network engineering applies technology and scientific principles to measure, model and control the traffic. [**?**].

The main objective behind this intervention is to enhance the network efficiency, as it is perceived from both traffic and resources views. The first one is usually referred to as traffic oriented, and is concerned with the enhancement of the quality of service (QoS) parameters as seen by traffic flows. Including : minimization of delay, minimization of packet loss and maximization of the throughput. In particular, the aim of this project as mentioned above, will be the minimization of loss rate as the main objective of our traffic engineering protocol.

Meanwhile, the second category of policies said resource oriented, deals instead with the aspect related to the optimization of the resource utilization. In fact, having some subsets of the network being over-utilized while others are underutilized is an indication of a network not being used optimally. Maximum path utilization is usually used in this context as the principle evaluation measurement for network performance. TEXCP (see 2.1.4), the protocol that we'll be comparing with the results of our own architecture is an example of such approach. But, for both categories, the question that many might ask is how much TE could enhance the network efficiency? And does it really bring a significant added value? Doubts about TE utility usually use the tremendous increase of equipments capacity to promote that the demand on the network could be answered by simply putting more capacity on the network and that it won't significantly increase the network operator bill since marginals cost are converging to effectively zero. However, this vision of the Internet is simplistic. Internet has evolved from a simple network for the research community to a sophisticated communication infrastructure that provides a rich panel of services. The new users of Internet can't settle any more for the range of best effort services that the network offered initially. Network operators need to make sure that their customers are receiving the service they are requiring, and that they are able to differentiate the charging according to these requirements. Moreover, the near deployment of optical fibers and the increase of users expectation will mean that there will be more demand on the operator core network. The efficiency boost that TE could bring will help the operators to keep their margin. This is especially crucial in open market where fierce competition is taking place. The global context of convergence is taking down the incomes of classical operators, and the only way to keep a profit margin is by reducing cost and take the most from network their key asset. This is what Traffic Engineering attempts to achieve.

### 2.1.2 Basic Concepts

**TE as a controllability problem :** Independently from the optimization policy approach, a set of control tools and mechanisms are usually invoked in order to accomplish this policy. These control actions could be divided in three categories:

Acting on traffic management parameters is usually used to make sure that some elastic traffic is not behaving too aggressively or to differentiate the Quality of Service QoS perceived by different categories or classes of traffic. This could be achieved by using techniques like queue management, differentiated scheduling and traffic policing and shaping.

Acting on resources attributes and constraints that include controlling link bandwidth, buffers etc. Modification of routing parameters to control how traffic steers within the network.

**Routing and Traffic Engineering :** This last category is particularly important in this project since it is primarily involved in taking advantage of path diversity within a network. The Internet hourglass implies that network should be kept simple with routing is the key provided function. The routing process involves core network nodes or routers to choose the path over which to send packets. One of the Internet design principle for the network is that it should be kept plane and without any central nodes that might introduce vulnerability. This means that routing protocol have as a requirement allowing the routers to take local decisions while ensuring consistency and packet being delivered to their destinations. Routing protocol like RIP and OSPF succeed in this purpose. However, the topology driven nature of these protocols implies that they optimise the path choice of each destination independently without considering the traffic demand on the network. A solution that allows to go around these limitations could be achieved solution by manipulating configuration parameters of the previous routing protocols to enhance network performance. In particular, by having an estimation of the traffic demand on the network, network management could choose the link weights used by OSPF for example, on a way that results in an optimized distribution of traffic over the network could be achieved. In [**?**] an iterative algorithm is describe to optimise the network utilization by adjusting the weight through the use of heuristic mechanisms. This solution presents many advantages like compatibility with existing routing protocols and stability. In the other hand weight update frequency should be kept low since during updates routing consistency is not ensured.

**Traffic Engineering in MPLS networks** Limitations with routing are one of the difficulties encountered with traditional IP networks. MPLS Multi Protocol Label Switching answers some of them by using a connection-oriented switching mechanism: At the edge of an MPLS domain, the ingress LSR (Label Switching Router) adds a header to the packet and forwards it (see figure ??). The label field in the header is used to look up for the corresponding FEC (Forwarding Equivalence Class) that indicates to the core LSRs how to process packets and where to send them next.

Figure 1 MPLS Heade

MPLS allows the network management to provision tunnels within the network by configuring the core LSR. Edge LSRs are also configured to how they should assigned incoming traffic to these tunnels. This open the way for new Traffic Engineering capabilities. After gathering information about the traffic, the network management configures tunnels and assign the traffic to them in a way that ensures efficient bandwidth utilization. As a result, Constraint-Based Routing that assign traffic to paths according to their bandwidth requirement could be achieved. For example this could be coupled with the Shortest Path First algorithm by simply removing from the network topology the links that doesn't satisfy the bandwidth requirement. Greedy heuristic is an example how the network management could proceed to assign routes to traffic. In this algorithm the flows in the network are aggregated in a decreasing order of bandwidth. The previous CBR algorithm is applied to the first flow on the list, and a tunnel is created in the network according to the result of this step. The network graph is updated and the process is repeated for the next flow in the list. Fairness is not ensured by this algorithm since the flows with the lowest requirement in bandwidth might end up not served.

