White Paper

Simulating Classes of Services over an IP/MPLS Backbone

VPN Case Study

Joël Repiquet
01/2005

http://www.joelrepiquet.com

Contents

List of Figures

List of Tables

List of Charts

1 Introduction

The object of this paper is to present a case study based on a simulation in order to provide a practical perception of how a VPN Service Provider runs and maintains delay-sensitive applications, such as video or voice, over its IP/MPLS network. Besides potentially many forms of layer-2 and layer-3 VPN services, it is likely that an SP network also would offer Internet transit and/or access services. The key information for an SP regarding the traffic entering its network at each access point are, as part of the service level agreement with the customer, the class of service along with possibly the committed and peak rates. Bandwidth availability is a necessary condition for ensuring QoS and could also be a sufficient one for an SP network that would be strictly oversized, at any time, with respect to the offered traffic. Unfortunately, this situation is very unlikely, even with an SP that would totally control its fiber and transmission infrastructure. Here are some reasons:

- Links or nodes become unavailable during some time, either for planned maintenance, or due to unexpected failures.
- Services are increasingly proposed with lower-priced burst rates well above the committed rate and even up to the access port capacity, especially with Ethernet interfaces.
- There is a minimum time needed for planning and realizing the capacity upgrade of the network.

There is definitely a need for differentiating the services across an SP backbone. We will refer to DiffServ terminology for defining the three “classes of service” or “aggregates” that are commonly used. The class with the most stringent requirements in terms of QoS, *Expedited Forwarding* or “EF”, is associated with services proposing a committed rate only, and very low delay variation – i.e., jitter. Packets belonging to this EF class are served with the highest priority and EF traffic should not be overbooked in order to ensure bandwidth availability, and therefore no queue delaying, at any time. The *Assured Forwarding* or “AF” class is associated to services that propose different levels of quality (and pricing) based on the rate. Typically, a committed rate is offered along with one or two peak rates, or burst sizes. AF flows are less sensitive to jitter, but they should have their throughput maintained from end-to-end whenever the committed rate is respected. In case they exceed this committed rate, they are eligible to packet dropping, even prior to congestion. Packet dropping at an opportune time is the basis of congestion avoidance mechanisms for TCP flows. There can be several instances of AF class. Finally, the *Best Effort* or “BE” class is the default one and has no specific requirements.

DiffServ (DS) and MPLS, especially traffic engineering (TE), are the main technologies that enable QoS with IP. We are not reviewing in detail DS and MPLS TE in this paper but our case study, focused on the forwarding performance aspects, assume situations that result in their application, either separately or in a combined way.

As a DiffServ domain, the Service Provider (SP) backbone controls the ingress customer traffic at its edges by classifying the IP flows into aggregates and, depending on the service level agreement with the customer, possibly conditioning this traffic by a metering function potentially followed by policing, shaping or re-marking functions. The appropriate forwarding treatment is then applied throughout the domain up to the egress point. Classes are typically associated with queues at each port in the network. These queues share the port capacity under the control of a scheduler, whose role is to select the next packet to be sent over the link. Each class is assigned a percentage of the port capacity and as a result it should be noticed that the BE class, in spite of its lowest priority, will be able to use its part preferably to any excess traffic from other classes. Memory attached to each queue is also managed according to congestion detection mechanisms that anticipate and prevent a full congestion. In practice, there is also another class, sometimes referred to as *network control* or “NC”, reserved for signaling traffic but this has not been simulated, for alleviating the analysis.

As an MPLS domain, the SP assigns paths to IP flows and is therefore able to manage bandwidth consistently inside its backbone. Traffic engineering enables the SP to gather individual flows in traffic trunks (TTs) between an ingress and egress nodes and have them dynamically routed depending on a number of constraints, essentially related to bandwidth. DiffServ Traffic Engineering is an emerging solution that combines the benefits of both well-established DS and TE technologies, and will be considered in our case study.

Simulations based on software programs stay in the theoretical domain and present the advantage of offering extremely accurate results, on a short but representative time period. In effect, the simulator does not need to be real-time by itself and as much information as necessary can be easily gathered at each time unit. Nevertheless, the real-time nature of traffic can be reproduced a posteriori thanks to samples collected at regular intervals that can lead to graphs or even an animation.

This paper is organized as follows:

- First the IPVCoSS simulator is presented shortly. It is a proprietary tool that was preferred to other available simulators such as NS2, just to have the full control of options, traces and any output data.
- Then the case study consisting of 10 sites for 3 VPNs over a backbone made up of 7 routers is presented under various aspects, followed by an in-depth analysis of a first run with 14 flows. This initial run is then followed by 5 variants.
- A specific trial completes the case study. Its purpose is to highlight the conditions that create jitter in a statistically-multiplexed environment such as IP, and show that jitter for EF flows remains in largely acceptable limits.
- Finally, two annexes illustrate with traces the implementation by IPVCoSS of fundamental mechanisms: TCP congestion control and packet scheduling.

2 IPVCoSS Simulation Tool

IPVCoSS (IP VPN CoS Simulator) is a stand-alone program, written in C and without any prerequisites, that enables you to set up a network topology and inject traffic at ingress points. IPVCoSS is focused on IP data forwarding performance and assumes that data paths are pre-established. It can be configured to handle classes of service and to simulate one or more IP VPNs over a Service Provider backbone. The network configuration is based on a data model and the data structures are built at initialization time. During its active phase, IPVCoSS runs a loop fictitiously cadenced at 100 nanoseconds (one tenth of microsecond) and at each tick, it analyses and ensures the progression of IP packets at each node and port. The transfer activity is triggered by the generation of IP flows, according to parameters such as volume, throughput and packet size. In output, IPVCoSS produces QoS final reports, QoS regular samples and, if required, a chronological trace of events.

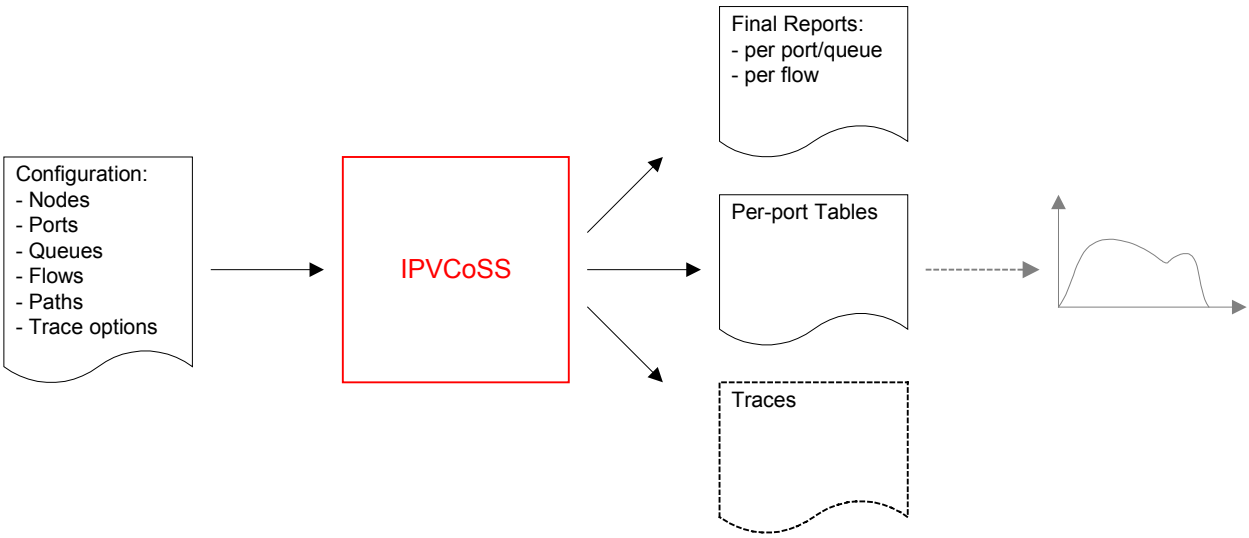


Figure 1: IPVCoSS basic principles

The key elements of a network configuration are the ports. Ports are unidirectional and belong to a node. They are identified by a number, and their main attribute is the interface type, determining the rate. The 100-nanosecond clock accuracy enables IPVCoSS to process correctly IP interfaces ranging from E1 (2 Mbps) to STM-16 (2.5 Gbps). There is no explicit notion of links between nodes and the network topology is defined by the mapping between ports:

- An input port (IPORT) is a traffic ingress point at which a single IP flow, identified by a letter, is generated. An IPORT maps to an output port within the same node.
- An output port (OPORT) is an internal OPORT when it maps to one or more OPORTs in another node. The distance to this adjacent node is a key parameter of an internal OPORT.
- An output port is an egress OPORT when it does not map to another OPORT.

The conventional IP routing scheme is not used and paths are pre-determined by simple filtering, at OPORT level, based on flow identifiers. These paths represent MPLS label switched paths (LSP) in a core network.

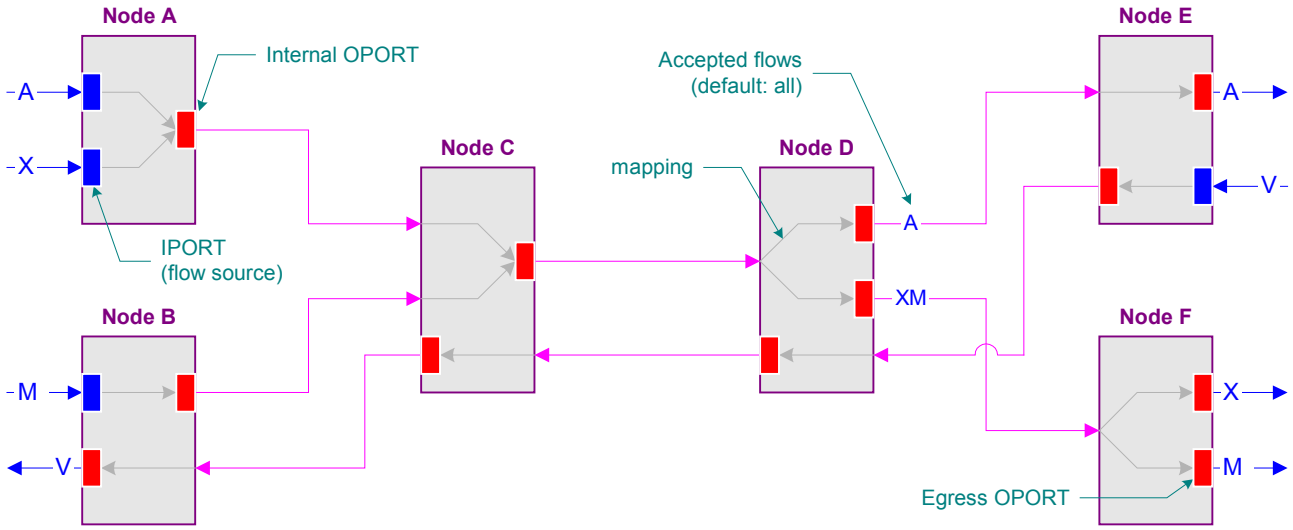


Figure 2: IPVCoSS – Nodes and Ports

For simplification and clarity, we have avoided having multiple logical interfaces per physical interface. ATM, Frame Relay and VLAN layer-2 framing is therefore not considered and the encapsulation type is automatically derived from the interface type:

- PPP HDLC framing for PDH and SDH interfaces: E1, E3, DS3 and STM-1, STM-4, STM-16.
- Ethernet (without VLAN) framing for Ethernet interfaces: FE, GE.

Figure 3 and Figure 4 show respectively the framing structures with Ethernet and PPP. With Ethernet the overhead reaches 26 bytes, with a minimum interframe gap of 12 bytes. With PPP HDLC framing, the overhead is only 9 bytes, assuming a frame check sequence (FCS) field of 4 bytes, and there is no interframe gap.

Figure 3: Ethernet framing

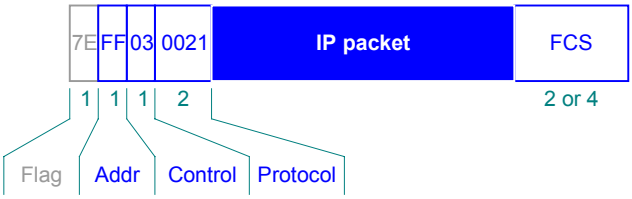


Figure 4: PPP HDLC framing

The end-to-end chain of an IP packet from source to destination is shown in Figure 5. The equipment that host the IP source or destination are not modeled by IPVCoSS but they are supposed to be co-located respectively with the ingress IPORT and the egress OPORT, which are IPVCoSS reference end-points for an IP flow. At each OPORT along the flow path, and eventually at the egress OPORT, the delay, jitter and throughput QoS parameters related to any IP packet are measured, and can be compared to their initial value when generated at the ingress IPORT. These measurements are summarized in per-flow final reports.

When an IP packet (that includes the IP header) crosses a port, it is encapsulated in a frame that depends on the interface type. Besides, according to BGP/MPLS VPN architecture, one or two 4-byte MPLS shim headers may be inserted before the IP header, depending on the role of the service provider node.

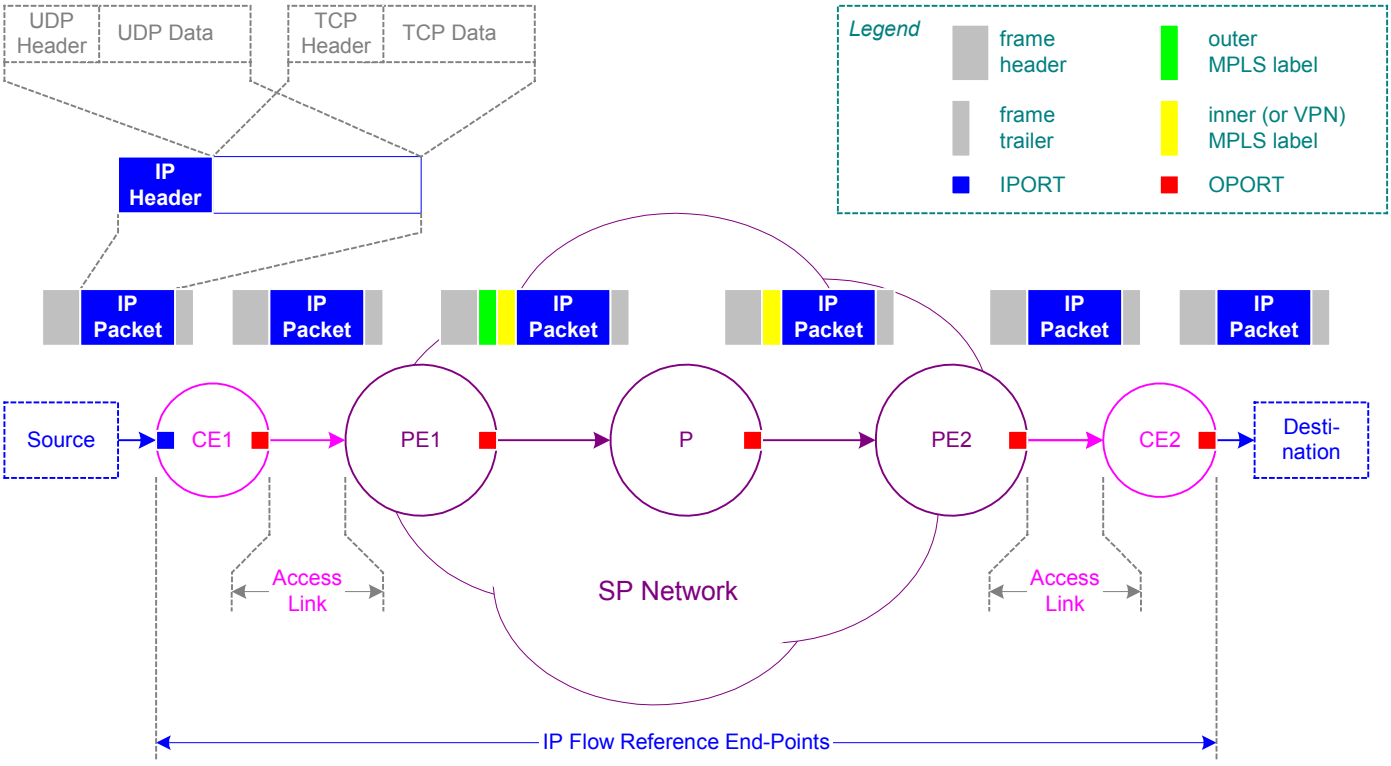


Figure 5: End-to End IP Flow chain

Here are the attributes that define an IP flow. Some attributes characterize the IP flow and are permanent from end to end, while the other ones are used at generation time only.

Name	Value	Description	
Identifier	Single printable character	Besides identifying the IP flow, this character is combined (in traces) with the packet serial number for identifying an IP packet. By convention, and although this is not a constraint, in a configuration using classes of services, the following ranges of letters will be used: ■ A - L for BE flows ■ M - U for AF flows ■ V - Z for EF flows	
Class of Service		Based on DiffServ “aggregate” terminology and assuming 3 physical queues per OPORT:	
	EF	Expedited Forwarding	low delay, low jitter, no loss, and highest priority (but not strict priority)
	AF	Assured Forwarding	guaranteed rate while throughput respects the committed rate, but lower probability whenever excess traffic; this class enables several levels of service while preserving per-flow packet ordering (it is an Ordered Aggregate)
	BE	Best Effort	the default class
Transport Protocol	UDP	UDP flows have no specific processing.	
	TCP	TCP-based flows are responsive to traffic load and congestion. TCP congestion control mechanisms, normally handled at host system level, are simulated at ingress and egress ports level. IPVCoSS supports slow start and congestion avoidance mechanisms according to RFC 2581 ([16]).	
Packet Size	46 - 1500 bytes	The packet size includes the IP header, and the allowed values are based on conventional Ethernet limits. This parameter specifies a fixed size for an isochronous flow, or a maximum size for a variable-rate flow.	
Traffic Profile	CBR	Isochronous flow with fixed length packets generated at regular intervals, depending on the required throughput	
	VBR	Packets are generated at random with a variable size and at irregular intervals. The size varies between 46 bytes and the value specified in “packet size”. There is also an option for having fixed size packets. The interval times are irregular but are constrained by the required throughput that is ensured, by adjustment, for each 3-full-sized-packet volume. With this profile, a SAVE / REPLAY option is offered for enabling the reproduction of the same variable flow from one run to another.	
Throughput	in Mbps	The throughput determines the interval between packets. Depending on whether this interval value is rounded, or not, the throughput will be strictly respected or very closely approached.	
Volume	Number of full-sized packets	For an isochronous traffic, IPVCoSS will generate the required number of packets while with a variable traffic, a larger number of packets will be effectively generated.	
Number of Occurrences		This optional attribute enables you to generate the required volume several times, with a fixed gap interval between two consecutive occurrences. This is useful for generating traffic bursts, instead of a continuous flow. The required throughput is ensured for each occurrence.	
Gap Interval	in milliseconds	Time interval between two consecutive volumes of traffic.	
Initial Tick	in milliseconds	Starting time for the generation of the first bit of the first packet of the IP flow.	

2.1 Serialization delay samples

The delay experienced by an IP packet for a flow crossing a single node consists in the serialization delays at the ingress IPORT and egress OPORT as well as the processing delay (a fixed value configured at node level) and the queuing delay, which is examined in more detail later in the case study. The serialization time is dependent on the interface rate and the packet size, more exactly the size of the packet's frame.

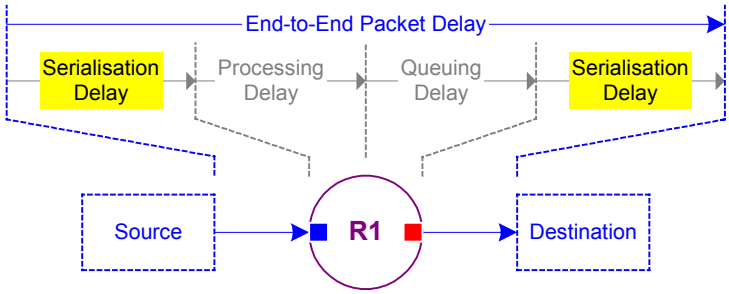


Table 1 mainly provides the serialization times in microseconds – with a precision of 100 nanoseconds – for some typical packet size values, according to PDH, SDH and Ethernet interface types. The real and useful rates are also provided for each interface type. As already mentioned, we do not consider multiple logical interfaces and assume a PPP encapsulation mode for PDH and SDH interfaces. For information, the maximum reachable IP throughput is shown, taking into account the frame overhead and the possible minimum gap interval between frames.

Interface Type	Physical Rate in b/s	Useful Rate in b/s	Packet Length in bytes	Frame Length in bytes	Frame Length in bits	Serialization Time in microseconds	Maximum IP Throughput in Mbit/s
E1	2,048	2,048	1,500	1,509	12,072	5894.6	2.04
			1,000	1,009	8,072	3941.5	2.03
			500	509	4,072	1988.3	2.01
			46	55	440	214.9	1.71
E3	34,368	34,368	1,500	1,509	12,072	351.3	34.16
			1,000	1,009	8,072	234.9	34.06
			500	509	4,072	118.5	33.76
			46	55	440	12.9	28.53
DS3	44,736	44,210	1,500	1,509	12,072	273.1	43.94
			1,000	1,009	8,072	182.6	43.81
			500	509	4,072	92.2	43.38
			46	55	440	10.0	36.80
FE	100,000	100,000	1,500	1,526	12,208	122.1	97.48
			1,000	1,026	8,208	82.1	96.27
			500	526	4,208	42.1	92.81
			46	72	576	5.8	54.12
STM-1	155,520	149,760	1,500	1,509	12,072	80.7	148.70
			1,000	1,009	8,072	53.9	148.42
			500	509	4,072	27.2	147.06
			46	55	440	3.0	122.67
STM-4	622,080	599,040	1,500	1,509	12,072	20.2	594.06
			1,000	1,009	8,072	13.5	592.59
			500	509	4,072	6.8	588.24
			46	55	440	0.8	460.00
GE	1,000,000	1,000,000	1,500	1,526	12,208	12.3	967.74
			1,000	1,026	8,208	8.3	952.38
			500	526	4,208	4.3	909.09
			46	72	576	0.6	525.71
STM-16	2,488,320	2,396,160	1,500	1,509	12,072	5.1	2352.94
			1,000	1,009	8,072	3.4	2352.94
			500	509	4,072	1.7	2352.94
			46	55	440	0.2	1840.00

Table 1: Serialization time according to port rate and packet size

2.2 Propagation delay samples

When an IP packet crosses 2 nodes, the serialization delay on the output port must not be accounted in input of the adjacent node. The distance of the link between the two nodes induces a propagation delay that is directly tied to the distance and is independent of both packet size and interface capacity. This delay cannot be avoided when the source and destination are located at far distant sites. It then depends on the topology of the SP network that might not be optimal with respect to these end points.

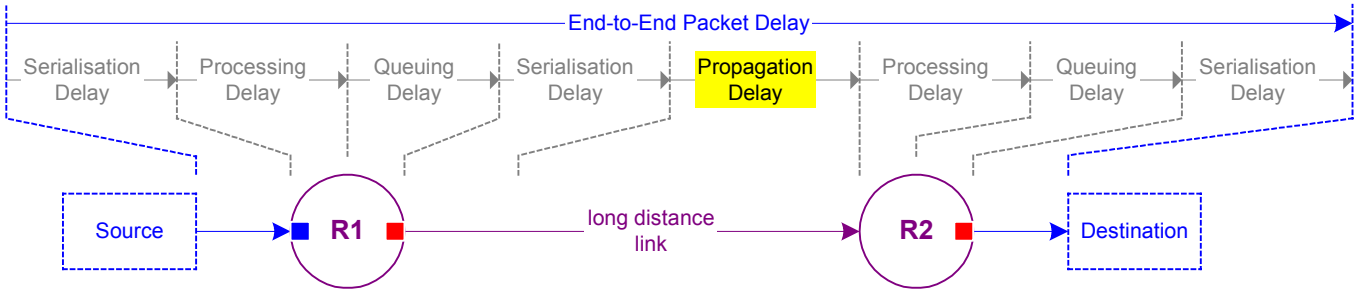


Table 2 provides propagation delay values according to several distances. The propagation delay is given by the formula:

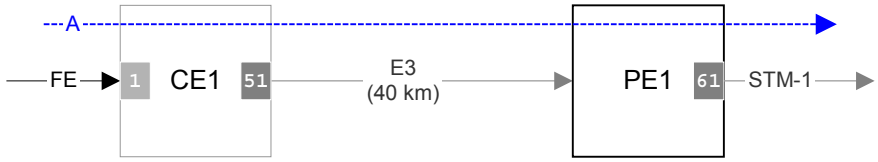
$$\text{Propagation Delay (in µs)} = \frac{\text{Distance (in km)}}{299{,}300 \text{ km} \times 0.6} \times 1000$$

Distance	Propagation Delay	Distance	Propagation Delay
1 km	5.6 µs	700 km	3,898.0 µs
1 mile	9.0 µs	800 km	4,454.8 µs
2 km	11.2 µs	500 miles	4,479.9 µs
3 km	16.8 µs	900 km	5,011.7 µs
2 miles	18.0 µs	600 miles	5,375.9 µs
4 km	22.3 µs	1,000 km	5,568.5 µs
3 miles	26.9 µs	700 miles	6,271.8 µs
5 km	27.9 µs	800 miles	7,167.8 µs
6 km	33.5 µs	900 miles	8,063.8 µs
4 miles	35.9 µs	1,000 miles	8,959.7 µs
7 km	39.0 µs	2,000 km	11,137.0 µs
8 km	44.6 µs	3,000 km	16,705.5 µs
5 miles	44.8 µs	2,000 miles	17,919.4 µs
9 km	50.2 µs	4,000 km	22,274.0 µs
6 miles	53.8 µs	3,000 miles	26,879.1 µs
7 miles	62.8 µs	5,000 km	27,842.5 µs
8 miles	71.7 µs	6,000 km	33,411.0 µs
9 miles	80.7 µs	4,000 miles	35,838.8 µs
100 km	556.9 µs	7,000 km	38,979.5 µs
100 miles	896.0 µs	8,000 km	44,548.0 µs
200 km	1,113.7 µs	5,000 miles	44,798.5 µs
300 km	1,670.6 µs	9,000 km	50,116.5 µs
200 miles	1,792.0 µs	6,000 miles	53,758.2 µs
400 km	2,227.4 µs	10,000 km	55,685.0 µs
300 miles	2,688.0 µs	7,000 miles	62,717.9 µs
500 km	2,784.3 µs	8,000 miles	71,677.6 µs
600 km	3,341.1 µs	9,000 miles	80,637.3 µs
400 miles	3,583.9 µs	10,000 miles	89,597.0 µs

Table 2: Propagation delay according to distances, in km and miles

2.3 IPVCoSS Trace example

Here is the trace of an IP Flow “A” that is isochronous, at a rate of 20 Mbps and a packet size of 1500 bytes. This flow is generated at node CE1 on IPORT#1 and is sent on OPORT#51 towards node PE1. The E3 link that connects CE1 to PE1 is 40 km long and it induces a propagation delay of around 223 microseconds. There is no congestion and absolutely no jitter.