Figure 2 MPLS QoS management (based on IPN notes)

### 2.1.3 Limitations

In both Traffic Engineering approaches, IP based and CBR in MPLS, the optimization of the routing is carried out for an estimation of the future demand based on long term measurement of the current traffic and expectation of evolution. This is why both mechanisms are referred to as offline solution and raises some limitations. Indeed, actual traffic might differ from the network estimation and hence the routing might result in suboptimal or even inadequate distribution of the traffic on the network. Attacks, changes on external routing and link failures are frequent events and affect the traffic demands on the network. Failures are particularly wake point for this approach of routing. Offline TE deals with network failures by pre-computing alternatives routes. However, this could be done only for a limited set of network failures and TE should look for reroutes that perform relatively for a range of failures. As a result, suboptimal utilization of the network is frequent. Online TE is necessary to ensure that the network adapts in real time to changes on the demand. A second characteristic of the previous approaches is that they require a global vision of the network to choose the configuration parameters. Some constraints are related to this property, like the scalability of the solution and the frequency of the update of the parameters. In distributive schemes, most of the decisions are made locally without requiring frequent interventions from a central entity. In the next section, an example of a new TE solution that mitigate some of the limitation described above.

### 2.1.4 Example of an adaptive trafic engineering protocol: TEXCP

Introduction to TEXCP: TEXCP is a a dynamic and distributed TE protocol that targets the optimization of network utilization. TEXCP basic concept is that using path diversity we could balance the load among available paths in a way that optimizes network utilization performance. TEXCP is not itself a routing protocol but it sets on the top of existing MPLS systems and control how the flows are distributed over the network. TEXCP is an online mechanism, meaning that it reacts to the changes on the traffic demands and potential network failures. It is distributed and doesn't necessitate an oracle that have a global view of the network, boosting though its capability to scale in large networks. But, most importantly TEXCP has proved a strong stability that lacked most of the dynamic TE solutions until now. It targets performance optimization within a single domain, though it is considered as an Inter-Domain TE method. To have a more concrete idea, let's consider a simple architecture of an ISP network (figure ??) : From a domain perspective, a flow enter the domain in an Ingress point (For instance I1 in figure ??) and leaves it in an egress point (E1 for instance). For this Ingress/Egress IE pair, two paths are available. In MPLS, as we've seen in the previous section, every arriving packet in a domain is seen a label attributed. TEXCP use this mechanism to implement its load balancing mechanism: this load balancer called an agent will be associated to each IE pair and try to balance the traffic among the previously pair selected paths (TEXCP designers suggest to select a set of shortest paths connecting the pair) . The agent is also responsible of continuously probe the state of the network, and feed the balancer by control inputs. Also, TEXCP provides a feedback mechanism that prevent oscillation and capacity management.

Probing Network State

TEXCP load balancer needs to keep a track of the utilization state for the paths that he manages. This implies that every agent, associated to an IE pair, has to send a probe on all the paths of this pair. This probe will be treated (see figure ??)by every router on the path until reaching the egress point, that will send back an acknowledgement message directly to the ingress with the final state of the probe. The format of the probe is shown in figure ?? The separation period $T_p$ between two probe should be larger than the maximum round trip time RTT. However, the smaller is this value the faster the algorithm will converge. And hence it should be chosen slightly larger than RTT, A typical value for an average network is 100ms. During network congestion periods, it is possible that the probe get lost. Hence, if the timer for next probe fired before the acknowledgement of the current probe didn't arrive, the agent will consider the probe as lost and estimate that path is witnessing a congestion and though the utilization will be increased exponentially.

The Load balancer As previously mentioned, TEXCP objective is minimizing the utilization in the network. Let's try to put this problem in a more formal way. Let s an IE pair, and $P_s$ the set of selected paths for pair s. If $x_{sp}$ denotes the fraction of IE pair s traffic that goes through path p, the load balancer problem is to find the $(x_s p)$ configuration that will minimize the maximum utilization over all the network links min max ul subject to the following conditions:

This could be achieved if every agent tried to equalize the loss that it sees over all the controlled paths. According to the path utilization and the current rate distribution, the agent will send more traffic on underutilized paths and less traffic on over-utilized ones. The equation describing this behaviour are bellow

The traffic fraction distribution is updated every $T_d$ seconds. This interval needs to be set to 5 times the probe interval.

Feedback

Due to the distributed nature of TEXCP, instability and oscillation may occurs. Let's consider the basic architecture in figure ??. Link AB, is shared between pair I1E1 and pair I2E2 and hence in the case of the link being underutilized, agents associated with both pair will react by sending more traffic over this link and potentially causing the link to be over-utilized. Hence, it is necessary to have a feedback mechanism that will allow the core nodes to not only till the utilization of the links but also control and synchronize the answer of the multiple TEXCP agents that use the link. This mechanism is similar to what is used for congestion control when multiple sources sharing the same bottleneck attempts to adjust their rate and end up causing the bottleneck utilization to oscillate. The solution for both problems is to explicitly inform the sources of how much they should increase or decrease their rate. To provide the sources with such information, every core router needs first to compute the aggregate feedback, updated ever $T_p$ since it should be included in every probe. The aggregate feedback is an upper bound of how much the the global traffic on the link should increase/decrease. Hence, the aggregate feedback is positively proportional to the spare bandwidth S (S = capacity -load) over the link and the negatively to the current buffer size Q.

$$\phi = \alpha.T_p.S - \beta.Q \tag{2.1}$$

The next step is to divide this aggregate feedback over the available sources, TEXCP adopt a Max-Min allocation policy to divide this aggregate feedback, and hence installing a fairness that will prevent congested IE pairs from starving the others. Additive Increase Multiplicative Decrease (AIMD) standard is used to achieve this fairness and hence:

$$\phi > 0 smalldelta^{+} = \phi/N, smalldelta^{-} = 0 phi < 0 smalldelta^{+} = 0, smalldelta^{-} = \phi/smalphi_l \tag{2.2}$$

Where N is the number of active IE pairs that the router is seeing, and $_l$ is the aggregate load on link l. Before continuing on how this information is sent back to the IE agents, we would like to turn the reader attention to the scenario where multiple solutions of the optimization problem exist: consider a simple example where the two IE pairs in figure ?? have the same traffic, and and that the two links have the same capacity. One solution is that each router send one half of its traffic over each link, while the second solution is that the the first pair send all its traffic in link 1 while the other pair use the other link. In both solution the network utilization is the same and optimal, however the second solution has the merit of a reduce delay. So it we be good in case where multiple solution exist, shortest path are prioritized. This condition could be implement through the use of a weighted fairness allocation of the aggregate feedback and the previous equation will become:

$$\phi > 0 smalldelta^{+} = v_{sp} sum_{p' \in P_l} v_{sp'},\ smalldelta^{-} = 0 phi < 0 smalldelta^{+} = 0, smalldelta^{-} = \phi_{/}$$ (2.3)

Where, $v_{sp}$ is the weight allocated to path p, which should be unversed proportional to p length. When the ingress router send the probe, each router in the path update the probe in the following way: PATH UTILIZATION = max (PATH UTILIZATION, ul ) POSITIVE FEEDBACK = min (POSITIVE FEEDBACK, + ) NEGATIVE FEEDBACK = max (NEGATIVE FEEDBACK, ) Once the probe reaches the egress router, the router send back directly the final values. The first field in the probe is the utilization of the most congested path and in a similar way, the value for the feedback. The agent will update his sending rate $g_s p$ according to AIMD : $g_s p = g_s p + + -g_s p -$ .

Splitting the traffic

Flare What the agent needs to do is to update $(x_{sp})$ the splitting ratio of the incoming traffic over the available paths. Traditionally, traffic splitting was tackled from two approaches. The first one is packet-based. However, this solution is very harmful to transport protocol that are sensitive to packet reordering like TCP. The second solution to split in flow basis, i.e. all packet of the same flows are sent over the same path. By assuring that a flow's packets goes over the same path, packet reordering won't take place and hence TCP could achieve a better performance. However, the flows have different sizes and characteristics hence this approach of traffic splitting deviates usually from the desired splitting ratio. An intermediate solution is described in [**?**], that take advantage from TCP burtsiness to do the splitting. A TCP flow life cycle is divided into a set of packets bursts called flowlet separated by idle periods. The flow-lets could be characterized by the minimum time that separate two successive flowlets. If we rightly choose this value larger then the value of delay difference , we could send the packets of the different flowlets in different paths without risking reordering problems: Let's consider a simple scenario where the last packet of a flowlet leaves the paths divvergence points at time $t_0$. If we know that the when the FLARE suggest also a splitting mechanism that uses a Token-counting algorithm, The final algorithm could be sum up in the following 3 steps: 1-when a packet arrive the counters of all the flows are updated as follows:

$$c_{sp} = csp + x_{sp}PacketSize. \tag{2.4}$$

2- if the packet flow is not TCP or the last time the flow has been seen is older then the time out value, the packet is sent on a the path with the highest token counter value. The packet is sent over the same path as the last packet of the flowlet. 3-The last seen time is upadated as wel as the counter of the selectedpath according to :

$$c_{sp} = csp - PacketSize. \tag{2.5}$$

## 2.2 Congestion Control in Internet networks

### 2.2.1 Problem description

A key characteristic of IP networks is its packet switching mechanism that allows a significant increase in the efficiency of bandwidth utilization. This efficiency is explained by the fact that multiple communications could take place over the same physical resource. However, this scheme rises the important question of how the capacity should be shared. And Internet discovered this in the hard way; during the mid 80' a serial of network crashes took place: high packet delay, high loss rate and low goodputs are the characteristics of the communications taking place then. These are now the symptoms of a network being over-utilized, and though these collapses had been baptised as Congestion Collapses [**?**]. As described in the RFC, the causes of this problems is that the offered load in the network is overpassing the network capacity resulting in full queues and so increase delay and more packets being drop. The same RFC state that the problem particularly occurs when TCP is the transport layer being used over IP. Rightly, TCP retransmission mechanism was really clumsy. Indeed, when the liable transport layer considers that a packet wasn't routed to destination it retransmits not only one but multiple packets. This means additional load over the already congested network and congestion getting worst. But even if this retransmission scheme was replaced, it is up to the end host to adapt their transmission rate to limit the congestion and enhance the overall network performance. This is what was achieved through the implementation of congestion avoidance algorithm proposed by Van Jacobson [Jacobson88] for TCP. The same report suggested also that the network performing congestion control might enhance further the global performance. Some of the techniques used by the network were introduced later in what is known today as Active Queue Management AQM.

### 2.2.2 TCP and congestion control algorithm

Transmission Control Protocol TCP is a transport layer protocol that provides a connection-oriented, reliable stream delivery service over the connectionless unreliable IP network. As a transport protocol it provides a demultiplexing function that allows several process or flows at a single IP address to communicate concurrently. It provides also reliability by establishing a connection with the destination end host, so data is transferred and acknowledge in terms of byte streams enabling thus the possibility of retransmissions when the losses are detected either by the expiration of the acknowledgement timer or the reception of acknowledgement of previously acknowledged bytes (duplicate acks). To avoid bandwidth wastes that usually accompany positive acknowledgement schemes, TCP use a sliding window algorithm that limits the number of segment a sender could transmit before receiving acknowledgements. Once the acknowledgement of the first packet arrives the window slides and a next packet could be sent. The number of packets that could be sent before receiving a first acknowledgement is called the window size(figure douglas book page 214). Thus, the transmission rate of a sender could be estimated in function of the window size as:

$$R_{Tx} = \frac{Windowsize}{RTT} \tag{2.6}$$

where RTT is the Round Trip Time which is the necessary time for the packet to arrive to the receiver and that the acknowledgement to get back. This window side is variable and limited by the receiver advertisement window, that allow the receiver to specify the number of bytes that it could receive, and also the congestion window that is used by the sender to limit the transmission rate to control the congestion. A variety of congestion control algorithms exist depending on TCP implementation but they all have Slow Start and Congestion Avoidance in common.