Tick value in microseconds,
Precision: 100 nanoseconds

Node name
Port number
Event type
IP packet identifier:
flow letter + serial number
(one '*' per previous hop)
Event-related parameters

Events:

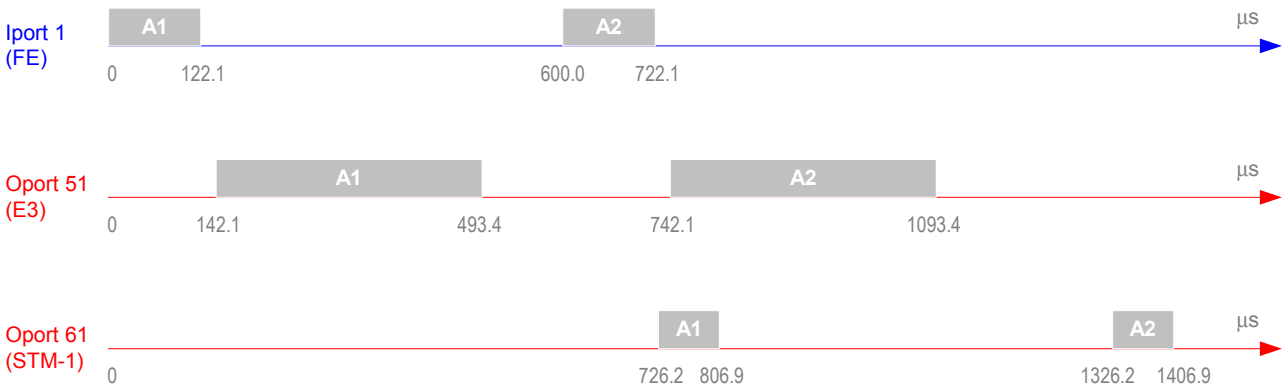
si	start input
ei	end input
qo	queue output
so	start output
eo	end output

Parameters:

pkln	packet length (bytes)
ovhd	frame overhead (bytes)
iser	input serialization time (µs)
oser	output serialization time (µs)
gd	global delay (µs)
fd	forwarding delay (µs)
qd	queuing delay (µs)
pd	propagation delay (µs)
j	jitter (µs)
ithr	input throughput (Mbps)
othr	output throughput (Mbps)

```
  0.0  CE1  i1  : si    A1    pkln:1500  ovhd:26 iser:122.1
122.1  CE1  i1  : ei    A1    gd:122.1   fd:20.0       remvol:1500
142.1  CE1  o51 : qo    A1    gd:142.1   Q:BE (00.35%)
       CE1  o51 : so    A1    pkln:1500  ovhd:9  oser:351.3  qd:0.0
                              gd:142.1   Q:BE (00.00%)                j:0.0
493.4  CE1  o51 : eo    A1    gd:493.4   pd:222.8   next-hop:PE1
600.0  CE1  i1  : si    A2    pkln:1500  ovhd:26 iser:122.1  ithr:20.00
716.2  PE1  o61 : ei   *A1    gd:716.2   fd:10.0    prev-hop:CE1  up-oport:51
722.1  CE1  i1  : ei    A2    gd:122.1   fd:20.0       remvol:0
726.2  PE1  o61 : qo   *A1    gd:726.2   Q:BE (00.08%)
       PE1  o61 : so   *A1    pkln:1500  ovhd:9  oser:80.7   qd:0.0
                              gd:726.2   Q:BE (00.00%)                j:0.0
742.1  CE1  o51 : qo    A2    gd:142.1   Q:BE (00.35%)
       CE1  o51 : so    A2    pkln:1500  ovhd:9  oser:351.3  qd:0.0
                              gd:142.1   Q:BE (00.00%)       othr:20.00 j:0.0
806.9  PE1  o61 : eo   *A1    gd:806.9
1093.4 CE1  o51 : eo    A2    gd:493.4   pd:222.8   next-hop:PE1
1316.2 PE1  o61 : ei   *A2    gd:716.2   fd:10.0    prev-hop:CE1  up-oport:51
1326.2 PE1  o61 : qo   *A2    gd:726.2   Q:BE (00.08%)
       PE1  o61 : so   *A2    pkln:1500  ovhd:9  oser:80.7   qd:0.0
                              gd:726.2   Q:BE (00.00%)       othr:20.00 j:0.0
1406.9 PE1  o61 : eo   *A2    gd:806.9
```

Here is a diagram that provides a visual perception of the occupancy of each port. As this can be seen in the trace here above, the serialization time of the 1500-byte IP packet in its Ethernet frame is 122.1 microseconds, while the same packet in its PPP frame on an E3 link is 351.3 microseconds, and only 80.7 microseconds on an STM-1 link.



3 IP VPN / CoS Case Study

Figure 6 shows an SP (service provider) network that offers VPN services to 3 customers: Red, Blue and Green. The internal backbone links connecting the edge routers to the core routers are limited to STM-1 in order to cope with the traffic throughput used in the context of this case study. The other backbone links between core routers are either STM-1 or STM4, thus creating the conditions for over-sizing or under-sizing. This core network topology ensures a minimum of resiliency since each PE (Provider Edge) router is connected to two P (Provider core) routers and each P router is in turn connected to two other P routers. However, the links in dotted lines will not be considered for this case study, not to complicate the simulation survey. The distances shown between the backbone routers represent a realistic regional network, either national or international. We consider primarily MPLS VPN services over this network but other services such as Internet Transit could be offered as well. In some scenarios we will introduce extra flows crossing the SP network and we can consider that these flows could belong to traffic related to other VPN, or Internet Transit, services.

We have not considered co-locations of a CE (customer edge) at a PE site. All the VPN sites are connected to the SP network via access links. This is simply for clarifying the picture of the overall network.

With the Simulator, links are unidirectional but we could easily have traffic in the two directions between two routers. However, for clarity, traffic will originate in sites on the left side and terminate in sites on the right side, obviously within their respective VPN.

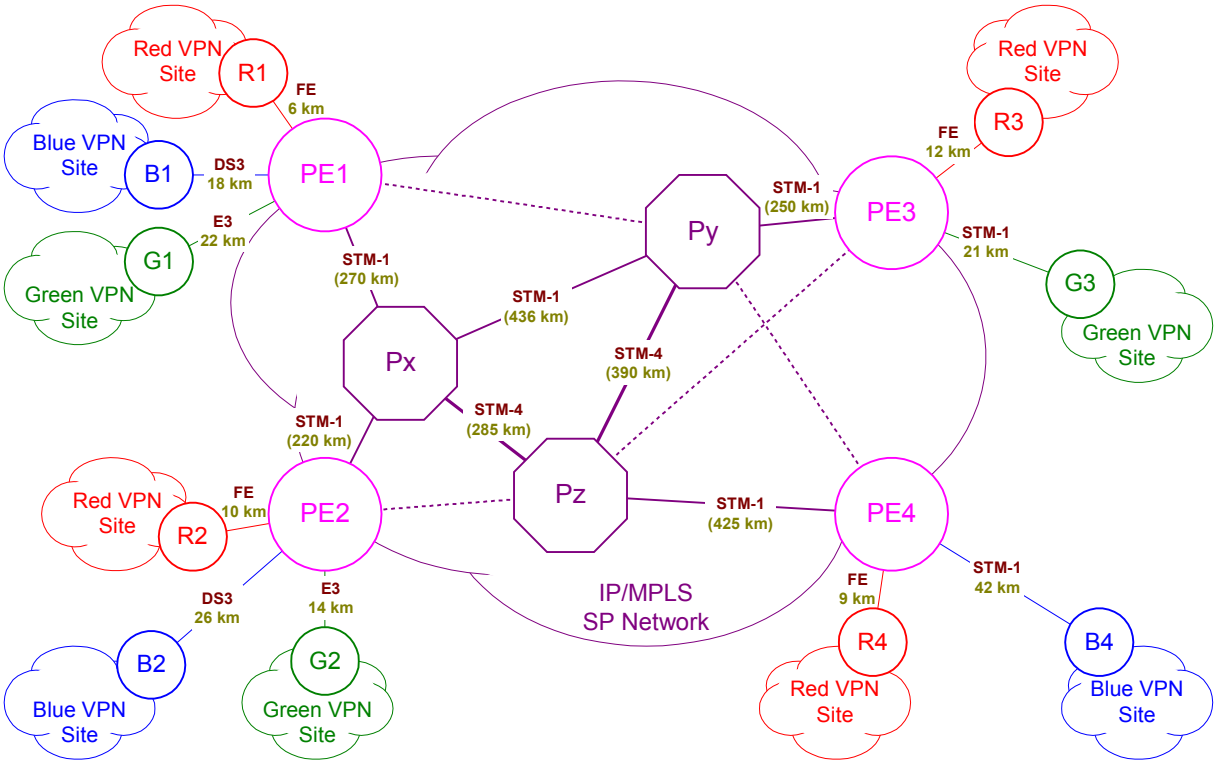


Figure 6: VPN Case Study – Topology

The purpose of this case study is to illustrate the CoS (Class of Service) mechanisms ensuring the required QoS, especially jitter, for EF flows such as MPEG-2 video or VoIP. We will first review in detail the initial scenario where there is some congestion at a few points but no packet loss, and QoS requirements for each class are fulfilled. Then we will run several variants and analyze the impact on the traffic flows.

Each scenario represents a snapshot of the same basic offered traffic during a short period, in a given network configuration. Our observation relates to the traffic forwarding only. IPVCoSS does not handle any signaling function that would enable us to understand the impact on the traffic flows of transiting from one situation to another, for instance in case of link failure.

It should be noticed that, although the observation period is short (around 50 ms) it is highly meaningful and representative of live situations. Anyway, the values of parameters such as queue depths are adapted to the case study and lower than they would be in a real configuration.

3.1 Physical Topology

Figure 7 provides a representation of the physical topology of the network built via IPVCoSS. This is stable information that will not change for subsequent scenarios. All the CEs (customer edge) have Fast Ethernet ports on the site side. These CEs could be routers or switch-routers, or even switches.

The access links connecting the sites to the SP network are either leased lines (E3, DS3, STM-1) or Ethernet services with FE ports.

The rates commonly associated to the physical interfaces are recalled hereunder, but more accurate values of physical and useful rates are given in Table 1 on page 8.

E3	34 Mbps
DS3	45 Mbps
FE	100 Mbps
STM-1	155 Mbps
STM-4	622 Mbps

Ethernet services may have several possible underlying architectures: switches connected by fiber, ATM bridges over SDH, an ATM network. Although the physical interfaces are Fast or Giga Ethernet, the service can be subscribed for a throughput lower than the port capacity. The R2-to-PE2 ingress access link and PE3-to-R3 egress access link will be rate limited down to 60 Mbps in our case study.

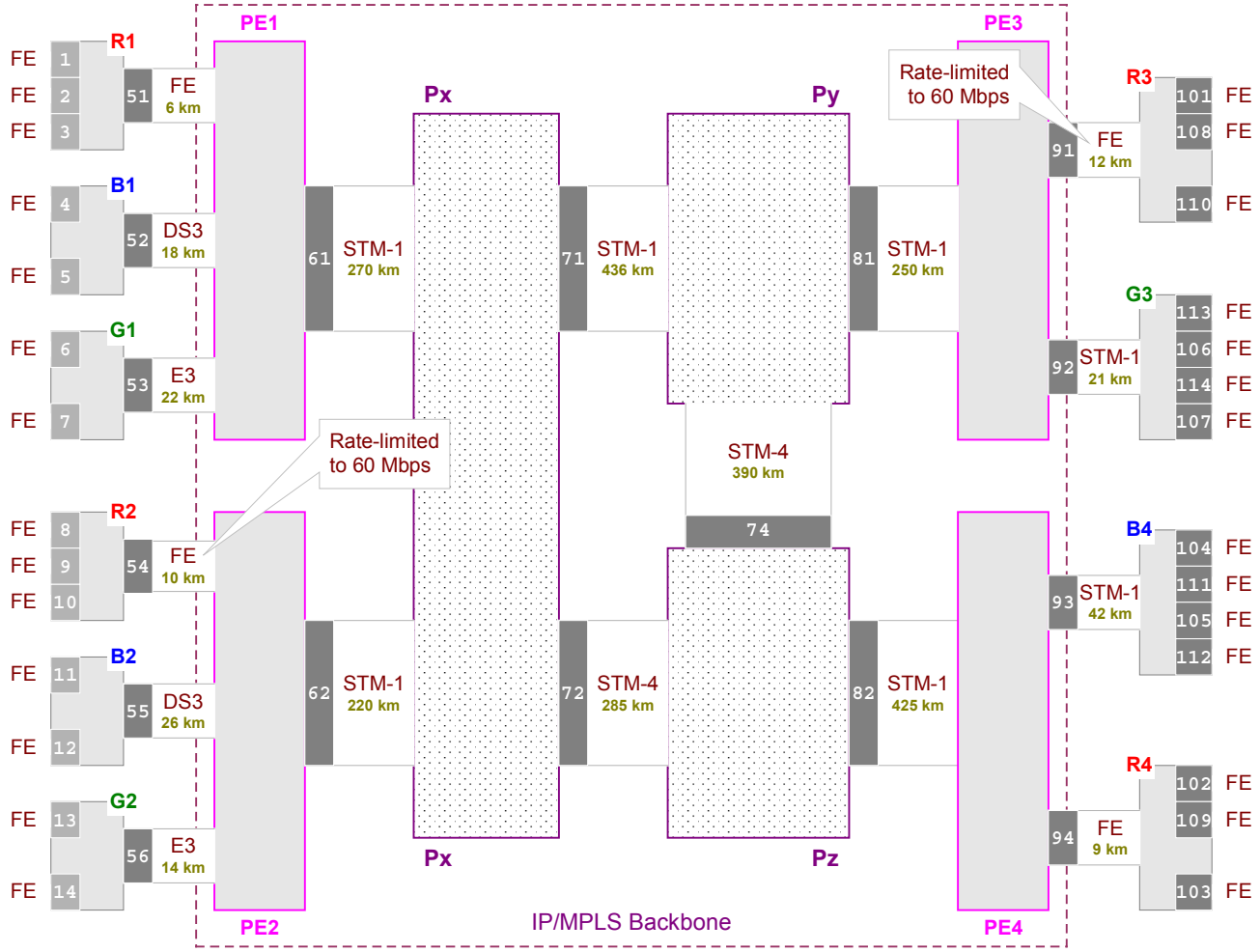


Figure 7: VPN Case Study – Physical Ports

Note 1: It is likely that in a real network, for offering Fast Ethernet accesses, Ethernet switches would be placed in front of PE routers via Gigabit interfaces, thus adding a node between the CE and the PE.

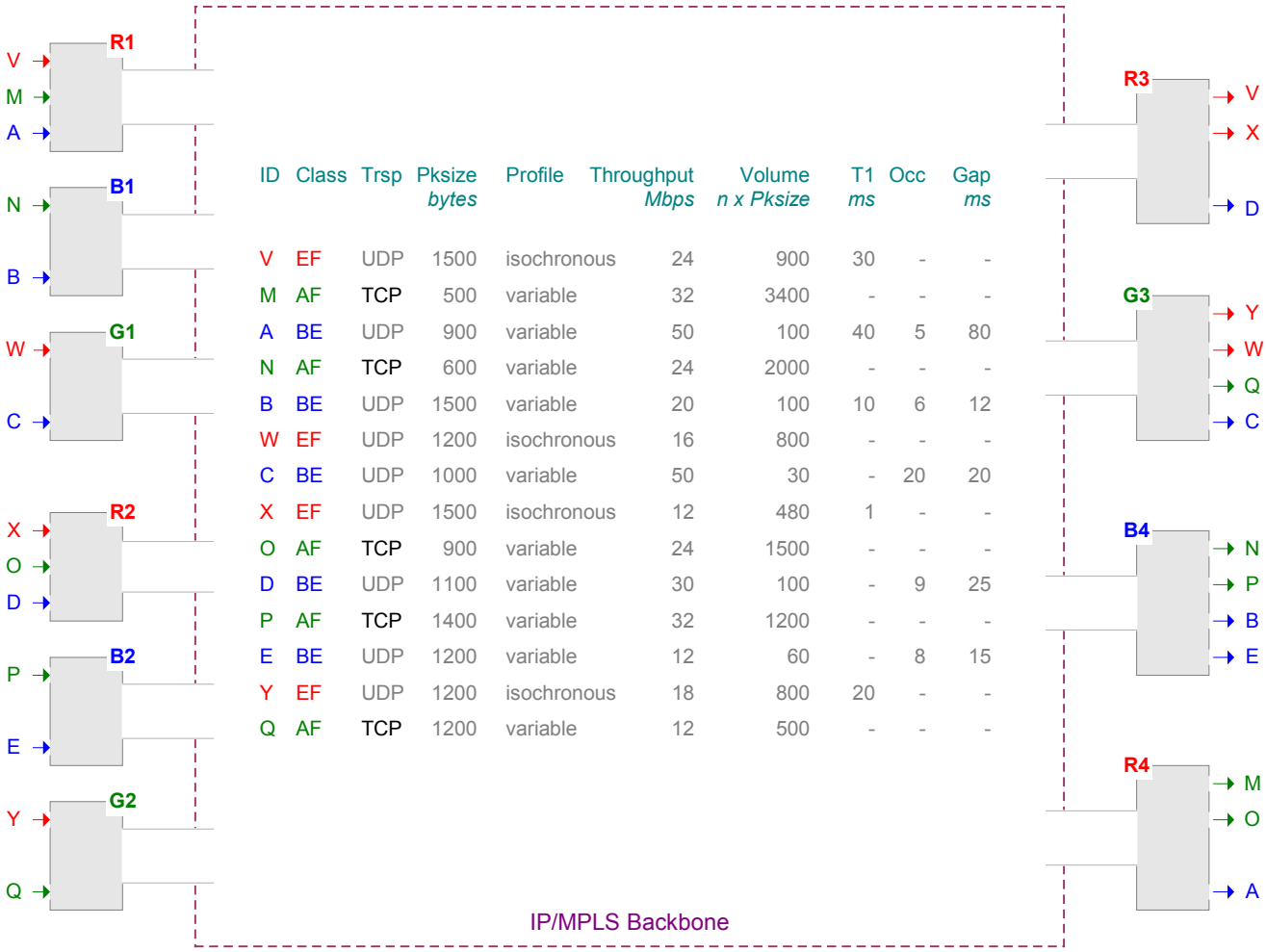
Note 2: From an MPLS perspective, PE and P nodes are Label Switching Routers (LSRs).

3.2 Traffic Flows

Figure 8 shows the ingress and egress points of the generated IP flows. We can assimilate them to the source and destination of each micro-flow by considering that a host (or for instance a video encoder/decoder) is immediately connected to the CE, as previously illustrated in Figure 5.

The characteristics of the 14 IP flows that will cross the network are summarized in the table placed in the center of Figure 8. In the context of this case study we have consistently assigned common attributes for all the flows of a same class. in some way, we have assigned a role to each class. For instance, all AF flows are based on TCP, however this is not to be understood as a characteristic associated to AF class.

- EF flows are based on UDP transport and are isochronous. They require very low jitter and their initial throughput must be maintained.
- All our AF flows are based on TCP transport and therefore TCP slow start will be normally applied as well as congestion avoidance mechanisms if any. These flows have a fixed packet size.
- BE flows are based on UDP transport and generated in several volume occurrences separated by regular gaps, in order to create traffic bursts (since with IPVCoSS, the throughput required for a variable traffic is ensured for each volume corresponding to 3 full-sized packets). The packet size vary between 46 bytes and the maximum packet size.



ID	Class	Trsp	Pksize *bytes*	Profile	Throughput *Mbps*	Volume *n x Pksize*	T1 *ms*	Occ	Gap *ms*
V	EF	UDP	1500	isochronous	24	900	30	-	-
M	AF	TCP	500	variable	32	3400	-	-	-
A	BE	UDP	900	variable	50	100	40	5	80
N	AF	TCP	600	variable	24	2000	-	-	-
B	BE	UDP	1500	variable	20	100	10	6	12
W	EF	UDP	1200	isochronous	16	800	-	-	-
C	BE	UDP	1000	variable	50	30	-	20	20
X	EF	UDP	1500	isochronous	12	480	1	-	-
O	AF	TCP	900	variable	24	1500	-	-	-
D	BE	UDP	1100	variable	30	100	-	9	25
P	AF	TCP	1400	variable	32	1200	-	-	-
E	BE	UDP	1200	variable	12	60	-	8	15
Y	EF	UDP	1200	isochronous	18	800	20	-	-
Q	AF	TCP	1200	variable	12	500	-	-	-

Figure 8: VPN Case Study – Generated Traffic Flows

These VPN flows will be generated in the same way for all the scenarios of this case study. The Save/Replay function applicable to variable flows will be used for ensuring the same traffic profiles at the ingress ports. However, TCP flows in congestion cases will behave according to TCP control mechanisms.

3.3 DiffServ Environment

Figure 9 illustrates the typical key components of the SP Network as a DiffServ (DS) domain. At PE1 and PE2 ingress boundary nodes, incoming traffic streams on each port are classified and, depending on the traffic conditioning specifications agreed between the VPN customer and the SP, possibly metered and re-marked. The classification at ingress boundary nodes is potentially more complex than the classification performed for input traffic at interior DS Nodes (Px, Py, Pz) and egress boundary nodes (PE3 and PE4). BA classification takes simply into account the DS code point (in the IP header DS field or the MPLS header EXP field) while MF classification discriminates a class from several fields in the IP and transport headers. Individual IP flows are ignored within a DS domain and only the aggregates resulting of classification are processed for ensuring the appropriate forwarding behavior at each hop (Per-Hop Behavior or PHB). Typically, the flows eligible to be metered and re-marked at the edge of the SP network are the AF flows. Single-rate and two-rate three color markers, respectively described in [19] and [20], will lead to the marking of packets in an AF flow as Green, Yellow or Red depending on some agreed committed and peak rates. These colors are actually DS code points of a same AF class that can be associated, for example, to three levels of drop precedence.

The DS domain could be extended to the CEs whenever these CEs would be managed by the SP as CPEs. Anyway, whatever the entity responsible for the CE (Customer or SP) there is a need for applying a differentiated forwarding treatment over the access link from CE to PE.