So during the Slow Start phase the sender increase exponentially the congestion window (cwnd) until reaching the slow start threshold (ssthresh) then TCP sender enters the Congestion Avoidance phase where the cwnd is increased by 1 every time an acknowledgement is received. The next part is when TCP implementations differ, so when the loss is identified (Retransmission TimeOut (RTO) or duplicate acks) the cwnd is reduced to 1 and the slow start is restarted, or to ssthresh in the case of Fast Transmit and TCP senders enter directly the phase of Congestion avoidance. ssthres is reduced also and usually put to the half of the cwnd before that the congestion occurs. Thus this algorithm is designed as Additive Increase Multiplicative Decrease AIMD.

### 2.2.3 Active Queue Management

An individual action from the end hosts toward the congestion presents some limitation. End hosts are only aware of the congestion when the packets start being dropped and they react simultaneously to the congestion causing thus a global synchronisation. RFC 2309 states that using some queue management techniques for congestion control will allow to decrease the number of dropped packets and avoid global synchronisation. RED or Random Early Discard is one of these scheme that permit to avoid oscillation and the global synchronisation by starting to discard packets randomly before the queue is completely full: a figures The algorithms could be described as If the queue size is smaller than $T_{min}$, packets are enqueued. If queue size is larger than $T_{max}$ discard the packet. If the queue size is between $T_{min}$ and $T_{max}$ packets are dropped according to a probability p. The key to make the algorithm work is to well choose the algorithm parameters. Some of the consideration are to ensure a high utilization of the outgoing link and provision enough difference between

$$T_{min}$$

and $T_{max}$ so the end hosts are informed and could react. Also differntial treatment could be applied by using different value of the discard probability p for different flow classes. Explicit Congestion Notification (ECN) RED could also be combined with ECN [**?**] another AQM technique that allows the network to explicitly inform the end user of the congestion instead of relying only on the the host detecting packet losses. The network feedback is passed to the end user through two unused bits in the IP TOS byte. Thus 4 code points are available : 00: Non ECN-Capable Transport - Non-ECT 10: ECN Capable Transport - ECT(0) 01: ECN Capable Transport - ECT(1) 11: Congestion

Encountered - CE ECT(0) and ECT(1) indicates that the originator of the packet is ECN capable and in this case, if the packet has experienced congestion the router will mark the packet with CE code point. This is carried out in a similar scheme to RED and allows transport protocols and particularly TCP of being notified of congestion without having to experience loss. By marking packets instead of dropping them, ECN reduces the number of dropped packets. It also reduces the delay for the network layer to consider the packet as dropped and hence the delay of its reaction to the congestion. Finally, is in-band signalling mechanism and doesn't add additional traffic which is particularly harmful when the network is experiencing congestion.

## 2.3 Congestion Exposure and Re-Feedback Principles

### 2.3.1 Motivation

As explained in the previous section, congestion control in TCP/IP networks is mostly based end users running congestion avoidance algorithm. In fact, every TCP routine increases its transmission rate until packets are dropped due to a congestion in bottlenecks. This mechanism might seems weird, but it is fundamental in the Internet architecture where there is no circuit dedicated for a communication and though no fixed rate. The second outcome of TCP congestion avoidance algorithm is capacity sharing. From a first glance at the mechanism , we might think that it could be considered as fair since all the users are running the same algorithm and respond the same way to the state of the network. However, this fairness is only a mirage. A first way to walk around TCP limitation is by running simultaneously multiple TCP instance, since every instance gets an equal share of the available capacity. But, the real problem is that the users are not all the same, having different behaviours and participating differently to congestion. This problem is a real puzzle for ISPs today. A small part of their customers that uses greedy applications, like peer-to-peer and video streaming, are taking more and more shares of their network capacity [**?**]. Hence, they couldn't leave any more the capacity sharing task for TCP and they turned to new methods for controlling the traffic and explicitly making sure that heavy users are not taking the whole bandwidth. However, the success of these methods is mitigated because the lack of the information about the congestion a flow is causing and though they are unable to treat them according to what really matters. Hence, making congestion visible in the network layer is a necessity to build a fairer sharing mechanism. ECN(see section ??), the protocol that the core routers uses to explicitly inform the end hosts of the congestion they are causing is a step in the right direction. Using ECN, a network is able to calculate the congestions caused by the flows that a user receive. But this is not enough, the problem with P2P applications for example is the congestion that they users are making in their upload sense and also receiver could also ask politely the senders to change their rate and can't imply how they work, congestion exposure goal is to make the congestion fully visible in both direction and for all the network nodes.

### 2.3.2 Overview of the current traffic control methods

As pointed out in the previous section, ISPs are attempting to control their customers traffic. These methods could be divided in three categories: network layer measurement based, transport and application.

L3 Measurement policing

The original design of Internet supposes that routers looks only for information in the detained in the network layer. Two possible measurement could be carried on: volume and rate.

Volume accounting is the easiest information that a network provider could calculate to have a clue about who are heavy users . To obtain this information, it is enough to look for the packets sizes at the IP header and then add them up. ISPs could use this information to impose the maximum limit for a user in a period of time. However, volume accounting doesn't give a real vision of the damages that a volume of traffic is making since it might depends on the period of time as well as the part of the network where it goes.

The second tool in this category is the rate measurement. It is especially the case for the accounting between ISPs, where for instance the charging could be made on the percentage of peak rate crossing the border. A simple way to measure the rate is by accounting the volume for short periods of time. Traffic shapers could be used to make sure that the traffic doesn't overflow the maximum rate. Even, if such policies will allow to limit the damage of heavy users on the others, but they induct situations where some users are allocated more capacity than they really need to prevent the risk of all the users sending fast at the same time, resulting though a poor bandwidth efficiency.

Higher Layer Discrimination

ISPs are aware of the limitation hourglass design of the Internet and in the same time the equipment capacity is increasing exponentially, and the reasons for keeping the routers simple are diminishing. Hence, some ISPs have introduced DPI operations that allow to investigate the packet to identify the flows that belongs to the applications that they thought being responsible for the network congestion. A common example is P2P file sharing. ISP regard these applications as ith low value for the customers and cause most of the congestion problems in the network. These technique are in the center of the ongoing debate about Net Neutrality. Even more, the efficiency of these techniques is being questionnaire since P2P applications responded to the DPI by using encryption techniques.

The last example is usually called bottleneck rate policing. They require the deployment of the rate policers at the bottleneck that based on some assumption over what an flows might accept about traffic shaping. Another problem with these approaches is that the traffic usually traverses multiple bottlenecks and therefore limiting hence the utility of the solution.

### 2.3.3 Re-inserting the Feedback Re-Feedback

All the approaches described above fail in controlling the traffic based on what counts the most, the congestion caused by the customer. The main reason behind this failure is in Internet design itself that doesn't provide enough information about congestion to the network. Re-Feedback attempts to correct the weak points in Internet feedback mechanism. As explained, previously, ECN markings till routers about the congestion on the upstream route (the part of the route between the sender and receiver). Similar accounting could be achieved with the Time To Live (TTL) field. The modification that Re-Feedback is suggesting, is that instead of aligning the information at the source, the sender modifies the field by targeting to reach a certain metric at the destination. For example, TTL field shouldn't be initiated to 255 at the source, but should be modified to reach an agreed value at the destination (say 16). This means that the routers by inspecting the IP header will be able to have an estimation of how many nodes left for the packet to reach destination.

Re-ECN is the implementation of this principle using ECN markings that will attempt to provide a metric for accounting the congestion. As table ?? depicts, a value is associated with each markings, for example a neutral packet has a value of 0 and if it experience congestion it will become negative with a value of -1, and the accounting at the routers is decremented. Thus, the sender should transmit enough packets with a positive value +1 (router increment their accounting when they see positive packets)so the accounting near the destination won't go bellow 0. If these packets experienced itself congestion they will be cancelled 0. Now the network needs to control the network at two critical points. At the egress where it could have a vision of the congestion for the full path, the operator should ensure that the source is putting enough credit or positive packets so the accounting stays positive, and provision incentives for that ( for instance dropping packets where the metric is negative). Now the network is capable to account the congestion created by the source by accounting the positive packets that correspond to the congestion the source make. Based on this metric the network will be able to identify the sources that cause most of the congestion and could treat them accordingly. For instance, each source will be allocated a limit of the positive packets and once it over pass this limit the quality of the service might drop (traffic policing, limited bandwidth etc.)

### 2.3.4 Use Cases

Now that the operator is able to charge the customer on explicitly the congestion they are causing, Heavy users are motivated to act less aggressively when the network is experiencing congestion. This won't significantly delayed their communication, since when light users acting more aggressively will have a bigger share allocated and they will finish their communication faster, so the heavy users could recuperate the whole bandwidth again.

Other benefits could be drawn from the Congestion Exposure scheme like the use of the new encoding to identify the attacks by their patterns. And the aim of this project is to evaluate the use of another congestion exposure scheme in Traffic Engineering.

Chapter 3

# PREFLEX

Still working on it.

In this chapter, we introduce PREFLEX, a new and simple architecture that attempts to provide the basis for new resource pooling approach while resolving some of the obstacles described in the previous chapter. In section 3.1 we are going to see Path RE-Feedback the first component of the architecture and that will allow the network to explicitly inform the end host the preferred path. While, the end-host role is to expose the congestion experienced through LEX the loss Exposure protocol described in section 3.2, that will allow the network by only inspecting the IP header to have a good vision about the congestion state of the different part of the network. The cooperation of these two components make possible a balancing scheme congestion rather than load as it is the case for most of the current traffic engineering protocols. The algorithm is described in section 3.3.

## 3.1 Loss Exposure

We discussed in section 2.3, how there is a lack of congestion information available at the network layer and how this affects the network efforts towards controlling the traffic and optimizing capacity sharing among the users. This is the main incentive behind the effort of enabling the network with mechanism that will reveals explicitly information that were confined until now in the transport layer, by revealing this information the network won't have the need to inspect higher layers to get the information. Keeping the same spirit, Loss EXposure is a protocol that allows to reveal in the IP header information related to losses that the transport layer is expecting.