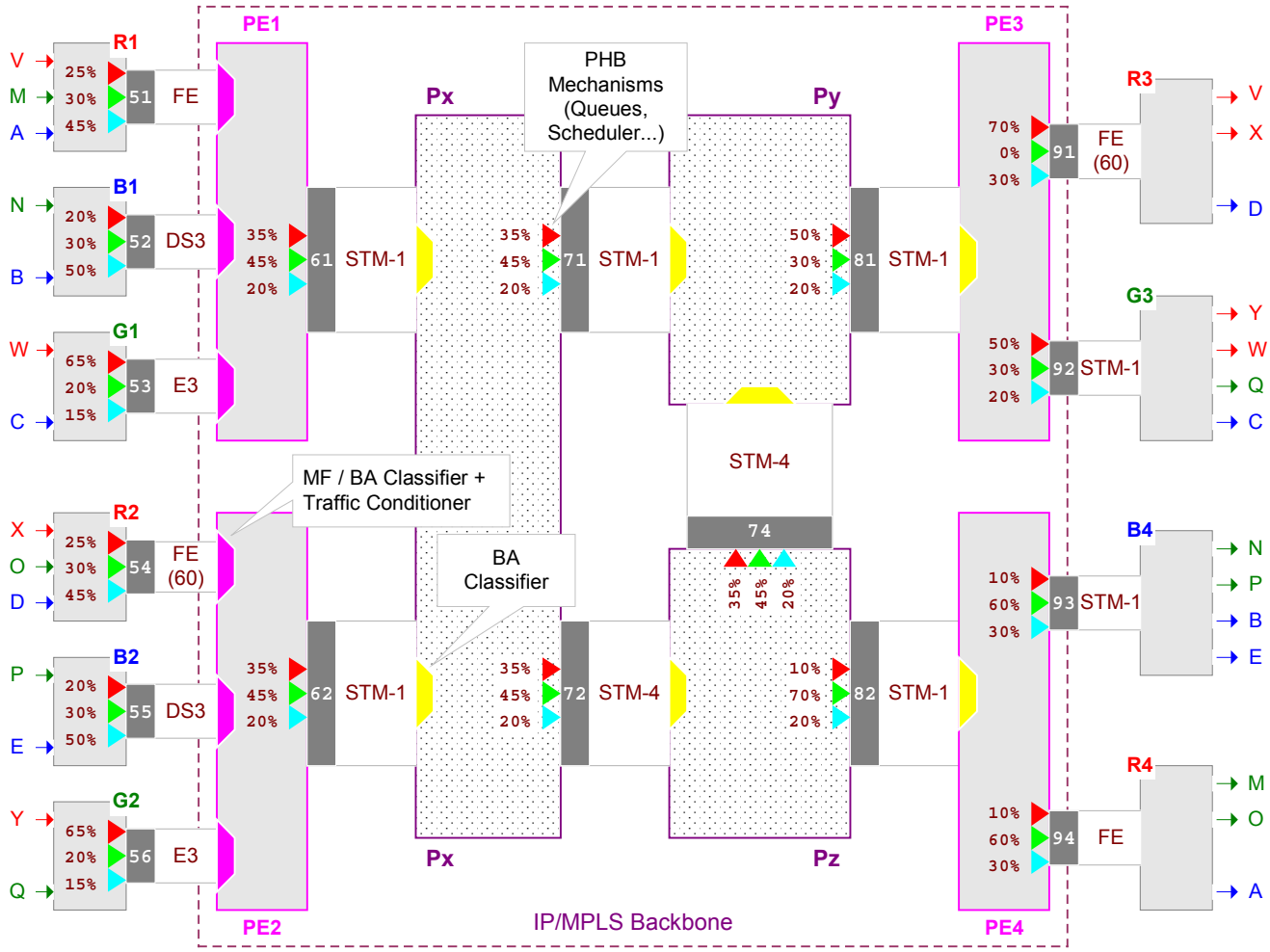


Figure 9: VPN Case Study – DiffServ Environment

Actually, with IPVCoSS, the class is statically defined as a parameter of the flow to be generated. We can therefore consider, although the DS code point (DSCP) itself is not processed, that there is a BA classification in input of each node. Metering of AF flows is not implemented yet but a potential usage of this capability is outlined in section 9 on page 31.

DiffServ standards do not specify how the Per-Hop Behavior should be implemented and instead only describe an externally observable forwarding behavior for each aggregate. IPVCoSS uses at each OPORT a classical packet scheduling based on 3 physical queues (EF, AF and BE) for which are defined a percentage of the port capacity and memory in the form of depth (not shown) expressed in milliseconds. The scheduling mechanism is described and illustrated by a trace in Annex 2:.

Note: In a real world, another queue (and minimum bandwidth) should be dedicated to control traffic (i.e. signaling protocols) that is crucial for enabling the forwarding of customer traffic.

3.4 TE Traffic Trunks Options

As defined in [21] related to traffic engineering (TE), a traffic trunk (TT) is an aggregation of traffic flows belonging to the same "class" (forwarding equivalence class, or FEC) which are forwarded through a common path. In practice, a TT may be characterized by an ingress and egress LSRs, and a set of attributes which determine its behavioral characteristics and requirements from the network. A TT is unidirectional and it is distinct from the LSP through which it traverses. A TT can be moved from one LSP onto another whether the network conditions do not meet anymore the TT requirements.

When TE is used independently of DiffServ, TTs aggregate all the VPN flows between two PEs, whatever their CoS.

We can see, for each TT, the maximum amount of bandwidth per TT. For information, the respective amount of EF, AF and BE traffic is also shown.

However, only the global amount is taken into account for establishing the LSP through which the TT will traverse.

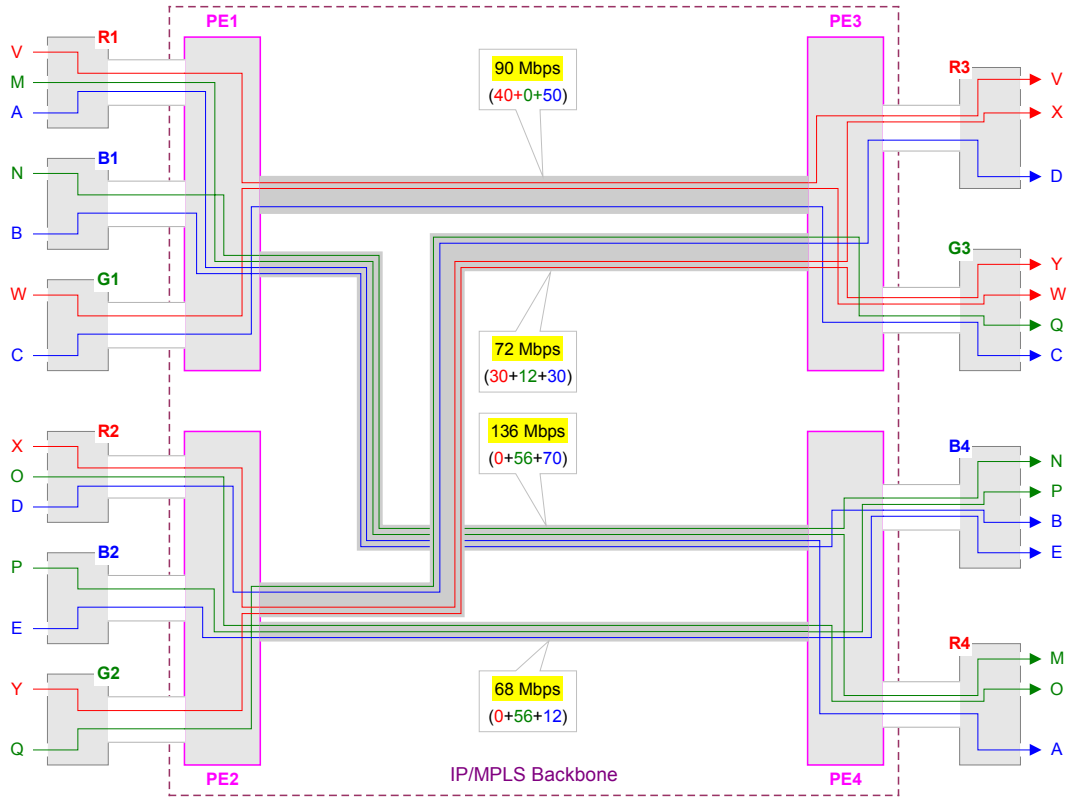


Figure 10: VPN Case Study – Aggregate-based Traffic Trunks

With DiffServ-Aware TE (DS-TE) we could define two class-types:

- one class-type for EF traffic

- another class-type for both AF and BE traffic

The constraint for Traffic Trunks belonging to the "EF" class-type, in terms of resource, would be under-allocation for ensuring bandwidth in any case.

In contrast, Traffic Trunks belonging to the "AF-BE" class-type could remain over-allocated.

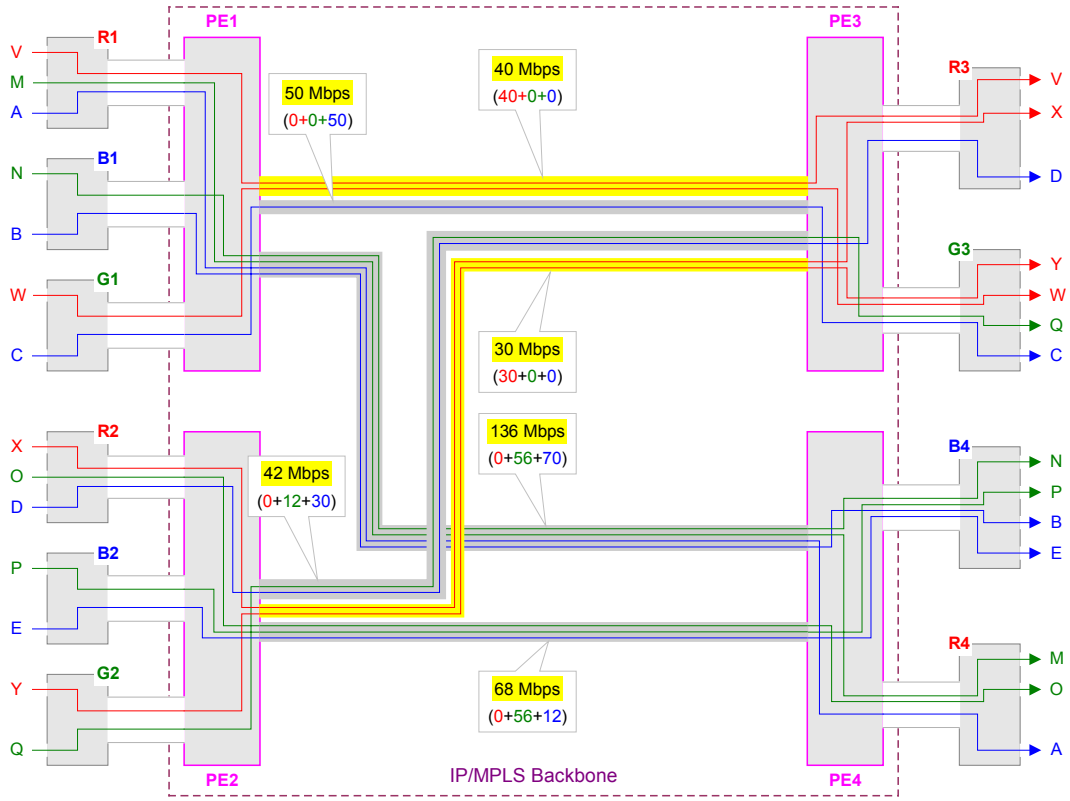


Figure 11: VPN Case Study – Class-based Traffic Trunks

4 VPN Case Study – Initial Run

In the initial situation, traffic trunks are unaware of classes of services and have been mapped on LSPs accordingly. Some oversubscribing was allowed because of the bursty BE traffic and therefore the shortest path has prevailed. Thus, PE2-to-PE3 TT uses a 906 km-long LSP that goes through STM-1 port#71 instead of an 1145 km-long LSP that would go through STM-4 ports 72 and 74.

Purposely, there is some congestion within this network in order to discuss various situations. For this initial run, it is worth to have a detailed view of the traffic profile at each internal OPORT and therefore the remainder of this section will successively review – with graphs and summary reports – the access and backbone links.

As a preview, here are some indications about the traffic load:

- All ingress access links, but port 56, experience some congestion when there are BE traffic bursts
- STM-1 Ports 61 and 62 are heavily loaded, with a little congestion
- Port 72 is an STM-4 and is under loaded
- STM-1 Ports 71 and 81 receive the same flows and are normally loaded
- Port 82 experiences heavy congestion but no packet loss
- All egress links are oversized with the exception of port 91 that is rate-limited to 60 Mbps
- Among the BE flows, flow “C” has many very short bursts

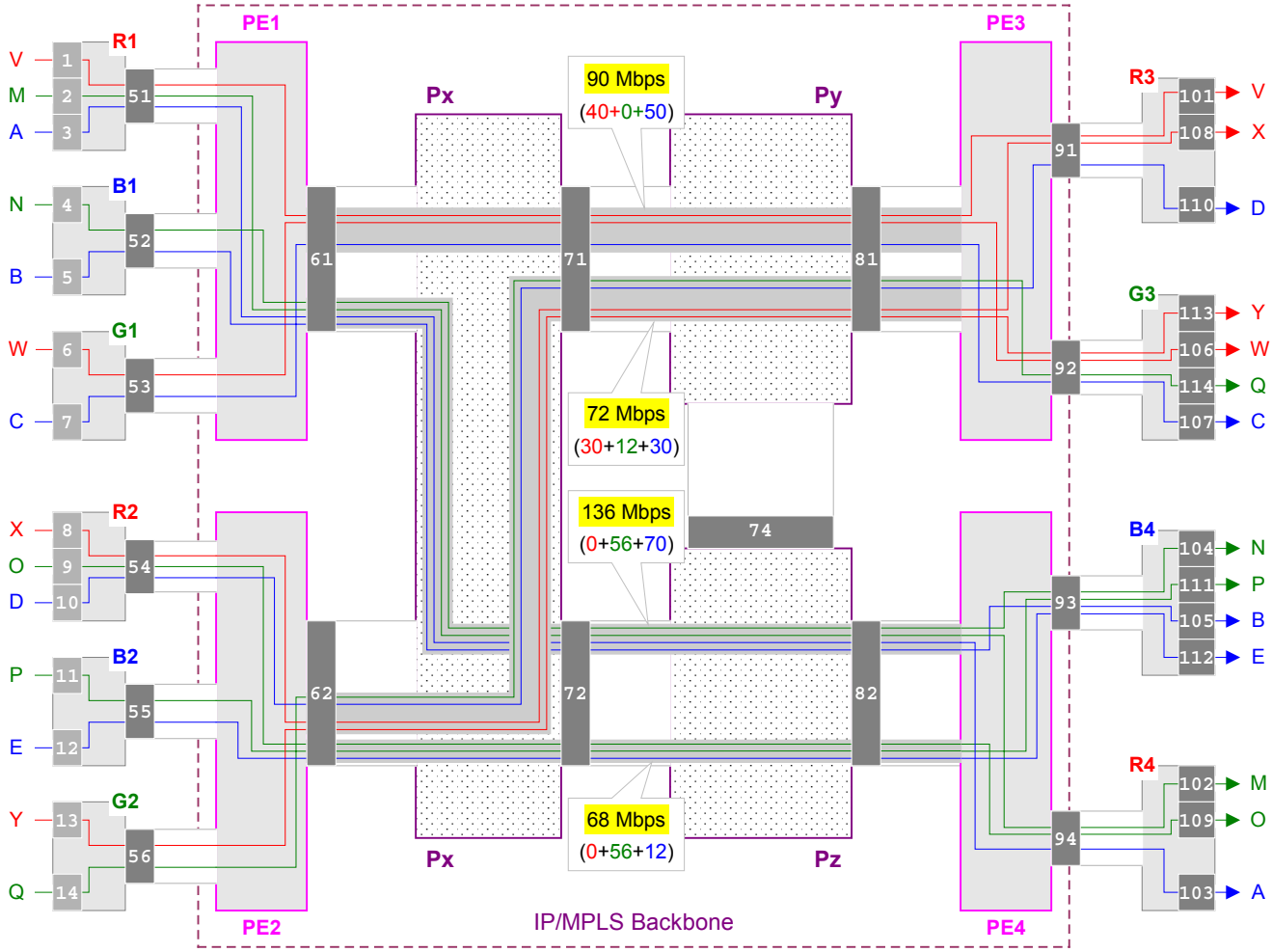


Figure 12: VPN Case Study – Initial Run Configuration

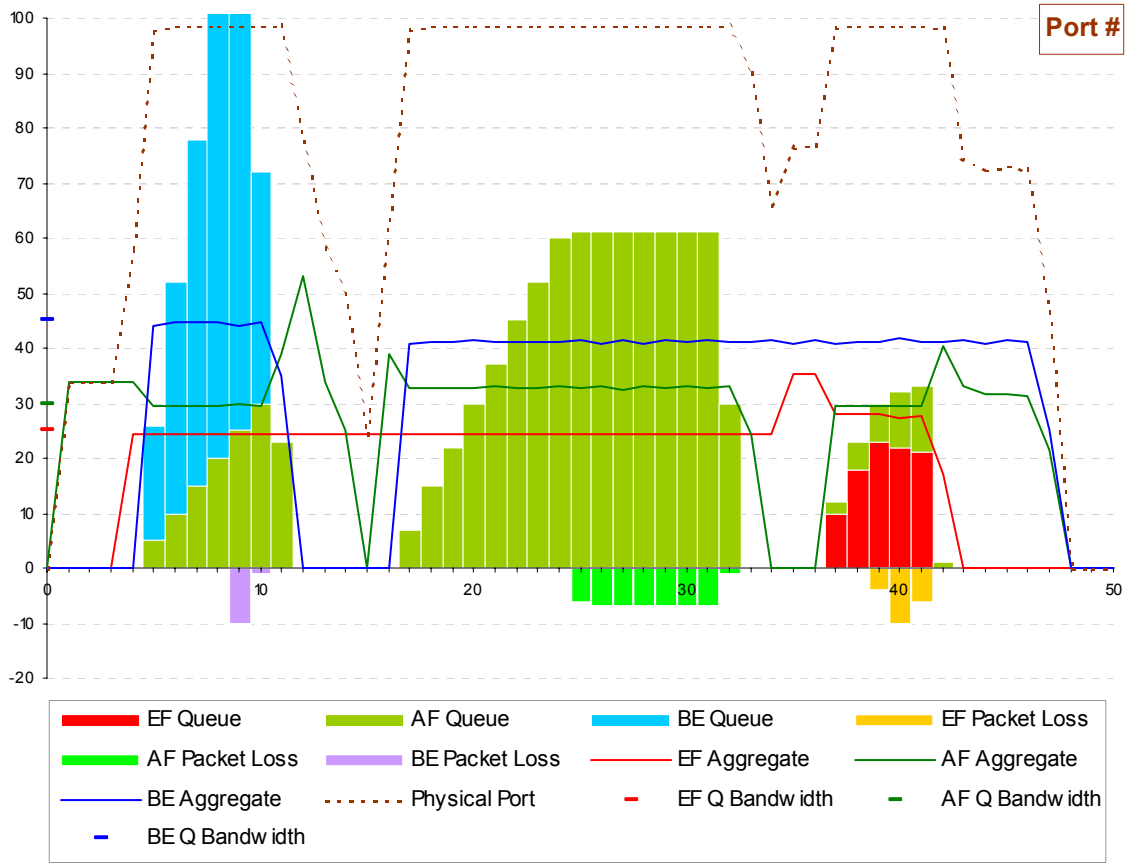
With the subsequent runs, we will have the same offered VPN traffic but we will introduce some changes only at port 71 and port 82. These changes are extra flows, link failure or variation of some configuration parameters.

4.1 Traffic Analysis – Reader's Guide

Chart 1 illustrates a trial with two consecutive flows per class that successively created strong congestion in each queue; as a result this graph covers all possible cases. This representation will be used for analyzing the traffic in the subsequent scenarios of the case study. The information elements are as follows:

- Colors are consistently applied for each class of traffic: red for EF, green for AF and blue for BE.
- The vertical axis is scaled (in Mbps) at the port capacity, and possibly less whenever the port is rate-limited. Besides, the bandwidth value assigned to each queue is shown by a fixed cursor.
- The horizontal axis for elapsed time is scaled at 10ms per unit, based on the generated traffic volume with this case study: every 10ms, IPVCoSS collects measurements related to average throughput and queue occupancy.
- Line charts represent traffic throughput for each aggregate (EF, AF, BE) as well as, in dotted line, the global throughput. It should be noticed that these throughputs, conversely to IP flow throughputs, integrate the frame overhead and therefore are slightly higher that the sum of the flow throughputs.
- Column charts represent queue occupancy with one unit for 1000 bytes. They are stacked from bottom to top respectively for EF, AF and BE queues, if any. The vertical axis is primarily scaled for bandwidth and not for queue volume and whenever queues are very busy the stack can be truncated at the top of the graph. Supplementary information will be found anyway in the summary report that follows each graph.
- Column charts below the horizontal axis show packet losses when queue capacity overflows, if any.

Chart 1: Example for reader's guide



Each chart is followed by a summary report that: (1) recalls the characteristics of the port; (2) for each queue, recalls bandwidth and depth, and provides occupancy ratio as well as the number of dropped packets if any; (3) for each flow crossing this port, provides the information about jitter experienced at this port as well as, if any, packet loss and retransmitted packets because of fast retransmit or time-out condition.

```
Port  Node  Rate  Limit  Adjnode  Distance  Propagation-Delay
----  ----  ----  -----  -------  --------  -----------------

  Queue  Bandwidth       Depth                Used: Max   Mean  Dropped
  -----  -------------  ------------------    ----------------  -------

    Flow  Class   hopJmax  hopJmean  cumJmax  cumJmean   Loss  Retransmitted
                                                              FastRxmit  TO
    ----  -----   -------  --------  -------  --------   ----  --------- ---
```

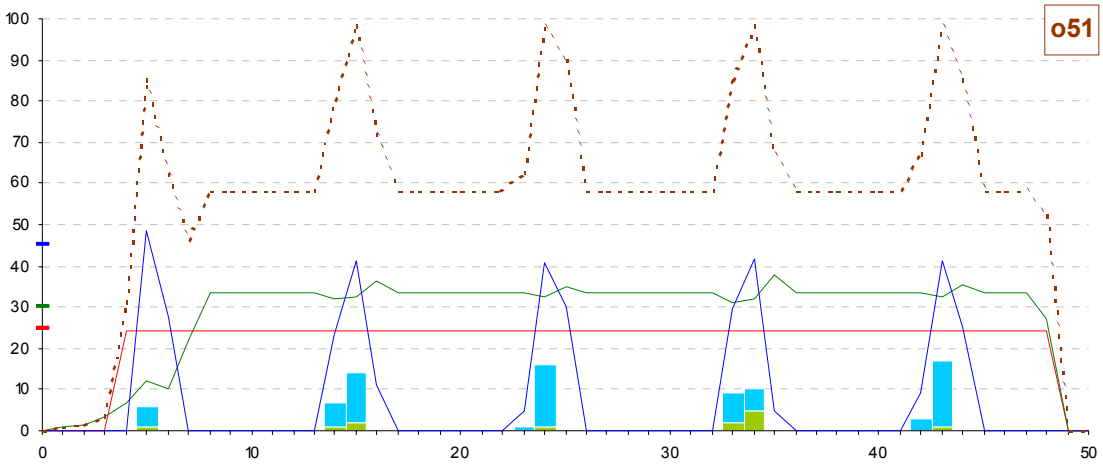
4.2 Traffic Analysis – Ingress Access Links

Chart 2: Case Study Initial Run – Port 51: R1-to-PE1 ingress access link

EF traffic throughput (24.4) is slightly below EF bandwidth.

AF traffic throughput (33.7) is above AF bandwidth.

When BE traffic uses its bandwidth and port capacity is reached, we can see AF throughput slow down, but when BE burst terminates, it goes back above its level because the traffic in queue is released.



One may observe that the AF flow, in its initial phase, is not ascending continuously. However, this is a normal TCP slow start and this is not at all related to the first BE burst. This is due to the longest possible distance over this network between the origin and destination for this "M" flow. A larger scanning period would have rendered a gradually ascending curb.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o51	R1	100		PE1	6 km	33.5 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	25%	25.0 Mbps	3ms	37500 Bytes	4.07%	2.03%	0
AF	30%	30.0 Mbps	10ms	125000 Bytes	4.63%	0.41%	0
BE	45%	45.0 Mbps	15ms	187500 Bytes	11.79%	4.74%	0

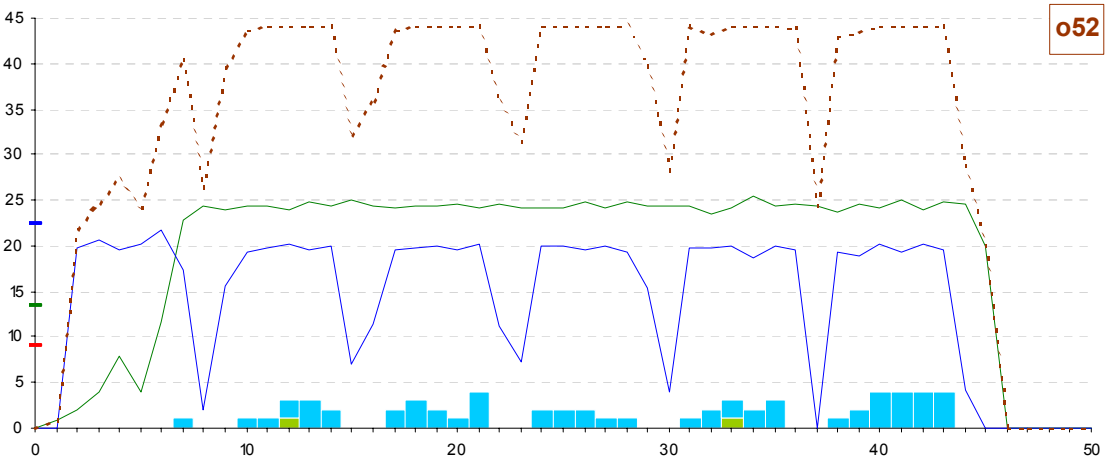
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
V	EF	65.7	4.7	65.7	4.7	0		
M	AF	1299.6	89.5	1299.6	89.5	0		
A	BE	4098.6	1672.6	4098.6	1672.6	0		

Chart 3: Case Study Initial Run – Port 52: B1-to-PE1 ingress access link

On this DS3 port, AF traffic throughput (around 24 Mbps) is well above AF bandwidth.

However, it can borrow from EF bandwidth since there is no EF flow.

It is therefore little disturbed when there are BE traffic bursts that create some port congestion.



Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o52	B1	45		PE1	18 km	100.3 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	20%	9.0 Mbps	3ms	16578 Bytes	0.00%	0.00%	0
AF	30%	13.5 Mbps	10ms	55262 Bytes	4.41%	1.07%	0
BE	50%	22.5 Mbps	15ms	82893 Bytes	8.39%	2.31%	0

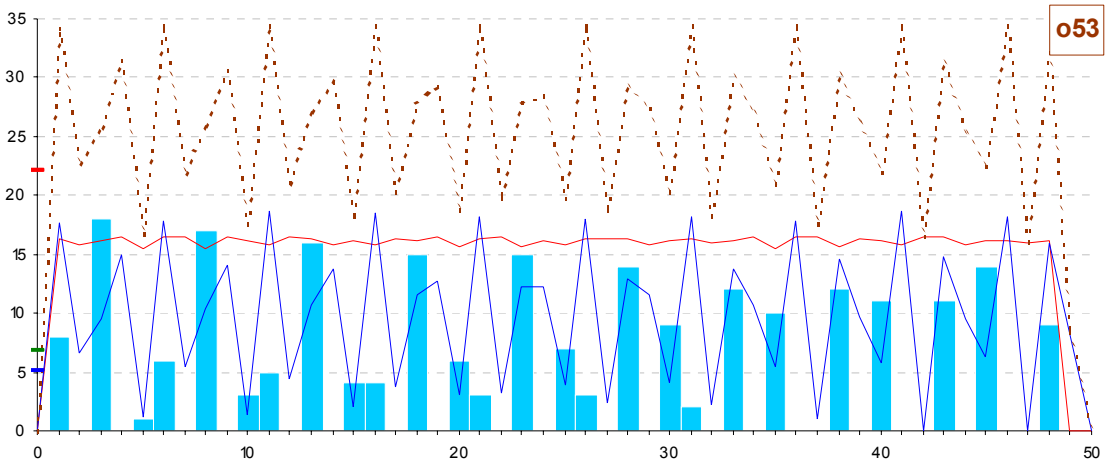
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
N	AF	759.9	171.8	759.9	171.8	0		
B	BE	2441.1	751.3	2441.1	751.3	0		

Chart 4: Case Study Initial Run – Port 53: G1-to-PE1 ingress access link

On this E3 port, the EF traffic throughput is well below EF bandwidth.

Since there is no AF flow, BE traffic can use the remainder of the port capacity.

We have generated many short consecutive BE traffic bursts, and therefore short congestions.



One can observe that the jitter experienced by the EF flow is higher than on port 51.

The reason is that we have traffic entering the router at a rate much higher (FE ingress ports) than the rate of this E3 port.

For the same packet size, frames in output have a much longer serialization time than in input, thus input packets will wait more in queue.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o53	G1	34		PE1	22 km	122.6 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	65%	22.1 Mbps	3ms	12888 Bytes	9.38%	4.69%	0
AF	20%	6.8 Mbps	10ms	42960 Bytes	0.00%	0.00%	0
BE	15%	5.1 Mbps	20ms	85920 Bytes	23.32%	11.09%	0

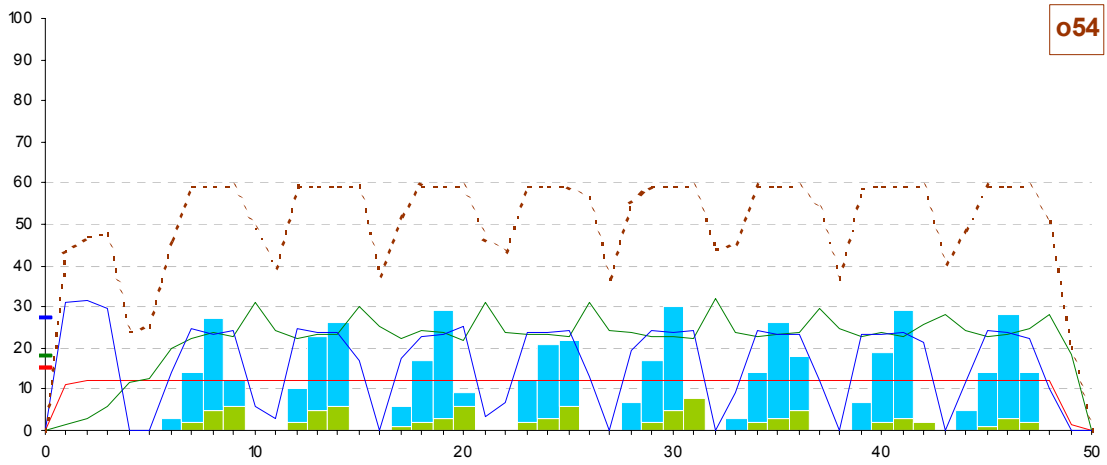
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
W	EF	232.8	46.5	232.8	46.5	0		
C	BE	8559.7	4055.0	8559.7	4055.0	0		

Chart 5: Case Study Initial Run – Port 54: R2-to-PE2 ingress access link

Here we have an FE port rate-limited to 60 Mbps.

The traffic profile at this port is similar to port 51, but more accentuated.

AF throughput (25 Mbps) is well above AF bandwidth. It can borrow a little (3 Mbps) from the remainder of EF bandwidth but when BE bursts occur, AF packets have to be queued.



When BE bursts terminate, these packets are immediately sent because the port is available. This entails these peaks at around 30 Mbps.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o54	R2	100	60	PE2	10 km	55.7 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	25%	15.0 Mbps	3ms	37500 Bytes	4.07%	2.03%	0
AF	30%	18.0 Mbps	10ms	125000 Bytes	7.41%	2.32%	0
BE	45%	27.0 Mbps	15ms	187500 Bytes	16.00%	6.63%	0

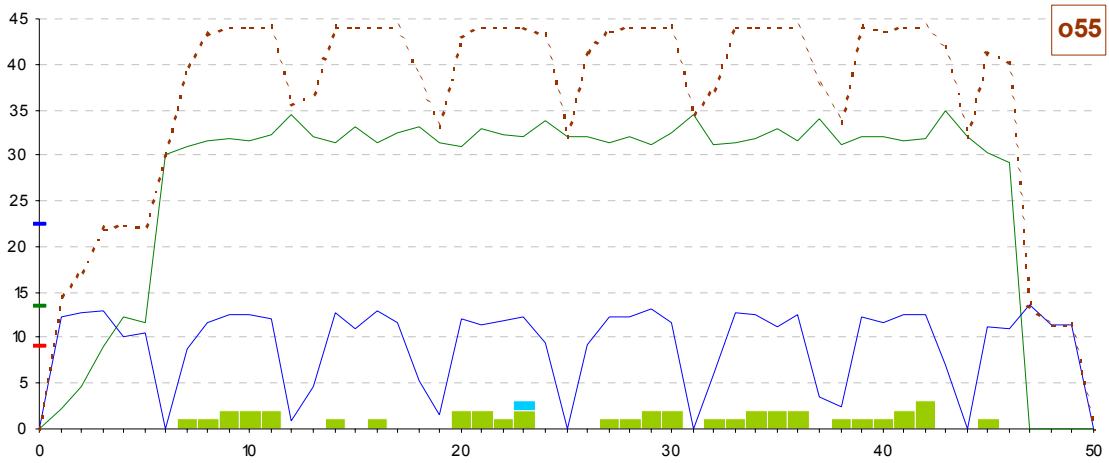
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
X	EF	129.5	52.3	129.5	52.3	0		
O	AF	2973.1	900.9	2973.1	900.9	0		
D	BE	10239.9	4155.3	10239.9	4155.3	0		

Chart 6: Case Study Initial Run – Port 55: B2-to-PE2 ingress access link

On this DS3 access link, there are only 2 flows.

The BE traffic throughput is well under BE bandwidth. When BE bursts at 12-13 Mbps occur they can use the port easily.

When the port is temporarily congested, the AF packets are queued because AF traffic is well above the remaining capacity.



Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o55	B2	45		PE2	26 km	144.8 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	20%	9.0 Mbps	3ms	16578 Bytes	0.00%	0.00%	0
AF	30%	13.5 Mbps	10ms	55262 Bytes	7.65%	2.47%	0
BE	50%	22.5 Mbps	15ms	82893 Bytes	4.61%	0.66%	0

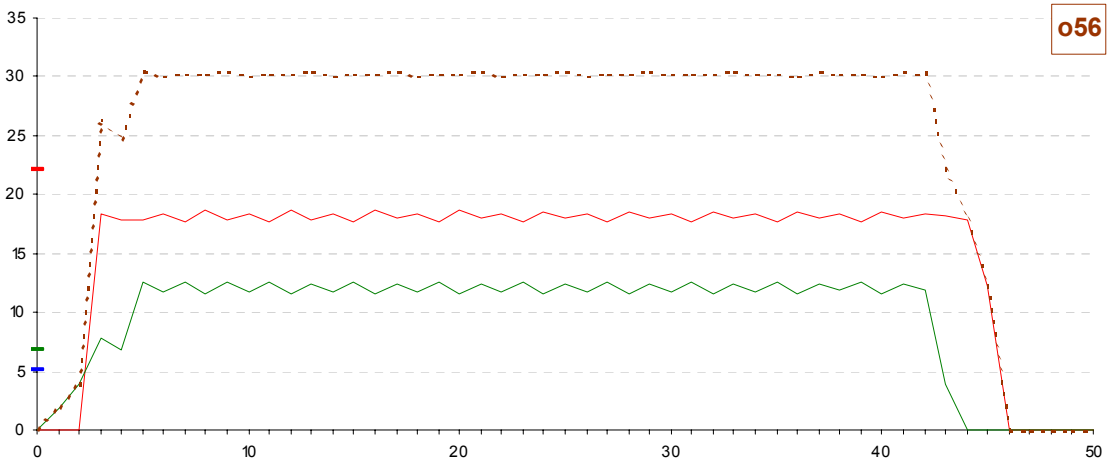
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	TO
P	AF	968.6	289.9	968.6	289.9	0		
E	BE	1621.9	217.5	1621.9	217.5	0		

Chart 7: Case Study Initial Run – Port 56: G2-to-PE2 ingress access link

On this E3 port, there is no congestion but the port is busy at nearly 90%.

However one can observe that there is a little jitter.

This happens for the same reasons as with port 53: the FE input ports are faster than this output E3 port., and the 1200-byte packets are still being serialized when input packets are arrived.



Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o56	G2	34		PE2	14 km	78.0 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	65%	22.1 Mbps	3ms	12888 Bytes	9.38%	4.69%	0
AF	20%	6.8 Mbps	10ms	42960 Bytes	2.82%	1.41%	0
BE	15%	5.1 Mbps	20ms	85920 Bytes	0.00%	0.00%	0

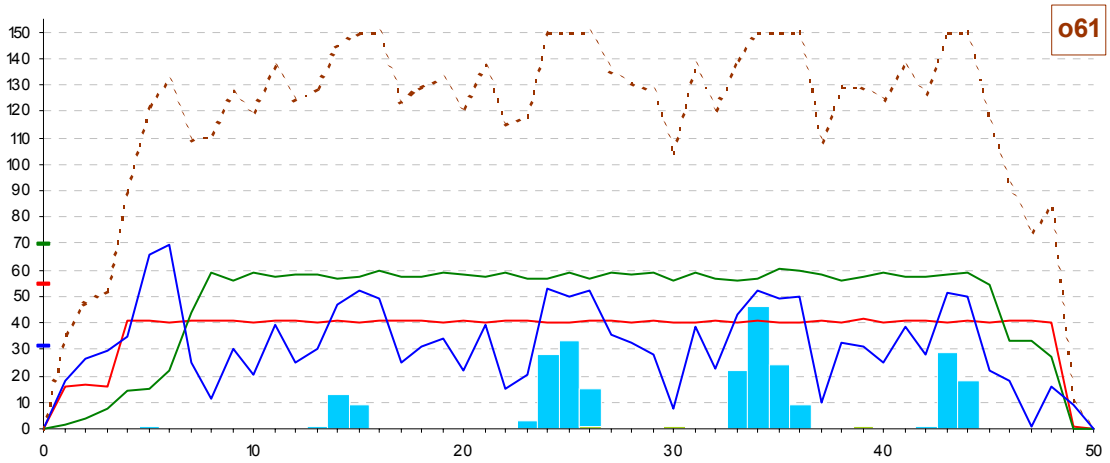
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	TO
Y	EF	221.3	69.3	221.3	69.3	0		
Q	AF	231.9	50.1	231.9	50.1	0		

4.3 Traffic Analysis – Backbone Links

Chart 8: Case Study Initial Run – Port 61: PE1-to-Px backbone link

At this STM-1 backbone link that receives traffic flows from 3 CEs (those at the top left of our reference picture) the situation is sane with respect to the offered traffic: EF and AF traffic aggregates are below their assigned bandwidth.

There is only some little congestion with BE bursty traffic.



There is little jitter added to EF flows because faster than the upstream access links.

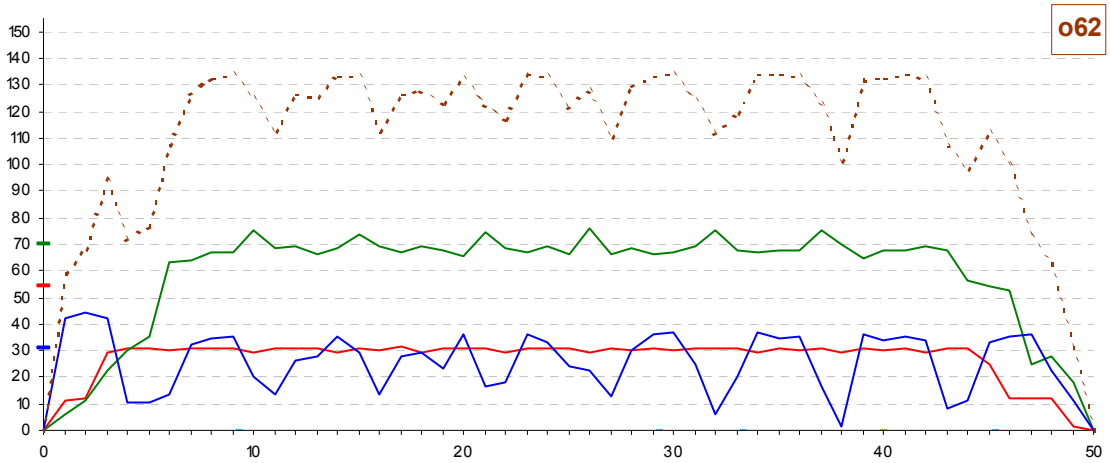
Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o61	PE1	155		Px	270 km	1503.5 microsec

Queue	Bandwidth	Depth	Used: Max	Mean	Dropped
EF	35% 54.3 Mbps	3ms 56160 Bytes	4.87%	1.38%	0
AF	45% 69.8 Mbps	10ms 187200 Bytes	1.49%	0.34%	0
BE	20% 31.0 Mbps	15ms 280800 Bytes	17.84%	3.56%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
V	EF	133.1	19.1	134.1	23.9	0	
W	EF	115.3	13.0	333.8	59.5	0	
M	AF	292.1	66.7	1357.3	156.2	0	
N	AF	278.7	45.4	823.9	217.2	0	
A	BE	8555.8	2811.9	12045.9	4484.6	0	
B	BE	8188.8	1099.5	9296.3	1850.9	0	
C	BE	7496.4	928.3	15749.1	4983.4	0	

Chart 9: Case Study Initial Run – Port 62: PE2-to-Px backbone link

At this STM-1 backbone link that receives traffic flows from the 3 other CEs (those at the bottom left of our reference picture) there is no congestion.



We can see that nearly no jitter has been added to the flows.

Jitter average values of respectively EF, AF and BE flows reflect simply the priority at the scheduler level. Even if there is bandwidth, contingency in packet arrivals entail very transient queuing but the scheduler selects EF packets first then AF ones and BE ones.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o62	PE2	155		Px	220 km	1225.1 microsec

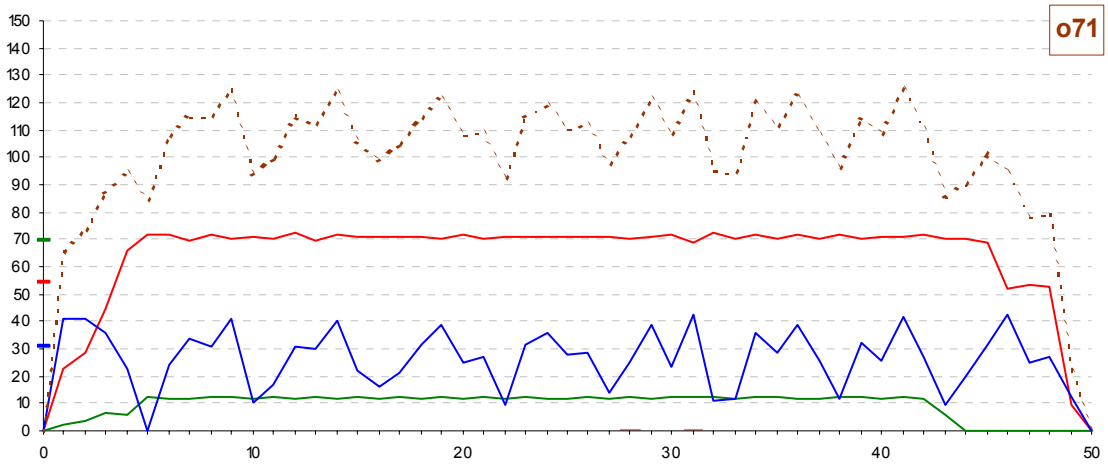
Queue	Bandwidth	Depth	Used: Max	Mean	Dropped
EF	35% 54.3 Mbps	3ms 56160 Bytes	4.87%	1.32%	0
AF	45% 69.8 Mbps	10ms 187200 Bytes	1.90%	0.52%	0
BE	20% 31.0 Mbps	15ms 280800 Bytes	1.17%	0.27%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
X	EF	107.1	16.8	213.2	69.2	0	
Y	EF	81.1	20.2	295.6	89.5	0	
O	AF	340.5	77.6	3041.7	978.5	0	
P	AF	195.5	30.9	1033.7	320.8	0	
Q	AF	233.4	59.0	328.1	109.2	0	
D	BE	367.0	106.5	10239.9	4261.9	0	
E	BE	499.7	93.6	1684.8	311.2	0	

Chart 10: Case Study Initial Run – Port 71: Px-to-Py backbone link

There is no congestion at this port considering the offered traffic. However there is a threat for the EF flows because the EF aggregate throughput is above the EF bandwidth assigned at this port.

Run#2 of our case study will illustrate the consequences of this inadequate configuration.



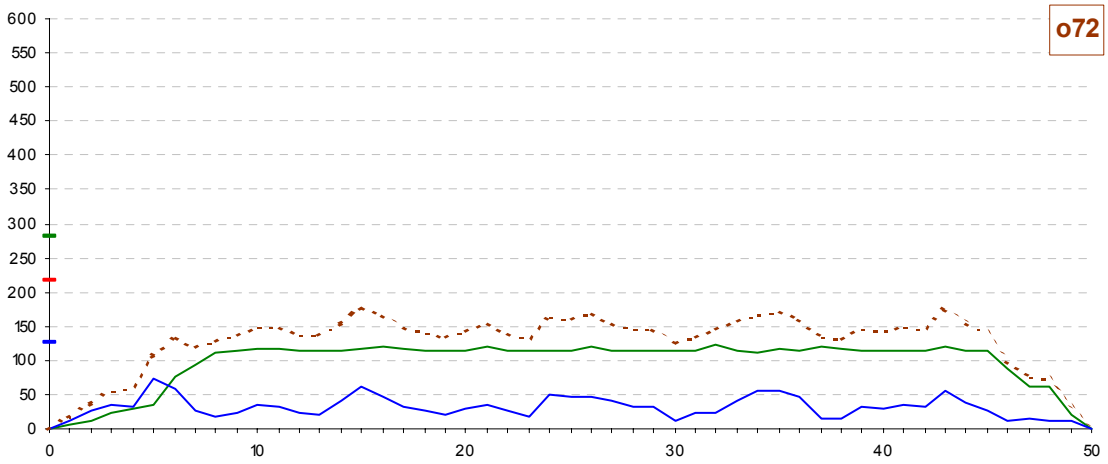
Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o71	Px	155		Py	436 km	2427.9 microsec

Queue	Bandwidth	Depth	Used: Max	Mean	Dropped
EF	35% 54.3 Mbps	3ms 56160 Bytes	5.41%	1.27%	0
AF	45% 69.8 Mbps	10ms 187200 Bytes	0.65%	0.32%	0
BE	20% 31.0 Mbps	15ms 280800 Bytes	1.40%	0.19%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
V	EF	115.4	16.9	177.0	40.8	0	
W	EF	136.7	17.6	349.8	77.2	0	
X	EF	77.4	7.3	268.6	76.5	0	
Y	EF	132.4	19.7	392.7	109.2	0	
Q	AF	142.0	18.9	352.0	128.1	0	
C	BE	534.1	42.5	15789.7	5025.9	0	
D	BE	475.5	52.8	10340.3	4314.7	0	

Chart 11: Case Study Initial Run – Port 72: Px-to-Pz backbone link

This port is an STM-4 port, oversized for the current traffic.



It should be noticed that, because of low port occupancy (30%) and the very short serialization time of packets due to the high rate of this port, there is very little jitter for each flow.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o72	Px	622		Pz	285 km	1587.1 microsec

Queue	Bandwidth	Depth	Used: Max	Mean	Dropped
EF	35% 217.7 Mbps	3ms 224640 Bytes	0.00%	0.00%	0
AF	45% 279.9 Mbps	10ms 748800 Bytes	0.21%	0.05%	0
BE	20% 124.4 Mbps	15ms 1123200 Bytes	0.18%	0.03%	0

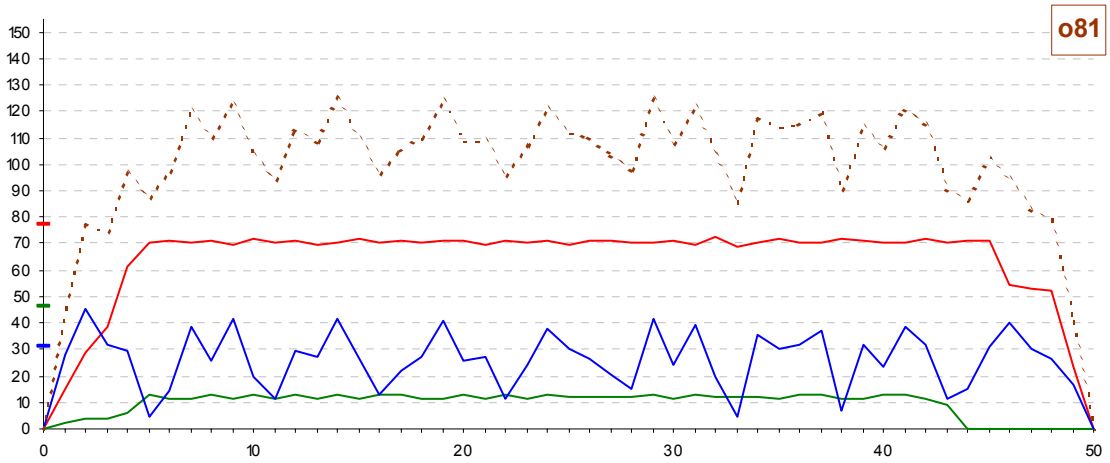
Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
M	AF	18.8	0.8	1357.3	157.0	0	
N	AF	18.9	0.8	823.9	218.1	0	
O	AF	18.1	0.5	3041.7	979.1	0	
P	AF	14.5	0.5	1033.7	321.3	0	
A	BE	23.9	1.1	12045.9	4485.8	0	
B	BE	31.7	1.1	9313.0	1852.0	0	
E	BE	23.3	1.1	1686.0	312.3	0	

Chart 12: Case Study Initial Run – Port 81: Py-to-PE3 backbone link

With this initial scenario, all the flows at this port arrive from the same upstream port (no. 71).

However, conversely to port 7, aggregated EF throughput is below EF bandwidth.

This port is well tuned with respect to offered traffic.



Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o81	Py	155		PE3	250 km	1392.2 microsec

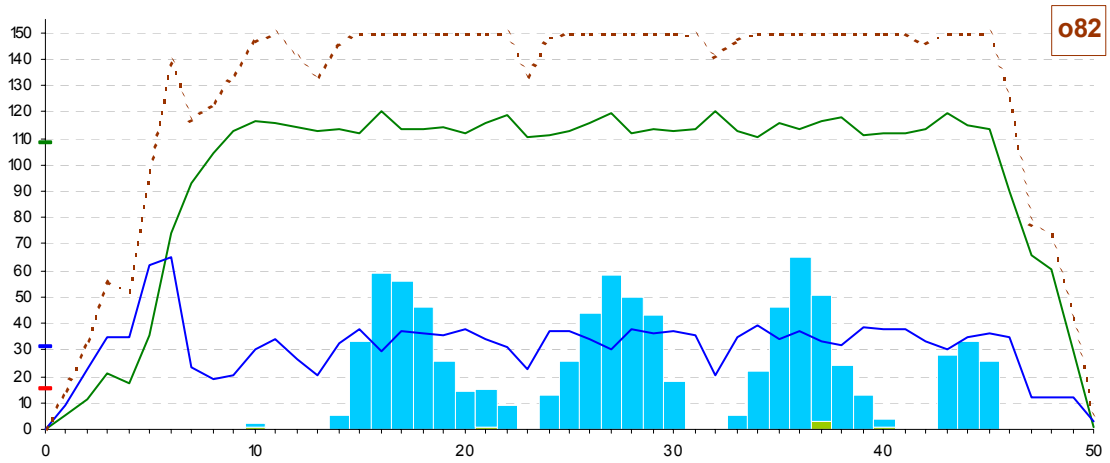
Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	50%	77.5 Mbps	3ms	56160 Bytes	2.70%	1.20%	0
AF	30%	46.5 Mbps	10ms	187200 Bytes	0.65%	0.32%	0
BE	20%	31.0 Mbps	15ms	280800 Bytes	0.53%	0.17%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
V	EF	0.0	0.0	177.0	40.8	0	
W	EF	15.8	4.9	365.6	82.2	0	
X	EF	0.0	0.0	268.6	76.5	0	
Y	EF	15.8	6.7	408.5	116.0	0	
Q	AF	15.8	6.1	367.8	134.3	0	
C	BE	202.6	35.1	15816.2	5061.1	0	
D	BE	198.7	36.6	10401.1	4351.3	0	

Chart 13: Case Study Initial Run – Port 82: Pz-to-PE4 backbone link

There is heavy congestion at this port but it impacts only BE traffic bursts.