Code Point Explanation/Meaning Not-LECT Not Loss Exposure Capable transport

LECT Loss Exposure Capable transport LEx Loss Experienced FNE Feedback Not Established

FNE

The first step in this protocol is similar to any connection-oriented transport protocol, establishing the connection. This means packets that belongs to flows where a feedback wasn't estavlished yet will be marked by the end-hosts with the FNE code points, similar to what is done in re-ECN. Using TCP terminology, FNE marked packet correspond especially to the SYN and SYN$_ACKpacketsexchangedatthestartofthecommunication, butitmightincludealsopacketssentafteral$
$markedpackets.Theassociationissimilartotransportlayerconceptofflow, anditwillbereferredtoasflo$
$let.Bymakingthenetworkallocatesthesamepathtothestreamofpacketbelongingtothesameflowletmany$
$basedsplittingcouldbeavoided.Whileflowsbeingasmallgraniularityfromtheflowswillallowtobenearer$
$basedspliting.Also, byknowingthataflowletwilltakethesamepath, thetransportlayercoulddividetheflo$

Echoing the Loss

Now that the feedback has being established, most of the tranport layer protocols like TCP will attempt to adjust their sendong rate based on the network state that they are observing (this include delay, loss rate and ECN marking). This means that a new congestion signal will be added to the existing ECN: this new signal will indicate the end-to-end congestion experience in the previous RTT, while the ECN marking gibes an idea about the current congestion experience at the upstream of a node. This new metric could be used for evaluating from one hand the congestion that end hosts are making and also the service offered by the network. This constitutes a simple form of congestion exposure compared to the one in re-ECN. LEX requires the end-hosts to mark the retransmitted packets with the (Lex) codepoint. Gence the network could accurately estimate the end-to-end loss experienced in every path by simply dividing the number of bytes marked with Lex codepoint, by the total number of bytes marked either LECT or Lex. As we are going to see in the balancing section, the information of the loss experienced in the different paths will be used to decide the amount of flows that will in every path. Another potential advantage that could be drawn from this marking is that the network could prioritize retransmitted packets.

Analysis of the LEX protocol and comparison with re-ECN It could be easily concluded from the description made of LEX, that it borrows many concepts from

## 3.2 Path Re-feedback

## 3.3 Balancing with PREFLEX

## Chapter 4

# Implementation

This chapter presents the framework that has been developed to simulate the new protocol PREFLEX and compare its performance with TEXCP. The framework was developed using ns-3, a new network simulator introduced in section 1. The description of PREFLEX module will follow in section 2. In section 3, TEXCP module will be presented by highlighting first the similarities with PREFLEX module before expending more on the specific elements that has been developed for TEXCP. At the end of this chapter, a broad discussion about the framework will describe the difficulties encountered for the the implementation and the assumptions and limitations of the framework.

## 4.1 Overview of ns-3:

ns-3 is discrete-event network simulator for Internet systems. The eldest versions of the simulator, ns and especially ns2, are famous and widely adopted by the research community. ns-3 is free, open source and relies mainly on the research community for development with a new update released every quarter. ns-3 is a new software and hence not-compatible with ns2, but its new design and concepts answers many of the problems encountered with ns2 and makes it more suitable with the current trends on the world of Internet research like software extensibility, extended realism, and integration of external tools, etc. ns-3 software core is written in C++, with the option of interfacing using Python.

**ns-3 architecture key concepts** ns-3 as any other network simulator is built mainly over concepts borrowed from networking, that have sometimes special meanings due the abstractions made by the system. In this conceptual architecture, node is certainly the most fundamental concept. It is used by ns-3 to indicate the basic computing device in an equivalent way to the concept of host used in Internet jargon for devices connecting to the network. Thus, ns-3 nodes are the recipients of all the functionalities that are going to be used. An analogy could be drawn with a computer where we add applications, peripheral cards, drivers and protocol stacks etc.

Among all the functionalities, enabling the nodes to communicate physically among each others is certainly the most common in all the simulation scenarios. The channel/net device pair concept fulfils this task in ns-3, where channels simulate physical medium (wireless for example) and connect nodes through the net devices added on them. Net devices are equivalent to peripheral cards on computers. Point-to-point channel/net device pair is the basic example in ns-3 and connects simply two nodes. Ethernet, Wi-Fi and WiMax are samples of the existing implementations in ns-3. Following the ascendant direction of the communication stack, routing is the next functionality to be added at the node to ensure that packets are routed from a end-to-end. Ipv4 and Ipv6 are both supported in ns-3 and a routing module is associated with each one. This module process incoming and outgoing process to decide which is the next destination. This module has a particular interest for the implementation of our framework since the central functionality of both TEXCP and PREFLEX is to use path diversity. An important property of routing in ns-3 is that it is possible to have a list of routing modules on the same node and if the first module failed in routing a protocol the next could be used. Having PREFLEX or TEXCP routing modules over ns-3 Ipv4GlobalRouting will allow to only implement the function related to their relative operations and not all the routing functions (for example, Ipv4GlobalRouting has a function that populate the routing tables of all the nodes).

The other key concept in ns-3 is application. ns-3 applications run over end hosts nodes and use the communication infrastructure built using the concepts introduced previously to communicate between each others. This abstraction is used for any program that generate activity and hence drive the simulation. Similar to software applications on real systems, sockets are the interface that ns-3 applications used to interact with the network. ns-3 has implementation

## 4.2 PREFLEX Framework

PREFLEX module has been developed prior to this project, hence this section will be confined to a brief functional description.