Actually, AF throughput is above AF bandwidth but it can borrow what is missing from EF bandwidth since there is no EF traffic.



Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o82	Pz	155		PE4	425 km	2366.7 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	10%	15.5 Mbps	3ms	56160 Bytes	0.00%	0.00%	0
AF	70%	108.5 Mbps	10ms	187200 Bytes	3.97%	0.61%	0
BE	20%	31.0 Mbps	15ms	280800 Bytes	24.35%	7.25%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
M	AF	422.9	79.5	1494.7	236.6	0	
N	AF	416.5	71.8	1000.7	289.9	0	
O	AF	386.9	71.5	3116.3	1050.7	0	
P	AF	353.7	39.5	1082.7	360.9	0	
A	BE	13706.5	4955.0	24961.7	9440.8	0	
B	BE	16949.4	4687.1	22119.1	6539.1	0	
E	BE	17170.2	3829.8	17308.1	4142.2	0	

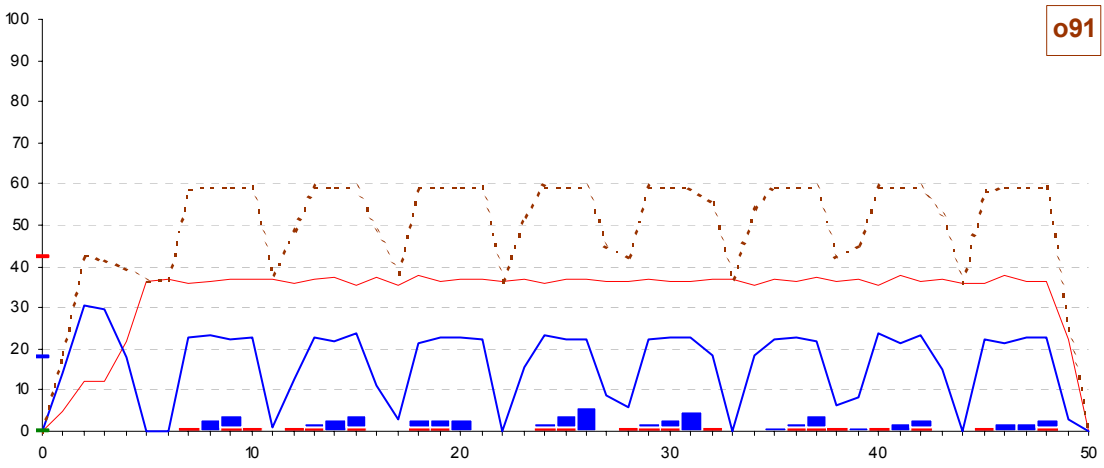
4.4 Traffic Analysis – Egress Access Links

Chart 14: Case Study Initial Run – Port 91: PE3-to-R3 egress access link

This Ethernet port is rate-limited to 60 Mbps.

The BE traffic bursts create some little congestion.

EF traffic is under EF bandwidth but there is some jitter. This is explained once again by the fact that the upstream port is at a higher rate.



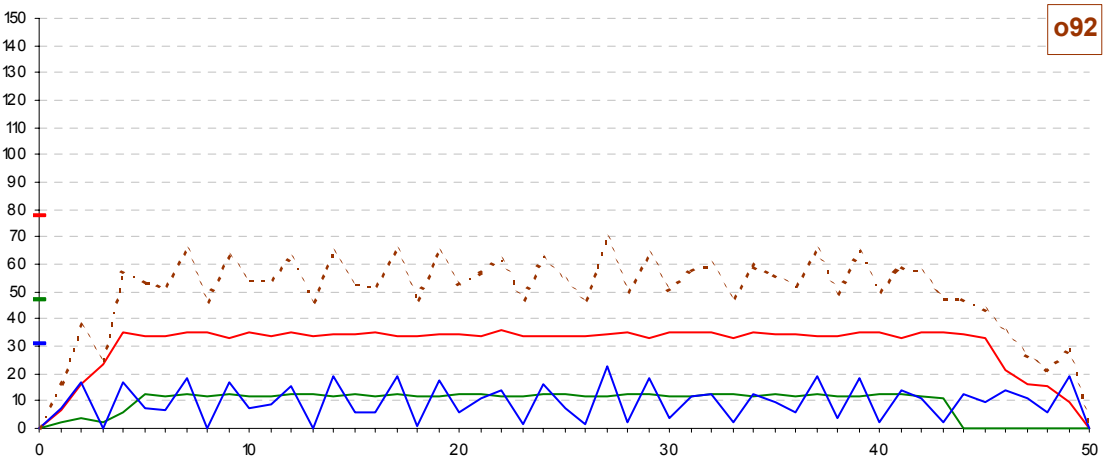
Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o91	PE3	100	60	R3	12 km	66.9 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	70%	42.0 Mbps	3ms	37500 Bytes	8.14%	2.09%	0
BE	30%	18.0 Mbps	15ms	187500 Bytes	5.00%	1.73%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss
V	EF	249.4	62.0	298.0	102.8	0
X	EF	202.9	37.0	471.5	113.6	0
D	BE	2983.8	1067.1	12158.4	5418.4	0

Chart 15: Case Study Initial Run – Port 92: PE3-to-G3 egress access link

This STM-1 port is oversized and only used at around 40%.



There is absolutely no jitter at this port, except a little for the BE flow simply because its packets are of variable size.

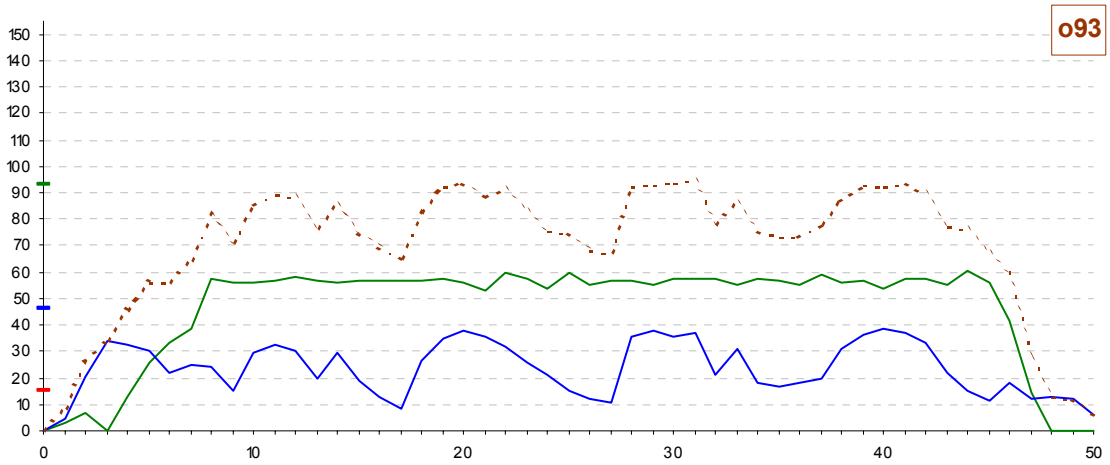
Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o92	PE3	155		G3	21 km	117.0 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	50%	77.5 Mbps	3ms	56160 Bytes	2.15%	1.08%	0
AF	30%	46.5 Mbps	10ms	187200 Bytes	0.65%	0.32%	0
BE	20%	31.0 Mbps	15ms	280800 Bytes	0.42%	0.11%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
W	EF	0.0	0.0	365.6	82.2	0		
Y	EF	0.0	0.0	408.5	116.0	0		
Q	AF	0.0	0.0	367.8	134.3	0		
C	BE	61.4	15.8	15816.2	5077.0	0		

Chart 16: Case Study Initial Run – Port 93: PE4-to-B4 egress access link

This STM-1 port is used at around 60%.



There is very little jitter, compared to port 92. This is explained by the higher percentage of occupancy.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o93	PE4	155		B4	42 km	233.9 microsec

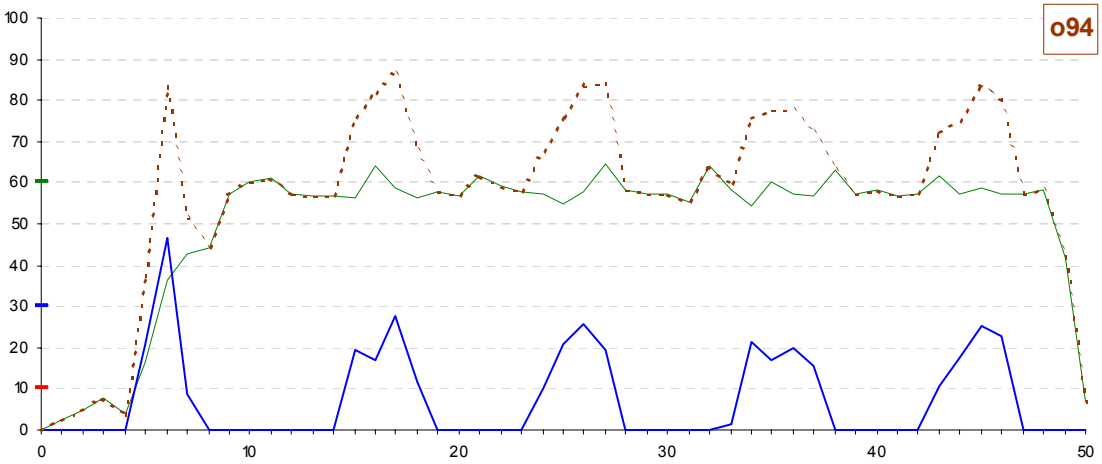
Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	10%	15.5 Mbps	3ms	56160 Bytes	0.00%	0.00%	0
AF	60%	93.0 Mbps	10ms	187200 Bytes	0.75%	0.24%	0
BE	30%	46.5 Mbps	15ms	280800 Bytes	0.54%	0.14%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
N	AF	47.9	10.3	1018.7	300.2	0		
P	AF	5.2	0.0	1082.7	360.9	0		
B	BE	104.7	11.6	22153.8	6550.7	0		
E	BE	98.9	11.4	17308.1	4153.6	0		

Chart 17: Case Study Initial Run – Port 94: PE4-to-R4 egress access link

This FE port is used at 60% with AF traffic.

There are some bursts at 80% due to the BE traffic.



There is more jitter than with ports 92 and 93 but less than with port 91, because although the upstream port is at a higher rate (STM-1) there is heavy load (the peaks) but no congestion.

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o94	PE4	100		R4	9 km	50.2 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	10%	10.0 Mbps	3ms	37500 Bytes	0.00%	0.00%	0
AF	60%	60.0 Mbps	10ms	125000 Bytes	2.00%	0.33%	0
BE	30%	30.0 Mbps	15ms	187500 Bytes	2.44%	0.29%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
M	AF	241.0	29.5	1542.3	266.2	0		
O	AF	109.9	9.0	3116.3	1059.7	0		
A	BE	870.4	70.9	24971.6	9511.8	0		

4.5 Traffic Analysis – EF Flows

In the following final reports related to EF flows, we can see for each flow and at each output port along the path, the value of the QoS parameters: throughput, delay, jitter and packet loss. The stringent QoS requirements associated to EF class of service have been respected:

- At the ingress IPORT, the flow has been generated at the required throughput and regularly maintained at each internal OPORT up to the egress OPORT.
- Jitter is acceptable: at the egress OPORT, the max jitter value is under 500 microseconds and the mean jitter around 100 microseconds. We can notice that where there is the more jitter is at OPORTs with an upstream port at a higher rate and if there was some congestion: e.g. from FE to E3 (100 to 34 Mbps).
- The delay is mainly dependant on the path length and was not impacted by queuing delay since there was very little jitter.

Flow ID	Trsp	Thrpt Mbps	Type	Class	maxPKsz bytes	T1 ms	Volume bytes	Occ	Gap ms	Flow total elapsed time
V	UDP	24	CBR	EF	1500	30	1350000	1	0	482735.5

Port	Rate	Thrput	Dmin	Dmax	Jmin	Jmax	Jmean	Nbpk	Loss
i1	100	24.00							
o51	100	24.00	264.2	329.9	0.0	65.7	4.7	900	0
o61	155	24.00	388.8	555.6	0.0	134.1	23.9	900	0
o71	155	24.00	1983.4	2233.5	0.0	177.0	40.8	900	0
o81	155	24.00	4502.2	4752.3	0.0	177.0	40.8	900	0
o91	100	24.00	6110.5	6450.4	0.0	298.0	102.8	900	0
o101	100	24.00	6252.6	6592.5	0.0	298.0	102.8	900	0

Flow ID	Trsp	Thrpt Mbps	Type	Class	maxPKsz bytes	T1 ms	Volume bytes	Occ	Gap ms	Flow total elapsed time
W	UDP	16	CBR	EF	1200	0	960000	1	0	485821.4

Port	Rate	Thrput	Dmin	Dmax	Jmin	Jmax	Jmean	Nbpk	Loss
i6	100	16.00							
o53	34	16.00	399.6	632.4	0.0	232.8	46.5	800	0
o61	155	16.00	597.3	931.1	0.0	333.8	59.5	800	0
o71	155	16.00	2175.9	2525.7	0.0	349.8	77.2	800	0
o81	155	16.00	4678.6	5044.2	0.0	365.6	82.2	800	0
o92	155	16.00	6145.4	6511.0	0.0	365.6	82.2	800	0
o106	100	16.00	6380.5	6746.1	0.0	365.6	82.2	800	0

Flow ID	Trsp	Thrpt Mbps	Type	Class	maxPKsz bytes	T1 ms	Volume bytes	Occ	Gap ms	Flow total elapsed time
X	UDP	12	CBR	EF	1500	1	720000	1	0	485903.7

Port	Rate	Thrput	Dmin	Dmax	Jmin	Jmax	Jmean	Nbpk	Loss
i8	100	12.00							
o54	100	12.00	344.2	473.7	0.0	129.5	52.3	480	0
o62	155	12.00	435.4	694.5	0.0	213.2	69.2	480	0
o71	155	12.00	1751.6	2081.0	0.0	268.6	76.5	480	0
o81	155	12.00	4270.4	4599.8	0.0	268.6	76.5	480	0
o91	100	12.00	5874.7	6411.0	0.0	471.5	113.6	480	0
o108	100	12.00	6016.8	6553.1	0.0	471.5	113.6	480	0

Flow ID	Trsp	Thrpt Mbps	Type	Class	maxPKsz bytes	T1 ms	Volume bytes	Occ	Gap ms	Flow total elapsed time
Y	UDP	18	CBR	EF	1200	20	960000	1	0	450358.5

Port	Rate	Thrput	Dmin	Dmax	Jmin	Jmax	Jmean	Nbpk	Loss
i13	100	18.00							
o56	34	18.00	399.6	620.9	0.0	221.3	69.3	800	0
o62	155	18.00	552.7	887.8	0.0	295.6	89.5	800	0
o71	155	17.99	1852.9	2283.1	0.0	392.7	109.2	800	0
o81	155	17.99	4355.6	4801.6	0.0	408.5	116.0	800	0
o92	155	17.99	5822.4	6268.4	0.0	408.5	116.0	800	0
o113	100	17.99	6057.5	6503.5	0.0	408.5	116.0	800	0

5 VPN Case Study – Run #2 – Disturbance of oversubscribed EF Traffic

Chart 10 on page 22 showed that EF traffic on port 71 was oversubscribed, considering the bandwidth assigned to EF aggregate, but since the port was not congested EF traffic flows were not disturbed. This situation resulted from the fact that traffic engineering was based on aggregated traffic.

Here we run the same VPN flows as previously but we introduce in the middle of the run an extra flow, at 50Mbps, that transits from Px to Py via port 71.

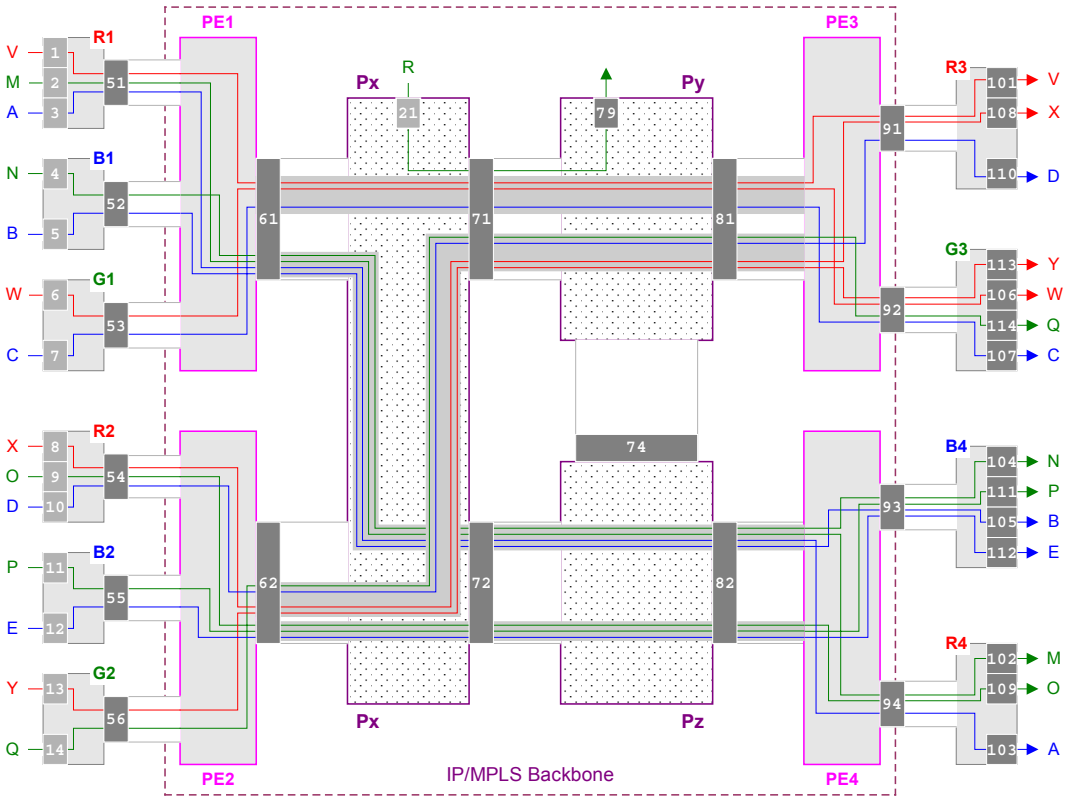


Figure 13: VPN Case Study – Run #2 Configuration

The extra AF flow ("R") consumes most of AF bandwidth and creates congestion at the port.

As a result, EF traffic cannot borrow any more from other aggregates, and especially AF aggregate. All EF flows experiment packet queuing and therefore jitter and even packet loss.

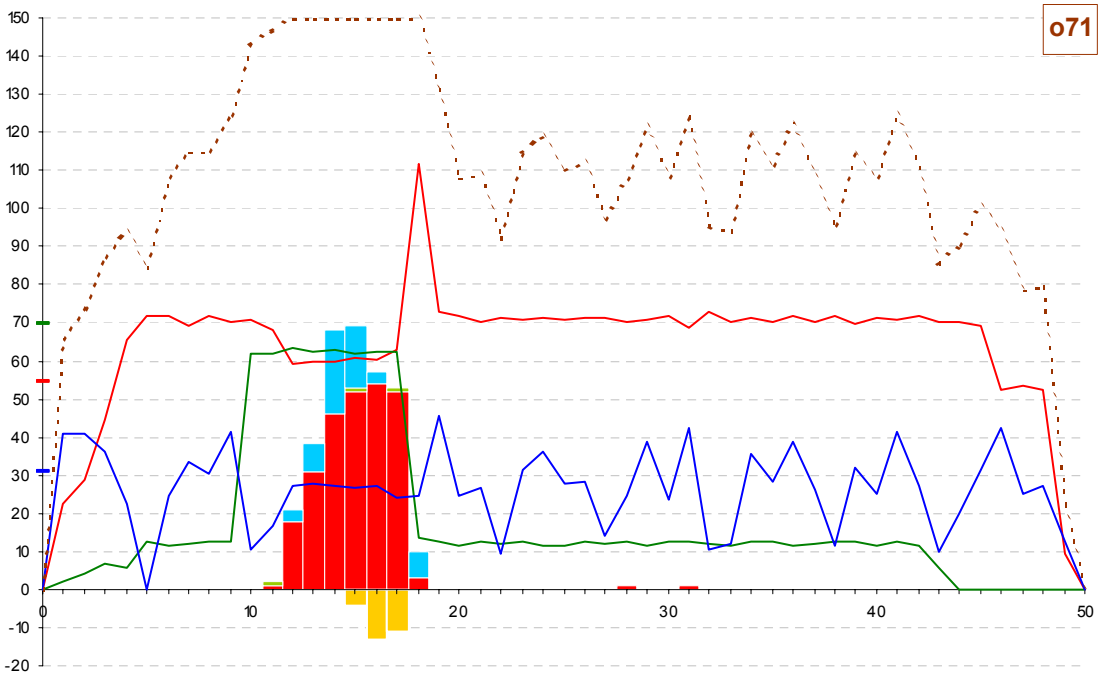


Chart 18: Case Study Run #2 – Port 71

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o71	Px	155		Py	436 km	2427.9 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	35%	54.3 Mbps	3ms	56160 Bytes	102.34%	11.25%	21
AF	45%	69.8 Mbps	10ms	187200 Bytes	2.81%	0.63%	0
BE	20%	31.0 Mbps	15ms	280800 Bytes	9.86%	0.68%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted
V	EF	7503.8	650.1	7552.0	673.8	8	
W	EF	7567.7	639.6	7584.4	699.0	3	
X	EF	7281.4	605.0	7343.4	673.6	4	
Y	EF	7543.2	698.5	7552.0	787.1	6	
Q	AF	595.7	56.4	799.8	165.6	0	
R	AF	630.8	180.6	630.8	180.6	0	
C	BE	7902.1	375.5	17345.6	5358.9	0	
D	BE	8419.2	515.8	16264.0	4777.7	0	

We can notice that AF flows are well under the assigned bandwidth and are not experiencing jitter, although as TCP slows they have no special requirements regarding jitter.

6 VPN Case Study – Run #3 – DiffServ Aware Traffic Engineering

With this run case, we have the same offered traffic that with the previous case but another situation with respect to TE traffic trunks and hence flow paths.

This situation would result from DiffServ aware traffic engineering with two types of traffic trunks: those based on EF traffic class (shown in yellow) and those based on other classes.

Instead of having a single TT between PE2 and PE3, there are two TTs mapped on MPLS LSPs using different paths.

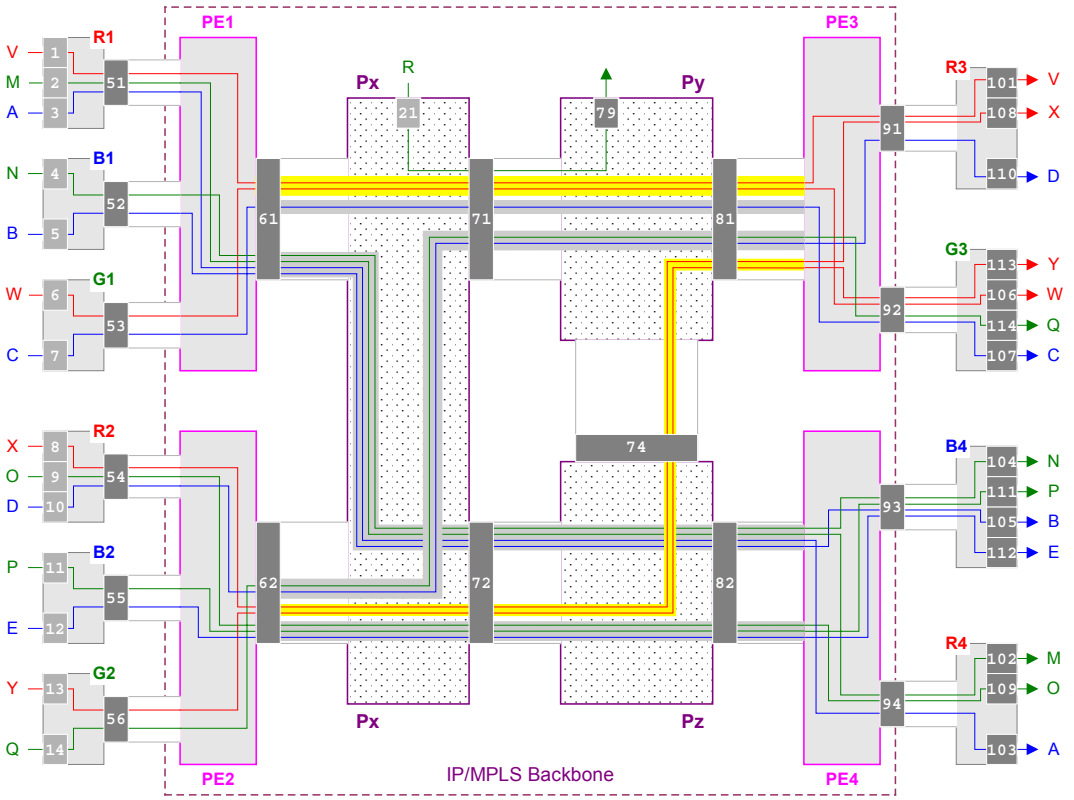


Figure 14: VPN Case Study – Run #3 Configuration

At port 71, the EF traffic (made up of V and W flows only) is now below EF bandwidth and the extra AF flow does not disturb EF flows.