As explained in chapter 3, PREFLEX is a mutualistic architecture that expose the loss to the network, and allows a cooperation between the end hosts and network on pooling path diversity. Hence implementing PREFLEX requires changes on both the end hosts and the network:

### 4.2.1 PREFLEX at the end hosts

The first functionality to be added at the end host is enabling LEX, which will allow to reveal to the network the congestion level observed by the transport layer. More precisely, LEX works by exposing the retransmissions that are naturally controlled at the transport layer. Hence, the TCP routine running at the end hosts should be modified to make it explicitly marks the retransmitted packets with the equivalent code point. In ns-3 two types of TCP/IP stacks are available, ns-3's proper stack and nsc (Network Simulator Cradle) which is a framework that embedded real world operating systems stacks like FreeBSD, lwIP and different kernel versions of Linux. re-ECN, was already implemented in nsc (linux-2.6.26) and as explained in 3.2, LEX compatibility with re-ECN makes it requires only the modification of the currently unused code point at re-ECN to signal retransmissions instead. Hence, for nsc, LEX was built over re-ECN module, while for ns-3 own stack, it was implemented in standalone.

The other task of the end host is related to Path Re-Feedback. The network carries out a reverse path lookup (see section 3.2) and choose on which path the network should send the new flowlet. This path choice is only a recommendation from the network based on the congestion observed for different paths. End host could follow this preference or choose to use another one (for instance if running Multi Path TCP) and in both cases it specifies to the network the path to use in ToS field. The interest of this project is to evaluate the network performance in balancing the congestion and hence the end hosts were not enabled and hence the end hosts were made to follow the network preference and not change the path allocated to the flowlet.

### 4.2.2 PREFLEX in the network

In order to reliably estimate the end-to-end congestion from the LEX markings, the aggregation of packets streams should to be carried close to the source. Hence, the part of the network where PREFLEX is located should be the first node after the end host (equivalent to end and which is equivalent to an ingress point in intra-domain TE (TEXCP for instance). The ingress point is also responsible of sending the packet on the path specified in the ToS field. This is a routing function and hence it was implemented as a new ns-3 routing protocol module that will be used by ingress routers. Every router has a counter that aggregate the LEX code points for the different paths and that allows to estimate the congestion level. This information will feed the balancing algorithm that will decide the split or the fraction of flows that should be routed over each path. Then this split is achieved through Path RE-Feedback. When a new feedback loop is established (exchange of FNE packets) the routing module identifies the ingoing FNE packet and decide on which path to send the outgoing flowlet according the predefined split. In practice, this was implemented by using a token counter mechanism similar to the one in FLARE (see section ): a token counter is associated with every path and incremented by the value of the desired split whenever a new flowlet comes. The path with the highest token will be chosen and its counter will be decremented by one. Conceptually, PREF is an independent element of the architecture and hence it was reimplemented for TEXCP framework to evaluate its performance.

The last function of the routing module is deciding the split in the first place. As explained, in the previous chapter, balancing by PREFLEX combines different approaches. Changing directly the algorithm at the routing module will induce a compilation of ns-3 core software. During the period of the algorithm optimization, the changes were frequent and as solution the update function was implemented in the scenario script and a callback is used by the routing module to call this function. Hence, only the scenario script needs to be compiled. This scheme is still in used for the configuration of the PREFLEX balancer.

## 4.3 TEXCP Framework

### 4.3.1 an overview of the framework

Recalling what was written in section 2.2.3, TEXCP is a traffic engineering protocol that targets the optimization of utilization within a single network. As figure ?? depicts, TEXCP framework could be divided in 4 elements: a routing module -similar to the one in PREFLEX-, a flowlet classifier -that is used to split the traffic using FLARE (see section 2.2.3), a special queue that process TEXP probes and a traffic shaper for congestion management.

### 4.3.2 Routing Modules: Ipv4TexcpRouting Ipv4TexcpTableEntry

TEXCP routing module is similar to the the one in PREFLEX framework. When a packet comes, the routing modules fetch the table of available routes for the packet destination and call the path selection process. Two selection approaches are available PREF (section 4.2.2) and FLARE (section 4.3.3) . The outcome of this process will determine on which interface the routing module will send the packet. This module contains also the implementation of the functions of TEXCP agents, that are mainly responsible for probing the available paths of the associated Ingress/Egress pair, and also the calculation of the split distribution over them. Agents and probing paths. Every $T_p$ seconds agents at ingress points send probes on all the paths that it manages. The probe message is carried over UDP and not as an ICMP message like suggested in [**?**]. The reason behind this modification is that UDP sockets are easier to manipulate in ns-3. However, using sockets don't allow to specify which path the probe will take since all the probe has the egress point as a destination address. As a solution, a tag is added to the probe indicating which path to follow and the Ipv4TexcpRouting::RouteInput method responsible of routing outgoing packets to use the interface indicated in the tag. This is the sending part, agent at the egress point is required to send back the probe to the ingress point. Since studying routing in one direction is sufficient for this project, it is possible to configure from the simulation scripts if the agent is an egress router and thus restrict its functions to listening for probes and sending back acknowledgement to the ingress routers. This requires also the agent to configure listening sockets over all their interfaces to intercept incoming probes and acknowledgements.

Calculating the split distribution

Probes messages allow TEXCP agents to keep track of permitted routes utilization. Agents set a second timer (with an interval equals to 5 time the probe interval) to update the split distribution: traffic is moved from over-utilized to underutilized according to both their utilization and their current split. In the implementation, a limitation has been set for the minimum fraction that could be sent over every path to make sure that paths stop being not utilized.