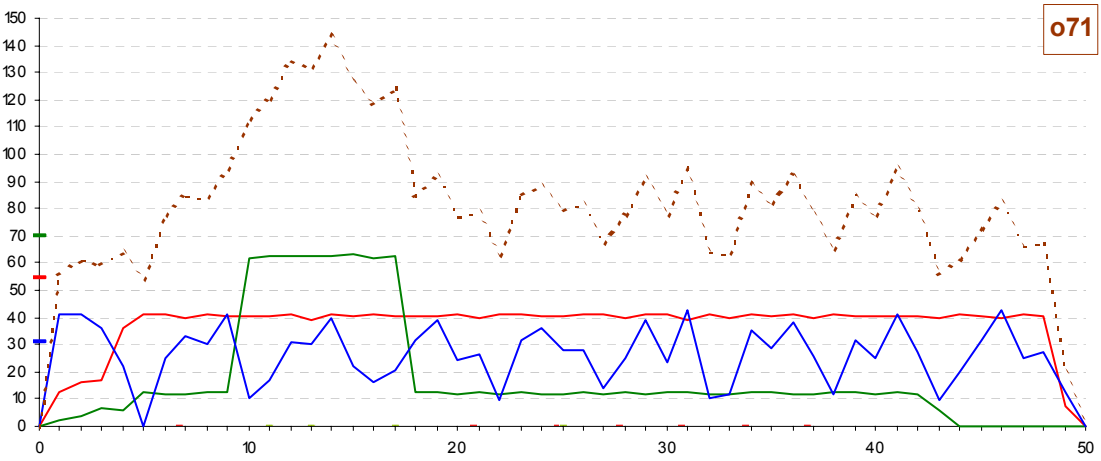


Chart 19: Case Study Run #3 – Port 71

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o71	Px	155		Py	436 km	2427.9 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	35%	54.3 Mbps	3ms	56160 Bytes	2.70%	1.22%	0
AF	45%	69.8 Mbps	10ms	187200 Bytes	1.19%	0.32%	0
BE	20%	31.0 Mbps	15ms	280800 Bytes	1.36%	0.17%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss
V	EF	64.7	7.9	161.3	31.8	0
W	EF	76.0	9.6	349.8	69.1	0
Q	AF	186.0	21.4	359.0	132.5	0
R	AF	184.4	31.1	184.4	31.1	0
C	BE	503.7	27.3	15749.1	5010.7	0
D	BE	567.2	48.5	10340.3	4311.0	0

EF flows X and Y have now a longer path that adds around 1.3 milliseconds to the end-to-end delay, compared to initial run.

There is also slightly more jitter at port 81 because we pass from an STM-4 port to a quite busy STM-1 port.

Flow X

Port	Rate	Thrput	Dmin	Dmax	Jmin	Jmax	Jmean	Nbpk	Loss
o62	155	12.00	435.4	694.5	0.0	213.2	69.3	480	0
o72	622	12.00	1690.8	1949.9	0.0	213.2	69.9	480	0
o74	622	12.00	3308.2	3567.3	0.0	213.2	69.9	480	0
o81	155	12.00	5570.9	6044.9	0.0	454.4	107.8	480	0
o91	100	12.00	7095.2	7580.2	0.0	454.4	123.0	480	0
o108	100	12.00	7304.2	7789.2	0.0	454.4	123.0	480	0

7 VPN Case Study – Run #4 – Link Failure

With the same offered traffic than with the initial run, we are here in a situation that would result from a failure of port 71.

The traffic trunks that crossed port 71 from Px to Py have now been rerouted via STM-4 ports 72 and 74.

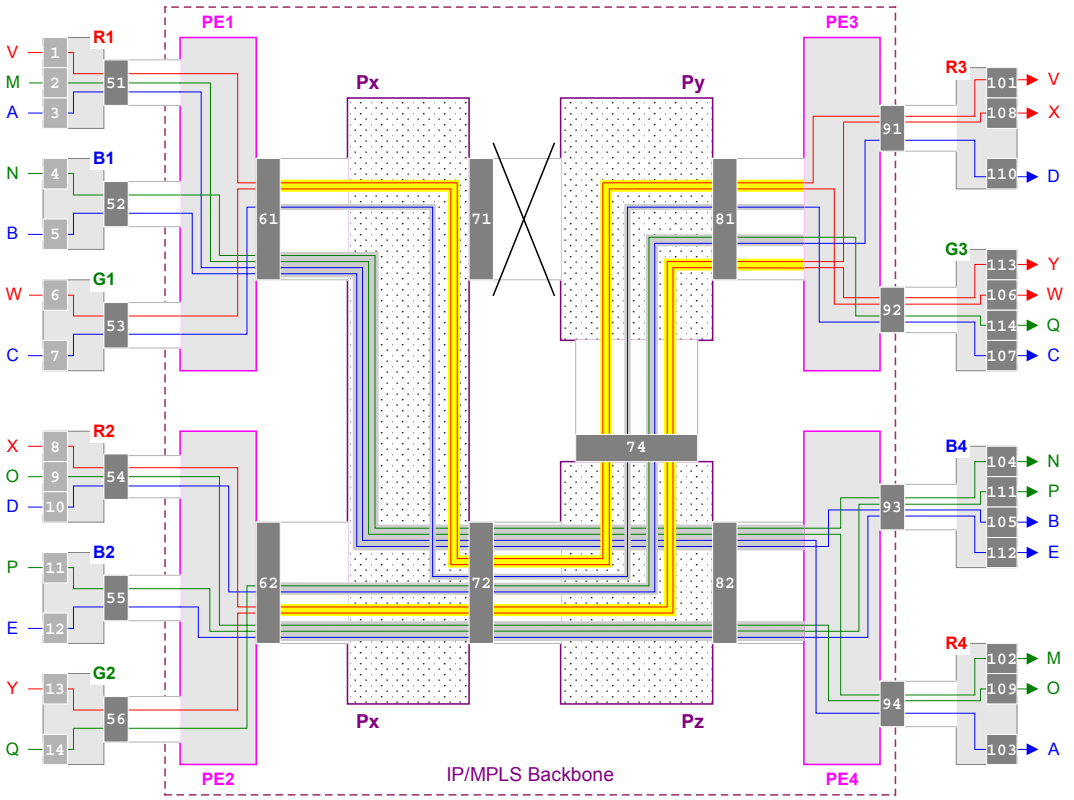


Figure 15: VPN Case Study – Run #4 Configuration

Port 72 that carries all the flows remains oversized.

However the path from Px to Py is longer via Pz and hence adds delay (not shown here).

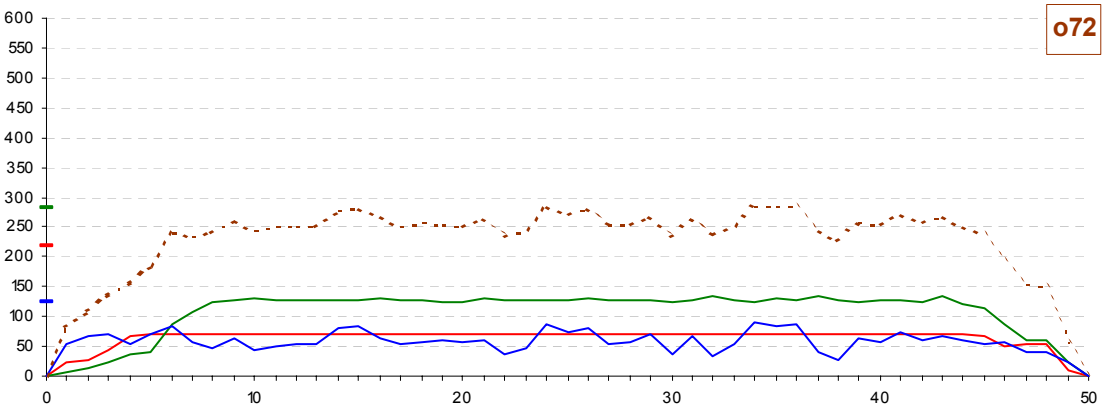


Chart 20: Case Study Run #4 – Port 72

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o72	Px	622		Pz	285 km	1587.1 microsec

Queue	Bandwidth	Depth	Used: Max	Mean	Dropped
EF	35% 217.7 Mbps	3ms 224640 Bytes	0.68%	0.30%	0
AF	45% 279.9 Mbps	10ms 748800 Bytes	0.21%	0.05%	0
BE	20% 124.4 Mbps	15ms 1123200 Bytes	0.14%	0.03%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
V	EF	18.9	1.4	152.6	25.9	0		
W	EF	19.8	1.6	302.3	61.4	0		
X	EF	18.9	0.9	222.3	69.9	0		
Y	EF	20.1	1.4	315.8	96.4	0		
M	AF	20.1	1.7	1393.0	177.2	0		
N	AF	20.1	1.8	850.2	218.7	0		
O	AF	20.1	1.5	3042.9	879.0	0		
P	AF	20.2	1.1	978.7	322.5	0		
Q	AF	20.1	1.1	317.1	99.1	0		
A	BE	25.2	2.1	12046.4	4494.2	0		
B	BE	25.1	1.9	9301.1	1870.4	0		
C	BE	22.7	2.1	15753.9	4995.9	0		
D	BE	32.3	2.4	11239.9	4390.4	0		
E	BE	33.3	2.6	1653.4	309.9	0		

8 VPN Case Study – Run #5 – TCP Congestion

Here is a scenario similar to the initial run excepted that we have tuned slightly differently the scheduling parameters at port 82:

Bandwidth percentages are respectively 10,67,23 for EF, AF and BE classes instead of 10,70,20. Besides AF queue depth is increased from 10ms to 12ms.

This tuning enable us to overflow the AF queue when we introduce an extra EF flow ("Z") that crosses port 82. Congestion avoidance mechanisms then apply to AF TCP flows.

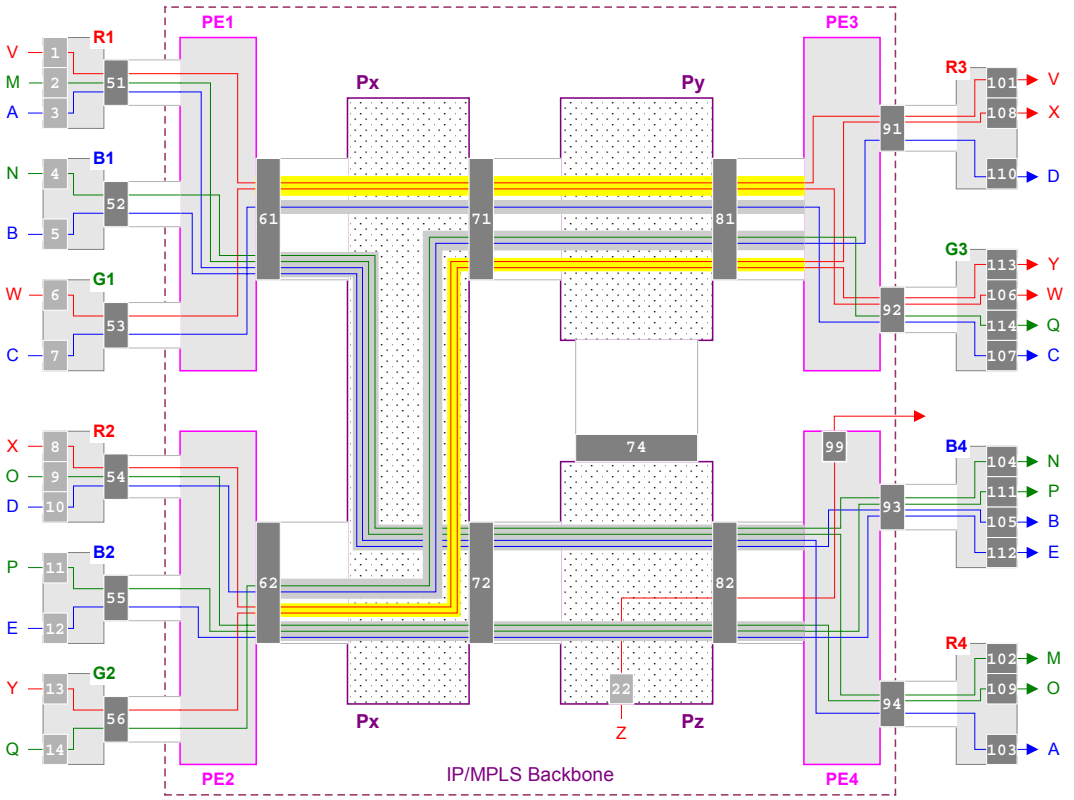


Figure 16: VPN Case Study – Run #5 Configuration

Since many packet discarding occur at the end of the TCP sessions, there are timeouts for the last packets (the destinations do not receive data and therefore do not send duplicate ACKs anymore, that would trigger fast recovery).

It should be noticed that this version of IPVCoSS (as explained in annex 1) does not implement yet Selective ACKs (RFC2018) nor partial ACKs (RFC2582 New Reno) nor timeout dynamic adaptation (RFC2988).

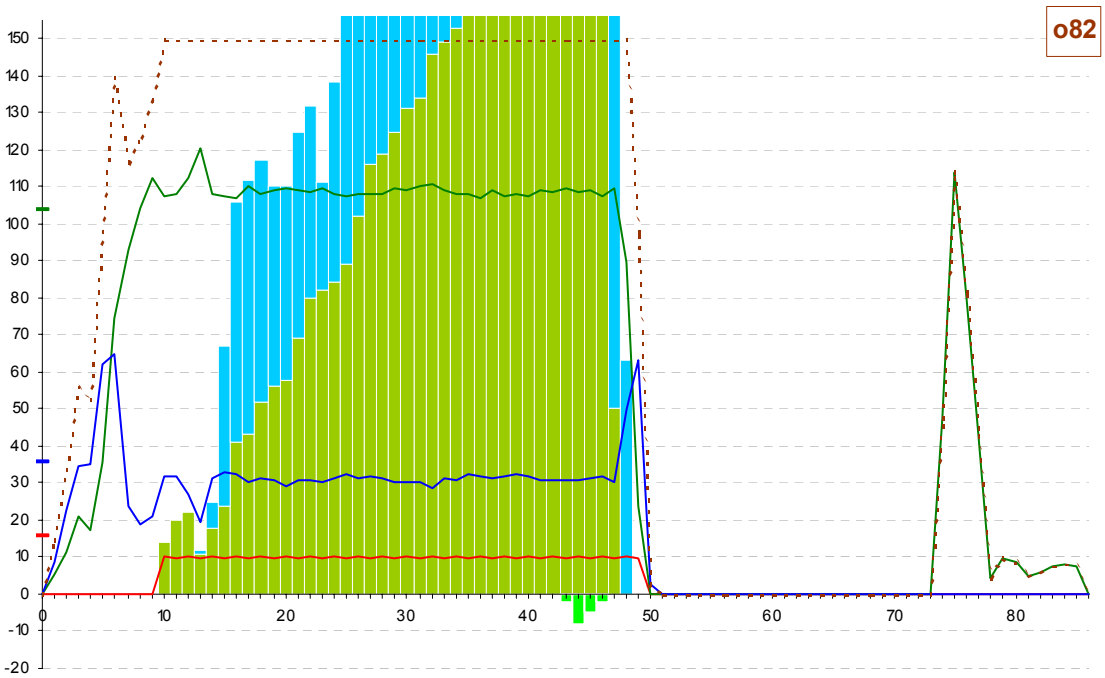


Chart 21: Case Study Run #5 – Port 82

The TCP timeout is fixed and purposely reduced to 30ms. Obviously the final throughput of AF TCP flows dramatically fall down. It is not shown in these reports but here are the values, in Mbps, compared to (between parenthesis) those obtained with the initial run:

M:	*18*	*(28.4)*
N:	*13.2*	*(21.4)*
O:	*13.5*	*(22.2)*
P:	*18.5*	*(29.3)*

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o82	Pz	155		PE4	425 km	2366.7 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	10%	15.5 Mbps	3ms	56160 Bytes	1.80%	0.90%	0
AF	67%	103.8 Mbps	12ms	224640 Bytes	100.45%	42.12%	21
BE	23%	35.6 Mbps	15ms	280800 Bytes	60.30%	24.23%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
Z	EF	245.8	29.8	245.8	29.8	0		
M	AF	16710.0	7323.3	17280.7	7478.1	6	6	204
N	AF	16598.8	7511.5	16728.2	7729.6	2	1	59
O	AF	16576.8	6791.6	17794.3	7703.5	8	7	82
P	AF	16584.4	7422.9	17244.8	7745.0	5	4	55
A	BE	40317.1	17949.7	48179.0	22435.5	0		
B	BE	40934.5	16883.4	46578.7	18735.4	0		
E	BE	41074.1	15867.5	41240.8	16178.7	0		

9 VPN Case Study – Run #6 – TCP Congestion Avoidance

Here is a run with the same tunings and the same generated traffic as with the previous run.

Here we will drop packet#360 from P flow in AF queue of port 82.

This simulates a random early detection (RED).

Actually, we have simplified the RED mechanism because we did not apply traffic conditioning at ingress nodes for AF flows, and discriminated and marked packets as low or high profile according to the flow throughput.

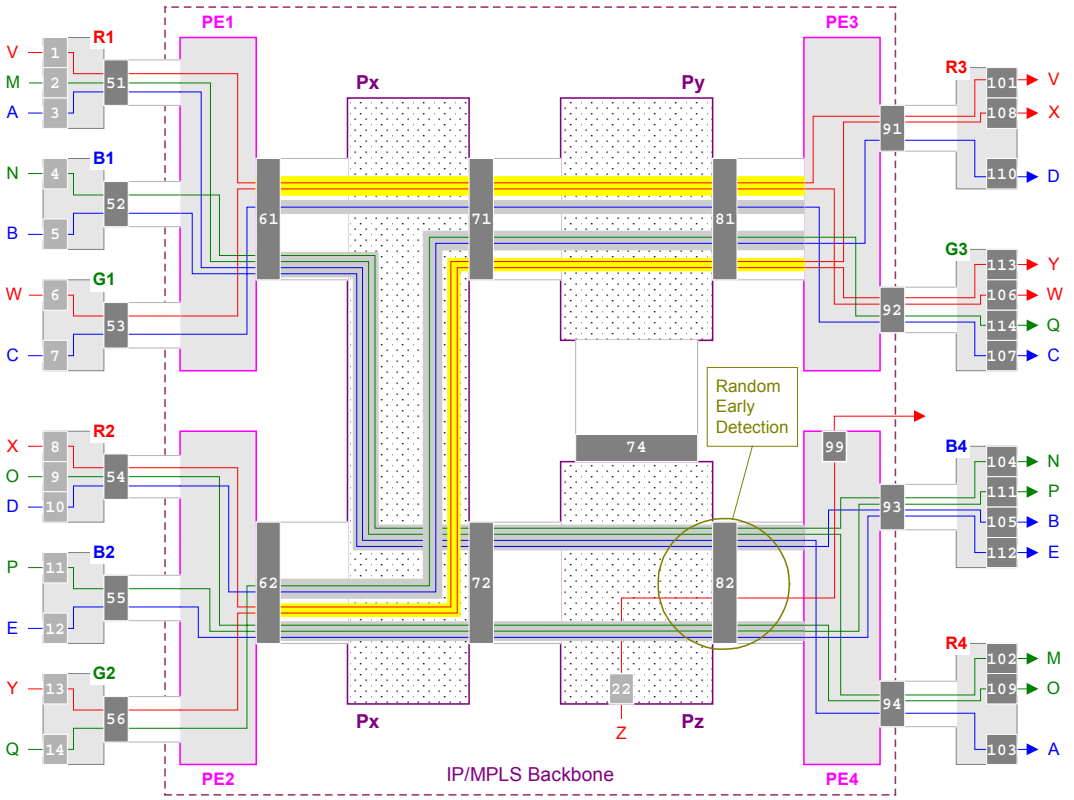


Figure 17: VPN Case Study – Run #6 Configuration

On this chart we can see the time when the packet drop occurs (rounded by a green circle). It happens when the AF queue is filled at 40%.

This test illustrates the benefits of applying RED mechanisms: by anticipating congestion and by dropping at random a packet. A single TCP flow among several ones will be slightly impacted.

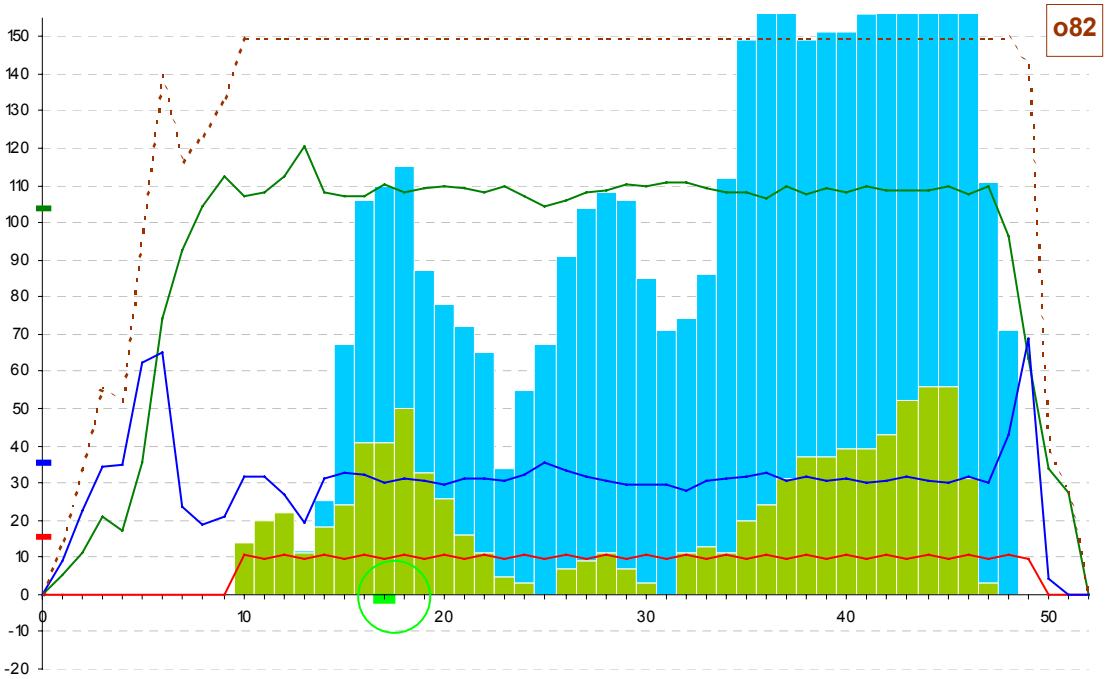


Chart 22: Case Study Run #6 – Port 82

The Fast Retransmit & Fast Recovery congestion avoidance algorithms will be easily applied and prevent a further heavy congestion impacting all the flows.

Only the final throughput of P flow is impacted:

26.6 compared to 29.3 Mbps with the initial run (not shown in adjacent reports).

Port	Node	Rate	Limit	Adjnode	Distance	Propagation-Delay
o82	Pz	155		PE4	425 km	2366.7 microsec

Queue	Bandwidth		Depth		Used: Max	Mean	Dropped
EF	10%	15.5 Mbps	3ms	56160 Bytes	1.80%	0.90%	0
AF	67%	103.8 Mbps	12ms	224640 Bytes	27.00%	9.35%	1
BE	23%	35.6 Mbps	15ms	280800 Bytes	59.92%	23.60%	0

Flow	Class	hopJmax	hopJmean	cumJmax	cumJmean	Loss	Retransmitted FastRxmit	Retransmitted TO
Z	EF	228.1	29.6	228.1	29.6	0		
M	AF	4457.0	1603.4	4998.1	1760.4	0		
N	AF	4496.8	1579.3	4522.2	1797.4	0		
O	AF	4467.7	1495.2	5387.2	2470.5	0		
P	AF	4333.4	1392.3	4518.8	1572.4	1	1	0
A	BE	39468.0	17384.6	47975.3	21870.3	0		
B	BE	40817.4	16470.9	45678.1	18322.9	0		
E	BE	40406.2	15755.9	40689.8	16034.7	0		

10 Jitter Bounds for EF traffic over IP/MPLS

With a statistical multiplexing system such as IP, it is impossible to define absolute bounds to jitter, because of the unpredictability of the offered traffic. However, here is a stress simulation test that pushes the limits of EF traffic in a realistic environment, with heavy load but without congestion. Single EF traffic flows are injected at CEs, with various characteristics. Throughput and packet size are indicated in the table overleaf, but other varying information are the access link rate (E3, DS3, FE, STM1) and length. Thus, EF packets enter PE1 and PE2 at random, without previous jitter. Besides EF isochronous traffic flows, we inject variable BE traffic. All these flows converge respectively to STM-4 ports 101 and 102 and load them quite heavily. From P1, the EF traffic crosses port 111 at a higher rate (GE) still mixed with other BE variable flows and then, from P2 to P3, it crosses port 121 at a lower rate (STM-4) before being split to egress PEs and CEs via lower rate links.

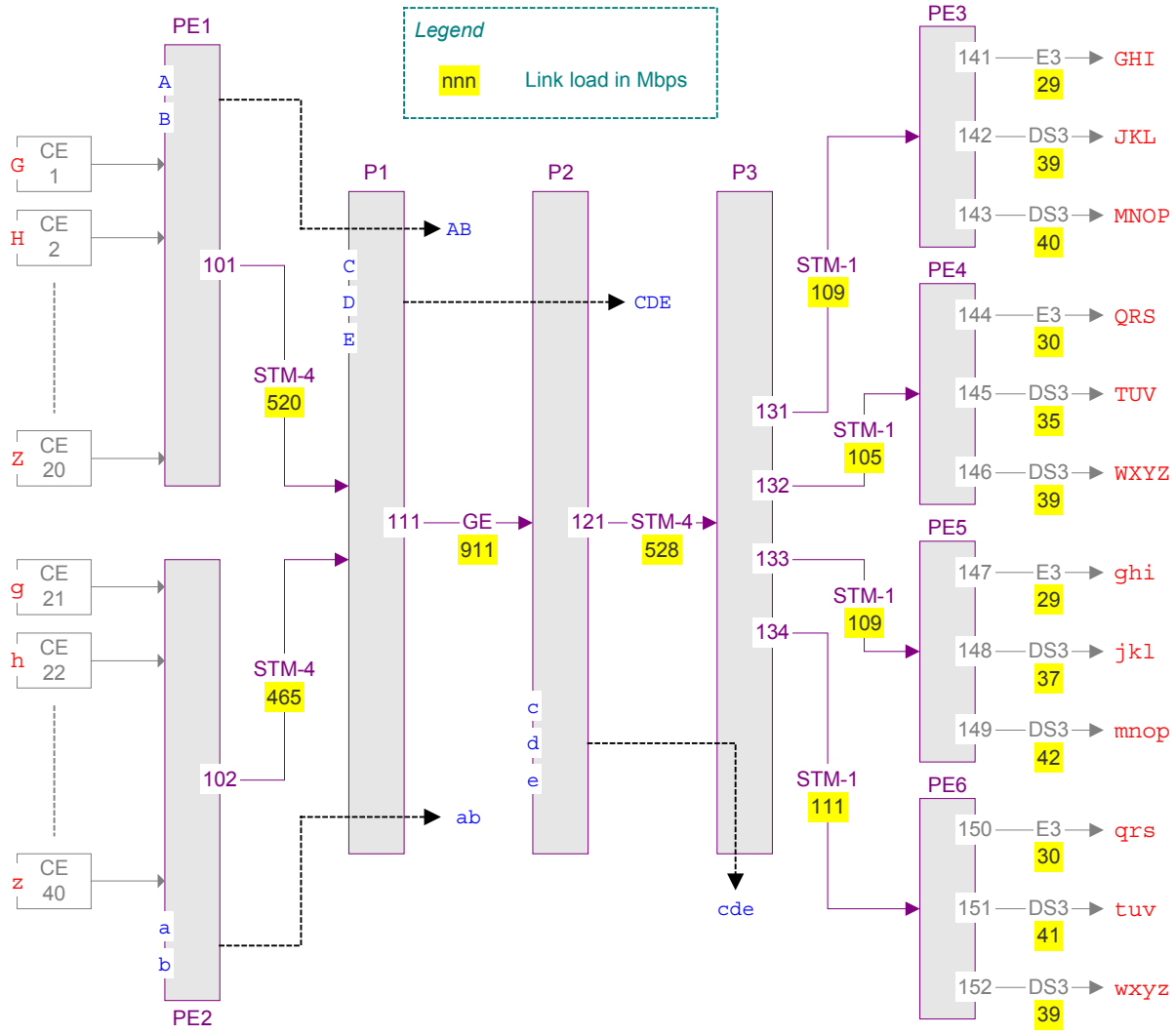


Figure 18: Configuration for testing EF jitter bounds

Table 3 shows the average jitter values experienced by individual EF flows at each crossed port, and the average and maximum jitter from end-to-end. Here are some observations we can derive from this trial about conditions that favor jitter:

- When traffic flows at an output port comes from several upstream ports, as this is the case typically for an edge router, there is a higher probability of having simultaneous ingress packets and therefore short transient queuing, even if bandwidth is globally available.
- When the output port is at a lower rate than an upstream port, the serialization time is longer in output than in input and it is therefore more likely, in case of heavy load, to have a packet waiting for the output port to be free.
- The slower the port rate, the higher the jitter because of a larger serialization time.
- The probability of having jitter increases with port loading, because packet contention in output will occur more often.

All these characteristics of IP traffic can be easily shown and analyzed with the traces associated with short elementary trials (only a few packets) via IPVCoSS.

Flow	Thrput	PKlen	Nb PKs	o101-o102 (STM-4)	o111 (GE)	o121 (STM-4)	o131-o134 (STM-1)	o141 - o152 (E3 / DS3 / DS3...)		
				+Jmean					Jmax	Jmean
G	10	1000	2500	15.4	7.1	30.4	39.3	84.6	622.4	177.0
H	11	1050	2619	15.0	6.2	33.8	42.2	76.4	579.2	173.7
I	8	1100	1819	11.9	5.7	29.4	44.8	90.6	619.5	182.6
J	12	1150	2609	11.9	5.8	31.4	37.0	85.1	614.7	171.4
K	13	1200	2709	10.2	5.5	29.4	35.2	78.9	644.9	159.3
L	14	1250	2800	9.6	5.3	28.7	31.7	70.8	534.1	146.4
M	9	1300	1731	9.2	5.1	26.1	33.7	135.2	756.9	209.5
N	11	1350	2037	7.7	5.0	26.0	27.3	126.1	711.0	192.4
O	13	1400	2322	7.1	4.8	24.3	25.1	115.7	746.8	177.2
P	7	1450	1207	16.8	5.5	32.1	37.9	130.3	700.7	222.7
Q	8	1500	1334	17.4	5.2	29.0	43.5	75.6	607.5	170.9
R	12	1025	2927	14.4	6.1	33.9	42.3	132.0	793.3	228.9
S	10	1075	2326	13.1	6.1	32.8	42.5	129.3	726.0	224.0
T	16	1125	3556	11.2	5.7	30.6	34.3	65.2	610.2	147.3
U	9	1175	1915	10.8	5.7	30.8	39.4	68.0	622.2	154.9
V	10	1225	2041	9.8	5.3	29.5	35.9	60.7	561.5	141.4
W	12	1275	2353	8.4	5.2	27.8	31.9	116.6	718.8	190.0
X	8	1325	1510	8.5	5.2	27.9	33.5	119.2	732.3	194.5
Y	10	1375	1819	7.9	5.2	32.1	25.9	106.4	721.0	177.7
Z	9	1425	1579	16.7	5.6	30.9	38.1	111.8	696.1	203.3
g	6	1000	1500	15.7	6.3	33.2	49.3	199.5	683.9	304.3
h	11	1100	2500	15.4	6.5	56.6	99.6	236.2	699.0	414.5
i	12	1200	2500	10.4	5.9	46.3	62.8	25.2	434.6	150.9
j	13	1300	2500	10.5	6.1	29.6	12.5	53.7	392.7	112.6
k	14	1400	2500	8.8	5.6	32.7	45.8	190.1	563.0	283.2
l	10	1500	1667	9.1	4.7	24.5	33.3	49.1	361.5	121.0
m	12	1000	3000	7.9	5.4	29.5	48.5	149.4	680.4	240.9
n	7	1100	1591	7.2	5.0	29.1	45.0	141.3	732.9	227.7
o	14	1200	2917	5.8	5.0	26.4	32.9	127.3	647.9	197.6
p	9	1300	1731	14.9	6.7	39.1	42.9	117.1	689.4	220.9
q	16	1400	2857	14.0	5.2	29.9	39.8	106.4	662.4	195.5
r	6	1500	1000	16.3	5.8	20.9	51.7	244.0	697.5	339.0
s	8	1000	2000	9.1	5.7	20.2	38.4	108.1	541.7	181.7
t	10	1100	2273	11.4	5.9	33.0	39.9	120.3	642.8	210.6
u	16	1200	3334	8.5	5.2	26.8	37.0	113.1	591.2	190.8
v	15	1300	2885	8.7	5.1	27.9	32.5	95.4	594.1	169.8
w	14	1400	2500	18.5	6.4	40.6	29.3	145.6	553.1	240.6
x	6	1500	1000	9.0	5.1	48.3	30.0	100.8	513.2	193.4
y	8	1000	2000	6.3	5.3	20.6	61.8	114.9	577.8	209.1
z	11	1100	2500	12.4	6.1	23.2	41.2	89.6	496.5	172.8
A	100	1500	31658	36.9						
B	200	1500	63222	34.8						
a	90	1500	28330	33.2						
b	150	1500	47377	32.9						
C	100	1500	31590		49.2					
D	200	1500	63376		48.2					
E	150	1500	47392		48.1					
c	20	1500	6302			73.9				
d	30	1500	9460			71.0				
e	40	1500	12681			70.5				

Table 3: Examples of jitter values with numerous EF flows, heavy load but no congestion

11 Conclusion

This paper has presented a case study, with the simulation of a fairly sophisticated network (SP backbone and VPN sites). Readers interested in analyzing accurate traces will find various elementary tests via IPVCoSS, with their full traces, at the web site mentioned on the front page of this document.

Our purpose was to show how jitter could be controlled, to some very acceptable extent, via an IP/MPLS backbone, in spite of IP flows with different characteristics, especially the variable packet size. Obviously, the real world is more complex and, for this case study, we did not take into account link redundancy, enabling more alternate paths. We did not consider techniques such as load balancing, or provider edge's front-end access equipment such as Ethernet switches. However, the main principles have been analyzed. Concerning the packet scheduling at a port, the method itself – in spite of the name defining it – is very dependent on the vendor's implementation. That is why the class-based queuing method implemented by this simulator is explained in an annex with the help of traces.

The engineering of the access links between a VPN site and the SP backbone is a major point in the delivery of the most stringent classes of services to VPN users. Actually, the access link (or the access network) introduces a different set of parameters for an IP flow's end-to-end path, compared to the SP network, which can be considered homogeneous in terms of equipment type and capacity. For economical reasons, the customer will choose the access link at the lowest possible rate. For example, an E3 leased line will entail an E3 interface for which the serialization time of a packet will be three times longer than with a Fast Ethernet interface; it will therefore be prone to jitter because of the higher rate interfaces – upstream or downstream – in the backbone. On the other hand, an Ethernet-based access loop subscribed at an "E3" rate (say 30 Mbps) will enable a faster packet serialization, but will raise other issues:

- The structure of the access network, which may be shared by several access loop's customers and which may introduce several elements to be crossed, such as ATM switches with, for example, a multi-LAN service.
- The capability of the CE and PE equipment's scheduler to take into account the subscribed rate instead of the physical interface rate. The simulator in this case study has treated this latter situation.

Even with the best network and access link tuning, jitter is dependent on traffic contingency and network loading. It is significantly minimized with high-speed interfaces and stay within acceptable limits, considering the de-jittering capabilities of specialized equipment in reception. We could also imagine some form of traffic shaping in output of each node along the path, that would totally eliminate jitter, but this seems unnecessary.

Maybe the most important issue is the impact of transiting from one situation to another, that is passing from one path to another as this is the case with DiffServ Traffic Engineering (DS-TE). This is not treated in this paper because the simulator itself focuses on the data plane and the performance aspects of pre-established paths. It ignores completely the control plane. Besides the "throughput", "jitter", "packet loss" and "delay" classical QoS parameters, there is a most important parameter, which is "availability". Availability can be considerably enhanced by techniques such as DS-TE, but there will still be the punctual impact, for a real-time video for instance, of the path transition due to a router failure or link failure within the network.

This network simulator was specifically developed for writing this White Paper, and having the suitable traces, graphs and reports. There are more elaborated network simulators, such as the famous NS-2 that helped evaluate – on a large scale – TCP congestion avoidance mechanisms. Network simulation, for an SP, cannot replace real tests on a dedicated platform representing its network and customers. However, the simulator could certainly supplement it by helping structure the overall topology of the test platform and find a first level of tuning for the QoS parameters.

Annex 1: IPVCoSS – TCP Congestion Control

Most IP flows are based on TCP transport protocol. These flows are responsive to network load thanks to congestion control mechanisms. RFC 2581 defines 4 main algorithms:

- Slow Start
- Congestion Avoidance
- Fast Retransmit
- Fast Recovery

One must retain that at any time either "Slow Start" or "Congestion Avoidance" is applied. "Fast Retransmit" and "Fast Recovery" are tightly related ("Fast Retransmit" is followed by "Fast Recovery") and are an enhancement of "Congestion Avoidance".

IPVCoSS takes into account these mechanisms but, for simplification and in order to focus on the essential aspects, the following hypothesis and restrictions are applied:

1) Only the data phase of a TCP session is considered, and the parameters that would result of the initial establishment phases are defined when configuring the TCP-based IP flow or forced to some default values.

2) There are no limitations on the receiver side and we assume that:
   - The receiver window (rwnd) is very high
   - RMSS (receiver maximum segment size) is greater than or equal to SMSS (sender maximum segment size)

3) The segment size (SMSS / RMSS: sender / receiver maximum segment size) does not include the TCP header but in the context of IPVCoSS, we will ignore the TCP header as well as the IP header and we will assimilate the TCP segment size to the IP packet size, which itself includes its IP header. For example, if a packet size of 1000 bytes is configured, we will show in the TCP specific traces a segment size of 1000 bytes, instead of 940 bytes (20 bytes of IP header plus 40 bytes of TCP header). The advantage of this "deviation" is that we can rely on the throughput calculations performed on IP packets, and that the evolution of the TCP parameters such as cwnd and flightsize shown in the TCP traces will be more readable. This abstraction does not prevent the application of the congestion control mechanisms we are examining. Implementing a true TCP layer would be far too complex to reach our goal. Moreover, if it is very likely that the IP header will have no options field, the TCP header in contrast would include the options required for high-speed and large-scale environments (larger window size) and would be greater than 40 bytes.

4) The delay back for a TCP "ACK" is calculated at the time of the first TCP segment transmission and is then systematically applied to all other "ACK" for this flow. This means that we assume that, in addition to having the same path from receiver to sender, there is no congestion on this path. This is consistent with the main assumptions made for the IPVCoSS environment: MPLS paths, no unavailability. Moreover, for a clear observation, it is better not to cross flows even if it is feasible.

5) The initial sequence number (each TCP byte is identified by a sequence number) will always start to 1, conversely to TCP real implementations. The purpose, once again, is to offer a better readability of the traces.

6) The receiver will send an ACK for each received segment: there is no deferred acknowledgment.

7) The timeout is fixed to 300ms for easing observations.

In contrast, the following key variables, on the sender side, are processed in full conformance with TCP rules:

- cwnd (congestion window)

  The number of bytes of this window limits the transmission of TCP data: a TCP sender cannot send any fragment with a sequence number greater that the sum of the last acknowledged sequence number and this window value. The initial value of this parameter corresponds to 2 full-size segments. This value is incremented when TCP ACKs are received but can be reduced when congestion is detected.

- fs (flight size)

  The amount of data that has been sent but not yet acknowledged.

- ssthresh (slow start threshold)

  The initial value of this threshold may be very high and is reduced when congestion is detected.

At any time, the respective values of ssthresh and cwnd determine which algorithm should be applied: SLOW START or CONGESTION AVOIDANCE:

```
cwnd <= ssthresh   : SLOW START
cwnd >  ssthresh   : CONGESTION AVOIDANCE
```

The difference between Slow Start and Congestion Avoidance is the evolution of cwnd according to received ACKs, and as a result controls the amount of data injected in the network:

- With SLOW START: cwnd is incremented by SMSS for each received ACK
- With CONGESTION AVOIDANCE: cwnd is incremented by SMSS at each RTT (round trip time) only; this is given by the following equation:

```
cwnd += SMSS * SMSS / cwnd
```

The following picture shows the evolution of the useful window that governs the injection of data:

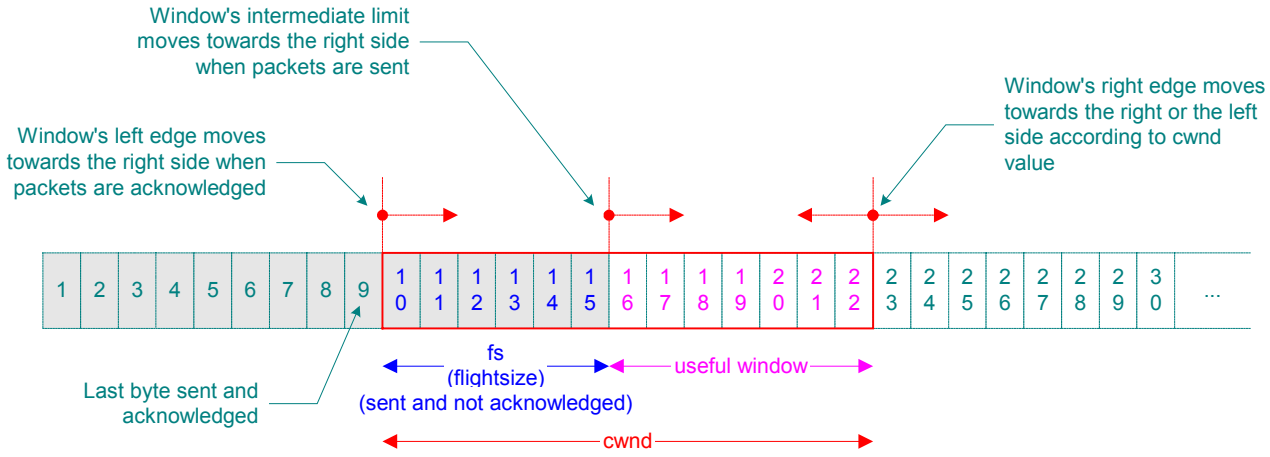


Figure 19: TCP Sender Windows

Congestion detection previously mentioned for cwnd and ssthresh variables is the strong presumption of packet loss in following cases:

1) Acknowledgement failure

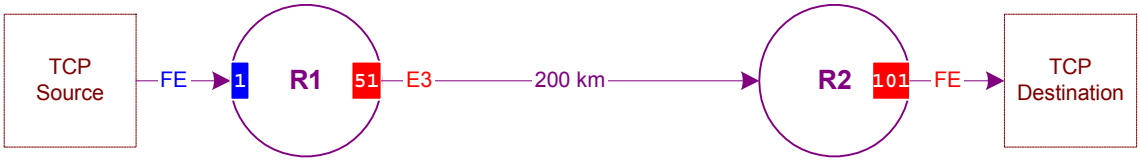
   When 3 consecutive duplicate ACKs are received for a segment, the following segment which acknowledgment was expected by the sender is considered lost.

2) Time-Out for a segment

   A time-out indicates a strong congestion in the network that prevents the communication between sender and receiver. A TCP time-out value is expressed in terms of seconds and is permanently re-negotiated for taking into account the traffic load.

We will concentrate on the first case only since it is tightly related to the RED (random early detection) queue management mechanism.

We will use the following configuration for first examining the Slow Start mechanism when applied at the beginning of a TCP data transfer phase. Then we will see the consequences of a single packet loss with the same traffic profile.



TCP traces are shown as time-diagrams in order to highlight the exchanges between sender and receiver. The evolution of cwnd and flightsize (fs) variables appears on the left of the events on the sender side. The IP packet identifier along with the first and last byte of the segment represents TCP data segments. The “push” event on the receiver side represents the communication, in the right order, of the segment contents to the application. An acknowledgement conveys the number of the next byte that the receiver expects to receive.

A1.1 – Slow Start

Here is a trace of the “slow” start of a TCP session. For clarity, segments have a fixed size of 1000 bytes and targeted at a steady throughput of 32 Mbps. The ssthresh variable is initialized at a very high value so as to immediately trigger the slow start algorithm.

The variable cwnd is initialized to 2 x SMSS, i.e. 2000 bytes. Thus, 2 segments (A1, A2) are sent (and flightsize is incremented) before the window is closed.

For each acknowledged segment, cwnd is incremented by 1 x SMSS, while flightsize is decremented.

As a result, 4 new segments will be sent at 250 microsecond intervals corresponding to the 32Mbps throughput.

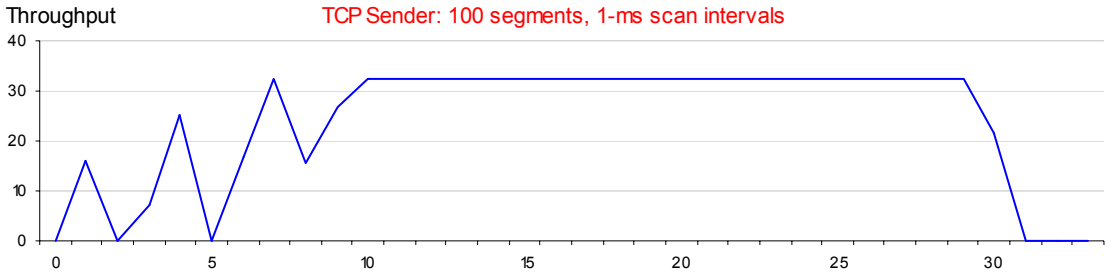
When A3 to A6 segments are acknowledged, we can send again 8 new segments (A7 to A14)

When A8 to A14 are acknowledged the window is still wider and enables TCP segments to be sent at the targeted throughput.

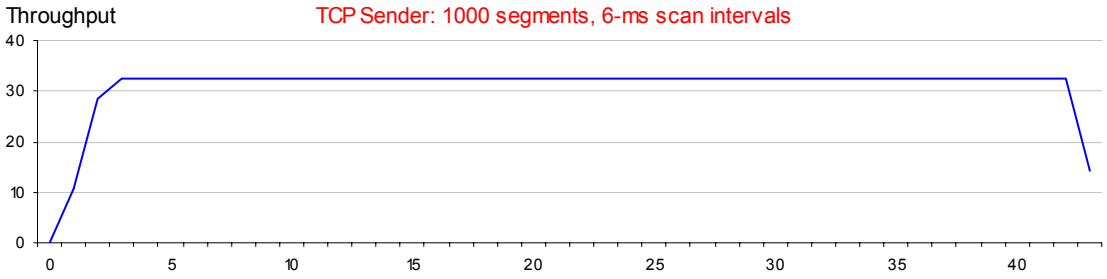
tick	cwnd	fs	TCP Sender		TCP Receiver	
0.0	2000	1000	ts A1(1-1000)	->		
250.0		2000	ts A2(1001-2000)	->		
1552.8					-> tr A1(1-1000)	-> push A1
					<- ta A2(1001)	
1802.8					-> tr A2(1001-2000)	-> push A2
					<- ta A3(2001)	
2689.6	3000	1000	ta A2(1001)	<-		
		2000	ts A3(2001-3000)	->		
2939.6	4000	1000	ta A3(2001)	<-		
		2000	ts A4(3001-4000)	->		
3189.6		3000	ts A5(4001-5000)	->		
3439.6		4000	ts A6(5001-6000)	->		
4242.4					-> tr A3(2001-3000)	-> push A3
					<- ta A4(3001)	
4492.4					-> tr A4(3001-4000)	-> push A4
					<- ta A5(4001)	
4742.4					-> tr A5(4001-5000)	-> push A5
					<- ta A6(5001)	
4992.4					-> tr A6(5001-6000)	-> push A6
					<- ta A7(6001)	
5379.2	5000	3000	ta A4(3001)	<-		
		4000	ts A7(6001-7000)	->		
5629.2	6000	3000	ta A5(4001)	<-		
		4000	ts A8(7001-8000)	->		
5879.2	7000	3000	ta A6(5001)	<-		
		4000	ts A9(8001-9000)	->		
6129.2	8000	3000	ta A7(6001)	<-		
		4000	ts A10(9001-10000)	->		
6379.2		5000	ts A11(10001-11000)	->		
6629.2		6000	ts A12(11001-12000)	->		
6879.2		7000	ts A13(12001-13000)	->		
6932.0					-> tr A7(6001-7000)	-> push A7
					<- ta A8(7001)	
7129.2		8000	ts A14(13001-14000)	->		
7182.0					-> tr A8(7001-8000)	-> push A8
					<- ta A9(8001)	
7432.0					-> tr A9(8001-9000)	-> push A9
					<- ta A10(9001)	
7682.0					-> tr A10(9001-10000)	-> push A10
					<- ta A11(10001)	
7932.0					-> tr A11(10001-11000)	-> push A11
					<- ta A12(11001)	
8068.8	9000	7000	ta A8(7001)	<-		
		8000	ts A15(14001-15000)	->		
8182.0					-> tr A12(11001-12000)	-> push A12
					<- ta A13(12001)	
8318.8	10000	7000	ta A9(8001)	<-		
		8000	ts A16(15001-16000)	->		
8432.0					-> tr A13(12001-13000)	-> push A13
					<- ta A14(13001)	
8568.8	11000	7000	ta A10(9001)	<-		
		8000	ts A17(16001-17000)	->		
8682.0					-> tr A14(13001-14000)	-> push A14
					<- ta A15(14001)	
8818.8	12000	7000	ta A11(10001)	<-		
		8000	ts A18(17001-18000)	->		
9068.8	13000	7000	ta A12(11001)	<-		
		8000	ts A19(18001-19000)	->		
9318.8	14000	7000	ta A13(12001)	<-		
		8000	ts A20(19001-20000)	->		
9568.8	15000	7000	ta A14(13001)	<-		
		8000	ts A21(20001-21000)	->		
9621.6					-> tr A15(14001-15000)	-> push A15
					<- ta A16(15001)	
9818.8	16000	7000	ta A15(14001)	<-		
		8000	ts A22(21001-22000)	->		
9871.6					-> tr A16(15001-16000)	-> push A16
					<- ta A17(16001)	
10068.8		9000	ts A23(22001-23000)	->		
10121.6					-> tr A17(16001-17000)	-> push A17
					<- ta A18(17001)	

tick	cwnd	fs	TCP Sender		TCP Receiver	
10318.8		10000	ts A24(23001-24000)	->		
10371.6					-> tr A18(17001-18000)	-> push A18
					<- ta A19(18001)	
10568.8		11000	ts A25(24001-25000)	->		
10621.6					-> tr A19(18001-19000)	-> push A19
					<- ta A20(19001)	
10758.4	17000	10000		ta A16(15001) <-		
10818.8		11000	ts A26(25001-26000)	->		
10871.6					-> tr A20(19001-20000)	-> push A20
					<- ta A21(20001)	
11008.4	18000	10000		ta A17(16001) <-		
11068.8		11000	ts A27(26001-27000)	->		
11121.6					-> tr A21(20001-21000)	-> push A21
					<- ta A22(21001)	
11258.4	19000	10000		ta A18(17001) <-		
11318.8		11000	ts A28(27001-28000)	->		
11371.6					-> tr A22(21001-22000)	-> push A22
					<- ta A23(22001)	
11508.4	20000	10000		ta A19(18001) <-		
11568.8		11000	ts A29(28001-29000)	->		
11621.6					-> tr A23(22001-23000)	-> push A23
					<- ta A24(23001)	
11758.4	21000	10000		ta A20(19001) <-		
11818.8		11000	ts A30(29001-30000)	->		
...						