### 4.3.3 Traffic Splitting: Ipv4FlowletClassifier

As explained in section 2.2.3, splitting the traffic is one of most difficult practical problem: once TEXCP agent calculates the split distribution, the real difficulty is to distribute the traffic over the different paths according the split accurately and without severely damaging the transport layer with packet re-ordering. TEXCP framework implement both FLARE and PREF and the user could choose between them from the simulation script. PREF implementation was described in section 4.2, so we will focus in this section on FLARE implementation. Recalling FLARE description in section 2.2.3, the basic idea behind FLARE is that a flow could be divided into smallest streams called flowlets and each of them will be routed to one path. Thus, there is a need to identify to which flowlet a packet belongs and then routed to the path associated with the flowlet. Ipv4FlowletClassifier is the that the routing layer calls to classifies packets into paths. The classifier requires to keeps a state of each flowlet, that contains the flow tuple Id, the time the last seen time and the path associated with the flowlet. So, if a packet belongs to a new flowlet ( a new flow or the flow hasn't been seen for a period of time larger then the time out) the flowlet select a new path according to the token counter mechanism of FLARE. If the packet belongs to an existing flowlet, the packet is routed to the flowlet allocated path. In both cases the last seen time attribute is updated. It is worthy to mention that only, TCP flows are concerned with process, and also that the list of the flows that the classifiers track is checked periodically to delete all the flows that had been idle for long periods.

### 4.3.4 Controlling congestion through traffic shaping: TrafficShaperNetDevice

In addition to informing TEXCP agents of the current paths utilization, probes are used by the core routers to control the transmission rate of ingress routers through the mechanism of feedback. This is especially useful when core routers are experiencing congestion so they are able to inform the ingress routers to reduce their transmission rate. Packets exceeding that rate will be queued if there is space or dropped. This is a traffic shaping functionality that the agent can't fulfil from the routing layer. In ns-3, all the task related of transmission and queuing are carried out by net devices. However, the already implemented net devices in ns-3 didn't allow to change the transmission rate neither to have a different transmission rate for every destination. Hence there was a need to change the current implementation of the point-to-point net device to add these functionalities. The new implemented TrafficShaperNetDevice is a subclass of ns-3 PointToPointNetDevice and hence is used with same channel and reuse many of its functionality. This configuration avoid the need to implement all the methods and function of a net device, and limits the modification required to implement the module to the ingress router. Traffic shaping is usually implemented through the leaky bucket algorithm. In the new net device a leaky bucket is associated with each destination, so when a new packet comes it is queued in the corresponding leaky bucket. Then packets are leaked into the global channel queue. The leaking rate, or the draining rate of each buffer is controlled by TEXCP agent according to core routers feedback.

### 4.3.5 Texcp at the core routers: TexcpQueue

All the modules described until now are located at the ingress router, however TEXCP and particularly its probing mechanism requires the intervention of the core routers as well. They are required to compute the link utilization and the feedback and to update passing probes with these. Information about the link utilization and dropping are usually confined in the lower parts of the node. In particular, ns-3 queue that are associated with net devices keep statistics about the passing traffic. Thus, the decision was made to implement the core routers functionalities in a queue module. This queue will be similar to a drop-tail queue but will be also enabled with the capacity required for probe processing. The first function is clearly the calculation of the link utilization. Queues in ns-3 keeps track of many statistics like received, dropped and currently in queue, but they don't keep a track of the link utilization. By definition, the link utilization rate is the percentage of time a link was active. It was complicated to calculate the utilization according to its definition, however an accurate approximation could be calculated as follows: $link_utilization = \frac{ReceivedBytesDroppedBytes}{LinkCapacity}$ The limitation of this formula comes form the fact that the queue buffer may delay the real link utilization. Secondly, the module needs also to compute the aggregate and individual feedback. The calculation of the aggregate feedback is straightforward usimg the queue statistics and equation ??. While the calculation of the positive feedback requires the queue to keep a track of the number of active IE pairs that uses the link. Finally, routers check all the packets to see wither they are probes. Probe messages have a special tag which is a facility that allow to avoid a complete processing of the headers, which will require a customization of the queue for layer 1 and 2 protocol. Once a probe is enqueued, the fields are updated according to the equations ??

## 4.4 Simulations using the framework

Now that the different element of the framework have been described, we are going to focus on some additional elements that have been used to run the simulations.

### 4.4.1 Application:

A new application module was implemented that allows to have multiple flows running between the end host. This module is constituted from two parts: a client application that initiate the flow and ask the server to send a certain amount of data. The application module handles multiple flows simultaneously and ensures that the flows when a flow finishes sending the amount of data specified at the beginning a new flow is created to replace it. The flow size is a random variable that could be configured from the simulation script. In particular, for the simulation presented in the next chapter, the flow size had a Weibull distribution.

### 4.4.2 Helpers

Helpers in ns-3 are modules according the design patern of wrappers that facilitate the use of the software core modules during the simulations. Therefore, helpers were developped for the new routing modules of the framework, that are used for adding the paths and the general configuration of the module, ns-3 net device helper allows to specify which queue to implement and so this was used to implement TexcpQueue in the core routers. However, some modifications were required to allow the use of the traffic shaper net device in the external interfaces of the ingress routers.

### 4.4.3 Tracing

Utilization and loss are the key variables that our study interest in. Tracing focused on two elements , the core routers and especially the bottleneck where the dropping rate and the utilization variable are located. And the ingress point to track the aggregation of the LEX marking and the splitting that the ingress routers are desiring.

## 4.5 Implementation highlights, limitations and assumption

Chapter 5

# Simulation Results and Analysis

## 5.1 Analysis of PREF as a splitting mechanism

## 5.2 Load balancing VS congestion balancing

## 5.3 Analysis of PREFLEX balancing algorithm

## 5.4 Using traffic shapers for controlling the congestion

Chapter 6

# Conclusions and future work

## 6.1 conclusion

## 6.2 summary of contributions

## 6.3 Future Work

Appendices

# Chapter 7

# Appendices

# Bibliography

[RFC 3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao. *Overview and Principles of Internet Traffic Engineering*. RFC 3272, IETF, May 2002. (pp 4, 14, )