The first graph with a short scan-interval of 1 millisecond shows the successive bursts of TCP segments due to the Slow Start algorithm.



This second graph with a longer trial and therefore a larger scan-interval shows a line chart more conventional in regards to Slow Start description.



A1.2 – Congestion Avoidance with Fast Retransmit & Fast Recovery

Here is the same trial as previously with the discarding of TCP segment #50. The trace is shown from the packet discarding till a little after the end of the Fast Recovery procedure. Note that “o-o-o” on the receiver side means “out-of-order”.

At this stage, cwnd is still increasing...

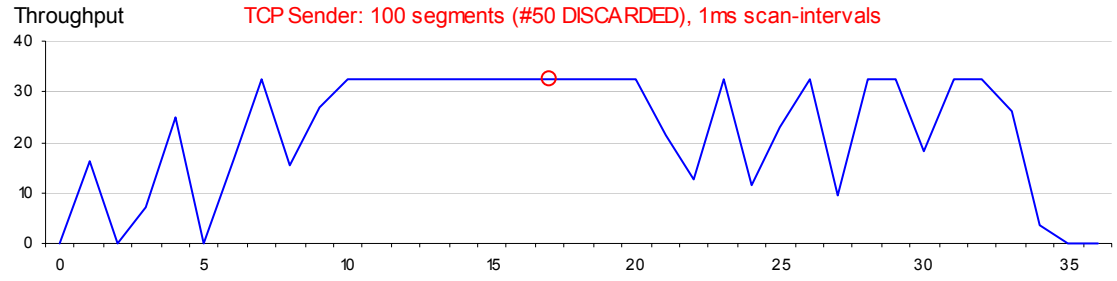
tick	cwnd	fs	TCP Sender		TCP Receiver	
16818.8		11000	ts A50(49001-50000)	->		
16871.6					-> tr A44(43001-44000)	-> push A44
					<- ta A45(44001)	
16920.9				A50 DISCARD SIMULATION		
17008.4	42000	10000		ta A41(40001) <-		
17068.8		11000	ts A51(50001-51000)	->		
17121.6					-> tr A45(44001-45000)	-> push A45
					<- ta A46(45001)	
17258.4	43000	10000		ta A42(41001) <-		
17318.8		11000	ts A52(51001-52000)	->		
17371.6					-> tr A46(45001-46000)	-> push A46
					<- ta A47(46001)	
17508.4	44000	10000		ta A43(42001) <-		
17568.8		11000	ts A53(52001-53000)	->		
17621.6					-> tr A47(46001-47000)	-> push A47

	tick	cwnd	fs	TCP Sender		TCP Receiver	
						<- ta A48(47001)	
	17758.4	45000	10000	ta A44(43001) <-			
	17818.8		11000	ts A54(53001-54000) ->			
	17871.6					-> tr A48(47001-48000)	-> push A48
						<- ta A49(48001)	
	18008.4	46000	10000	ta A45(44001) <-			
	18068.8		11000	ts A55(54001-55000) ->			
	18121.6					-> tr A49(48001-49000)	-> push A49
From now, A50 is expected by the receiver...						<- ta A50(49001)	
	18258.4	47000	10000	ta A46(45001) <-			
	18318.8		11000	ts A56(55001-56000) ->			
	18508.4	48000	10000	ta A47(46001) <-			
	18568.8		11000	ts A57(56001-57000) ->			
A51 and subsequent segments cannot be delivered to the TCP user and are stored, while A50 expectation is permanently notified in the ACKs	18621.6					-> tr A51(50001-51000)	#o-o-o
						<- ta A50(49001)	
	18758.4	49000	10000	ta A48(47001) <-			
	18818.8		11000	ts A58(57001-58000) ->			
	18871.6					-> tr A52(51001-52000)	#o-o-o
						<- ta A50(49001)	
	19008.4	50000	10000	ta A49(48001) <-			
	19068.8		11000	ts A59(58001-59000) ->			
	19121.6					-> tr A53(52001-53000)	#o-o-o
						<- ta A50(49001)	
	19258.4	51000	10000	ta A50(49001) <-			
	19318.8		11000	ts A60(59001-60000) ->			
	19371.6					-> tr A54(53001-54000)	#o-o-o
						<- ta A50(49001)	
	19568.8		12000	ts A61(60001-61000) ->			
	19621.6					-> tr A55(54001-55000)	#o-o-o
						<- ta A50(49001)	
	19758.4			DA(1) ta A50(49001) <-			
	19818.8		13000	ts A62(61001-62000) ->			
	19871.6					-> tr A56(55001-56000)	#o-o-o
						<- ta A50(49001)	
	20008.4			DA(2) ta A50(49001) <-			
	20068.8		14000	ts A63(62001-63000) ->			
After the third Duplicate ACK (DA) TCP segment A50 is retransmitted.	20121.6					-> tr A57(56001-57000)	#o-o-o
						<- ta A50(49001)	
	20258.4			DA(3) ta A50(49001) <-			
				FAST RETRANSMIT			
Then the Fast Recovery phase is entered, with ssthresh value set down to half the flightsize value: 14000/2 = 7000				ts A50(49001-50000) ->			
				ENTER FAST RECOVERY			
				ssthresh:7000			
	20371.6					-> tr A58(57001-58000)	#o-o-o
						<- ta A50(49001)	
	20508.4	11000		DA(4) ta A50(49001) <-			
	20621.6					-> tr A59(58001-59000)	#o-o-o
						<- ta A50(49001)	
For each additional duplicate ACK received, cwnd is incremented by SSMS.	20758.4	12000		DA(5) ta A50(49001) <-			
	20871.6					-> tr A60(59001-60000)	#o-o-o
						<- ta A50(49001)	
	21008.4	13000		DA(6) ta A50(49001) <-			
	21121.6					-> tr A61(60001-61000)	#o-o-o
						<- ta A50(49001)	
	21258.4	14000		DA(7) ta A50(49001) <-			
	21371.6					-> tr A62(61001-62000)	#o-o-o
We can send a TCP segment...						<- ta A50(49001)	
	21508.4	15000	15000	DA(8) ta A50(49001) <-			
				ts A64(63001-64000) ->			
	21621.6					-> tr A63(62001-63000)	#o-o-o
When A50 is received, A50 to A63 segments can be delivered (pushed) to the TCP user						<- ta A50(49001)	
	21758.4	16000	16000	DA(9) ta A50(49001) <-			
				ts A65(64001-65000) ->			
	21856.5					-> tr A50(49001-50000)	-> push A50-A63
						<- ta A64(63001)	
	22008.4	17000		DA(10) ta A50(49001) <-			
			17000	ts A66(65001-66000) ->			
	22258.4	18000		DA(11) ta A50(49001) <-			
			18000	ts A67(66001-67000) ->			
When the ACK that acknowledges new data arrives, the Fast Recovery phase terminates.	22508.4	19000		DA(12) ta A50(49001) <-			
			19000	ts A68(67001-68000) ->			
	22758.4	20000		DA(13) ta A50(49001) <-			
			20000	ts A69(68001-69000) ->			
	22993.3	7000	6000	ta A64(63001) <-			
				EXIT FAST RECOVERY			
	23008.4		7000	ts A70(69001-70000) ->			
	23061.2					-> tr A64(63001-64000)	-> push A64

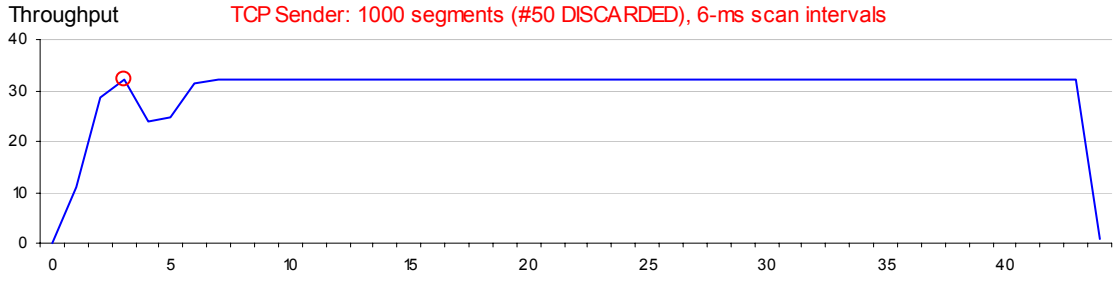
However, since the ssthresh value is still set to 7000 bytes, we have still to apply the classical congestion avoidance algorithm: cwnd is no more incremented by 1xSMSS for each ACK received, but it is incremented by 1xSMSS per RTT (round-trip time) according to the formula mentioned earlier.

tick	cwnd	fs	TCP Sender	TCP Receiver	
				<- ta A65(64001)	
23311.2				-> tr A65(64001-65000)	-> push A65
				<- ta A66(65001)	
23561.2				-> tr A66(65001-66000)	-> push A66
				<- ta A67(66001)	
23811.2				-> tr A67(66001-67000)	-> push A67
				<- ta A68(67001)	
24061.2				-> tr A68(67001-68000)	-> push A68
				<- ta A69(68001)	
24198.0	8000	6000	ta A65(64001) <-		
		7000	ts A71(70001-71000) ->		
24311.2				-> tr A69(68001-69000)	-> push A69
				<- ta A70(69001)	
24448.0	8125	6000	ta A66(65001) <-		
		7000	ts A72(71001-72000) ->		
24561.2				-> tr A70(69001-70000)	-> push A70
				<- ta A71(70001)	
24698.0	8248	6000	ta A67(66001) <-		
		7000	ts A73(72001-73000) ->		
24948.0	8369	6000	ta A68(67001) <-		
		7000	ts A74(73001-74000) ->		
25198.0	8488	6000	ta A69(68001) <-		
		7000	ts A75(74001-75000) ->		
25448.0	8605	6000	ta A70(69001) <-		
		7000	ts A76(75001-76000) ->		
25698.0	8721	6000	ta A71(70001) <-		
		7000	ts A77(76001-77000) ->		
25750.8				-> tr A71(70001-71000)	-> push A71
				<- ta A72(71001)	
25948.0		8000	ts A78(77001-78000) ->		
26000.8				-> tr A72(71001-72000)	-> push A72
				<- ta A73(72001)	
26250.8				-> tr A73(72001-73000)	-> push A73
				<- ta A74(73001)	
26500.8				-> tr A74(73001-74000)	-> push A74
				<- ta A75(74001)	
26750.8				-> tr A75(74001-75000)	-> push A75
				<- ta A76(75001)	
26887.6	8835	7000	ta A72(71001) <-		
		8000	ts A79(78001-79000) ->		
27000.8				-> tr A76(75001-76000)	-> push A76
				<- ta A77(76001)	
27137.6	8948	7000	ta A73(72001) <-		
		8000	ts A80(79001-80000) ->		
...					

The first graph with a short scan-interval of 1 millisecond shows in detail the impact of a packet discard, represented by a red circle, in the middle of the flow.



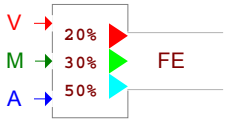
This second graph with a longer trial and therefore a larger scan-interval provides a broader view of the impact of the fast retransmit and fast recovery algorithms.



Annex 2: IPVCoSS – Scheduling Trace

Here is the simple configuration used to illustrate the scheduling algorithm used by IPVCoSS. This port will be heavily overloaded with EF traffic just above its assigned bandwidth (because of the frame overhead) and AF and BE traffic well above their assigned bandwidths. Considering the bandwidth ratios assigned to this FE port, the credits for EF, AF and BE queues will be respectively 20 000, 30 000 and 50 000 bits. These credits, as shown by 'q+' trace events, are refreshed at 1-ms regular intervals. As soon as the port is free (i.e. there is no current serialization and the interframe gap, for Ethernet frames, is respected) the Scheduler will select one packet among all the packets, if any, waiting in the queues.

CBR	20 Mbit/s	PK size: 1500 bytes
VBR	40 Mbit/s	Max PK size: 500 bytes
VBR	90 Mbit/s	Max PK size: 1000 bytes



The Scheduler will scan the queues according to their relative priority: EF > AF > BE. In a first step, it will select the first packet (if any) in the queue only if the credit allows it. Otherwise, if the first pass was unsuccessful, in a second pass it will select the first packet of the queue, thus basing its choice on priority only. Here is the trace for the first 2 milliseconds of the trial.

Events of interest, besides 'q+', are:

- 'qo' the packet placed in the queue
- 'so' serialization of the packet selected by the scheduler (highlighted in yellow)
- 'eo' end of serialization of the packet's frame

For each queue the following parameters are displayed, when their value changes: the current credit in bits ('Credit'), the packet at the head of the queue ('HdPk') eligible to selection by the scheduler, its frame size in bits ('Frlen'), and the number of packets in the queue ('Qn').

Note: The interframe gap with a fast Ethernet is 0.9 microsecond and therefore an 'eo' event cannot be immediately followed by an 'so' event.

V1 packet only stays 0.2 microsecond in queue. It arrived during the interframe gap while another packet (M2) was waiting and otherwise would have been serialized.

Tick	Ev	Pkid	EF Queue Credit	EF HdPk	EF Frlen	EF Qn	AF Queue Credit	AF HdPk	AF Frlen	AF Qn	BE Queue Credit	BE HdPk	BE Frlen	BE Qn
0.0	q+		20000				30000				50000			
28.0	qo	M1						M1	:800	1				
	so	M1					29200			0				
36.0	eo	M1												
55.0	qo	A1										A1	:3496	1
	so	A1									46504			0
74.0	qo	A2										A2	:1792	1
90.0	eo	A1												
90.9	so	A2									44712			0
106.4	qo	A3										A3	:3144	1
108.9	eo	A2												
109.8	so	A3									41568			0
135.1	qo	M2						M2	:4056	1				
141.3	eo	A3												
141.9	qo	M3								2				
142.1	qo	V1		V1	:12208	1								
142.3	so	V1	7792			0								
158.5	qo	A4										A4	:5120	1
187.6	qo	A5												2
223.3	qo	M4								3				
259.0	qo	A6												3
264.4	eo	V1												
265.4	so	M2					25144	M3	:576	2				
278.4	qo	A7												4
300.5	qo	M5								3				
306.0	eo	M2												
307.0	so	M3					24568	M4	:3304	2				
312.8	eo	M3												
313.8	so	M4					21264	M5	:2760	1				
319.1	qo	M6								2				
325.8	qo	M7								3				
346.9	eo	M4												
347.9	so	M5					18504	M6	:1752	2				
349.8	qo	A8												5
375.5	eo	M5												
376.5	so	M6					16752	M7	:576	1				
394.1	eo	M6												
395.0	so	M7					16176			0				
398.4	qo	A9												6
400.8	eo	M7												

			EF Queue				AF Queue				BE Queue			
Tick	Ev	Pkid	Credit	HdPk	:Frlen	Qn	Credit	HdPk	:Frlen	Qn	Credit	HdPk	:Frlen	Qn
401.7	so	A4									36448	A5	:2808	5
413.9	qo	M8						M8	:2632	1				
448.7	qo	A10												6
452.9	eo	A4												
453.9	so	M8					13544			0				
480.3	eo	M8												
481.3	so	A5									33640	A6	:6488	5
499.0	qo	A11												6
503.8	qo	M9						M9	:3048	1				
509.4	eo	A5												
510.4	so	M9					10496			0				
540.9	eo	M9												
541.9	so	A6									27152	A7	:1840	5
559.7	qo	M10						M10	:2568	1				
570.6	qo	A12												6
592.8	qo	M11								2				
606.8	eo	A6												
607.8	so	M10					7928	M11	:3184	1				
619.0	qo	M12								2				
628.3	qo	A13												7
633.5	eo	M10												
634.5	so	M11					4744	M12	:1240	1				
637.2	qo	M13								2				
666.4	eo	M11												
667.3	so	M12					3504	M13	:1720	1				
675.1	qo	A14												8
679.7	eo	M12												
680.6	so	M13					1784			0				
697.8	eo	M13												
698.8	so	A7									25312	A8	:7040	7
711.9	qo	M14						M14	:3880	1				
717.2	eo	A7												
718.2	so	A8									18272	A9	:4752	6
732.0	qo	A15												7
742.1	qo	V2		V2	:12208	1								
751.2	qo	M15								2				
786.4	qo	A16												8
788.6	eo	A8												
789.5	so	A9									13520	A10	:4080	7
817.6	qo	A17												8
820.0	qo	M16								3				
824.3	qo	A18												9
837.1	eo	A9												
838.1	so	A10									9440	A11	:4928	8
878.9	eo	A10												
879.9	so	A11									4512	A12	:7056	7
907.6	qo	A19												8
929.2	eo	A11												
930.2	so	V2	-4416			0								
940.2	qo	M17								4				
979.0	qo	A20												9
986.5	qo	M18								5				
1000.0	q+		20000				30000				50000			
1001.4	qo	A21												10
1023.2	qo	A22												11
1052.3	eo	V2												
1053.2	so	M14					26120	M15	:3824	4				
1077.7	qo	A23												12
1087.1	qo	M19								5				
1092.0	eo	M14												
1093.0	so	M15					22296	M16	:3408	4				
1131.3	eo	M15												
1131.9	qo	A24												13
1132.2	so	M16					18888	M17	:2016	3				
1142.1	qo	M20								4				
1147.3	qo	A25												14
1166.3	eo	M16												
1167.3	so	M17					16872	M18	:576	3				
1184.8	qo	M21								4				
1187.5	eo	M17												
1188.5	so	M18					16296	M19	:1680	3				
1194.3	eo	M18												
1195.3	so	M19					14616	M20	:4080	2				
1212.1	eo	M19												
1213.1	so	M20					10536	M21	:3896	1				
1216.0	qo	M22								2				
1219.0	qo	A26												15
1225.8	qo	M23								3				

When V2 is placed in the EF queue, A8 is being serialized.

When the port is free, neither V2 nor M14 are selected because there is not enough credit for them while there is enough credit for A9, and even subsequently for A10 and A11. In fact BE traffic takes its share of bandwidth.

Here is an example of a second scanning of the queues by the scheduler: No packet (at the head of each queue) has enough credit. As a result, V2 takes advantage of EF priority over other classes and is serialized. But V2 has waited 188.1 microseconds in queue, thus experiencing jitter. This is normal considering that the IP throughput is 20Mbps and therefore the actual throughput a little higher than the assigned bandwidth because of the frame overhead.

Note that at the "1000.0" millisecond boundary, credits were refreshed.

V3 has enough credit, because of the previous refresh, and is serialized as soon as the port is free.

One can observe that, when there is no pending EF packets, AF packets are serialized because they have enough credit and a higher priority than BE packets.

When AF credit is no more sufficient, BE packets are then serialized, thus using BE credit.

Here is another example of the selection of a packet although its credit becomes negative. This happens actually close to the refresh time, and is due to the contingency of offered load and packet sizes.

Tick	Ev	Pkid	EF Queue Credit	EF HdPk	EF Frlen	EF Qn	AF Queue Credit	AF HdPk	AF Frlen	AF Qn	BE Queue Credit	BE HdPk	BE Frlen	BE Qn
1238.8	qo	A27												16
1253.9	eo	M20												
1254.9	so	M21					6640	M22	:1000	2				
1293.9	eo	M21												
1294.9	so	M22					5640	M23	:576	1				
1304.1	qo	A28												17
1304.9	eo	M22												
1305.8	so	M23					5064			0				
1311.6	eo	M23												
1312.5	so	A12									42944	A13	:5192	16
1324.7	qo	M24						M24	:1904	1				
1342.1	qo	V3		V3	:12208	1								
1348.3	qo	A29												17
1373.1	qo	M25								2				
1383.1	eo	A12												
1384.0	so	V3	7792			0								
1389.1	qo	M26								3				
1419.0	qo	A30												18
1459.0	qo	A31												19
1469.0	qo	A32												20
1491.4	qo	M27								4				
1506.1	eo	V3												
1507.1	so	M24					3160	M25	:4040	3				
1519.0	qo	M28								4				
1525.8	qo	M29								5				
1526.2	eo	M24												
1527.2	so	A13									37752	A14	:4584	19
1546.4	qo	M30								6				
1548.1	qo	A33												20
1563.8	qo	A34												21
1579.2	eo	A13												
1580.1	so	A14									33168	A15	:5600	20
1582.7	qo	A35												21
1616.0	qo	M31								7				
1626.0	eo	A14												
1627.0	so	A15									27568	A16	:4632	20
1653.2	qo	M32								8				
1654.7	qo	A36												21
1683.0	eo	A15												
1684.0	so	A16									22936	A17	:3024	20
1699.0	qo	A37												21
1705.3	qo	M33								9				
1730.4	eo	A16												
1731.4	so	A17									19912	A18	:576	20
1742.0	qo	M34								10				
1746.5	qo	A38												21
1760.7	qo	M35								11				
1761.7	eo	A17												
1762.7	so	A18									19336	A19	:7368	20
1768.5	eo	A18												
1769.5	so	A19									11968	A20	:7040	19
1819.1	qo	M36								12				
1823.5	qo	A39												20
1843.2	eo	A19												
1844.2	so	A20									4928	A21	:2136	19
1858.0	qo	M37								13				
1868.7	qo	M38								14				
1879.7	qo	A40												20
1914.6	eo	A20												
1915.6	so	A21									2792	A22	:2072	19
1937.0	eo	A21												
1938.0	so	A22									720	A23	:4808	18
1939.1	qo	A41												19
1942.1	qo	V4		V4	:12208	1								
1942.7	qo	M39								15				
1958.8	eo	A22												
1959.8	so	V4	-4416			0								
1986.5	qo	M40								16				
1990.3	qo	A42												20
1997.0	qo	A43												21
2000.0	q+		20000				30000				50000			
...														

List of abbreviations

AF	Assured Forwarding
ATM	Asynchronous Transfer Mode
BA	Behavior Aggregate
BE	Best Effort
CE	Customer Edge equipment
CoS	Class of Service
CPE	Customer Premises Equipment
CRC	Cyclic Redundancy Check
CT	Class-Type
DS	Differentiated Services (DiffServ)
DSCP	DS Code Point
DS-TE	DS Traffic Engineering
EF	Expedited Forwarding
E-LSP	EXP-Inferred-PSC LSP
FCS	Frame Check Sequence
FE	Fast Ethernet
FEC	Forwarding Equivalence Class
FR	Frame Relay
GE	Gigabit Ethernet
L-LSP	Label-Only-Inferred-PSC LSP
LSP	Label Switched Path
LSR	Label Switching Router
Mbps	Megabits per second
MPLS	Multi Protocol Label Switching
MTU	Maximum Transfer Unit
OA	Ordered Aggregate
P	Provider core equipment
PDB	Per-Domain Behavior
PDH	Plesiochronous Digital Hierarchy (E1, E3, DS3)
PE	Provider Edge equipment
PHB	Per Hop Behavior
PPP	Point-to-Point Protocol
PSC	PHB Scheduling Class
QoS	Quality of Service
RED	Random Early Detection
RFC	Request for Comments
RTO	Retransmission Time-Out
RTT	Round Trip Time
SDH	Synchronous Digital Hierarchy (STM-1, STM-4, STM-16...)
SFD	Start Frame Delimiter
SLS	Service Level Specification
TCP	Transmission Control Protocol
TE	Traffic Engineering
TOS	Type Of Service field
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VPN	Virtual Private Network

References

White Papers

[1] QoS Support in MPLS Networks (Victoria Fineberg, MPLS/Frame Relay Alliance)

[2] Supporting differentiated service classes in Large IP Networks (Chuck Semeria, Juniper)

[3] Supporting differentiated service classes: Queue Scheduling Disciplines (Chuck Semeria, Juniper)

[4] Supporting differentiated service classes: Active Queue Memory Management (Chuck Semeria, Juniper)

[5] Supporting differentiated service classes: TCP Congestion Control Mechanisms (Chuck Semeria, Juniper)

[6] Supporting differentiated service classes: MPLS (Chuck Semeria, Juniper)

[7] DiffServ – The Scalable End-to-End QoS Model (Cisco)

[8] Advanced Topics in MPLS-TE Deployment (Cisco)

[9] Virtual Leased Line Services Using Cisco MPLS DiffServ-Aware Traffic Engineering (Cisco)

[10] Voice Trunking and Toll-Bypass Trunking Using Cisco MPLS DiffServ-Aware Traffic Engineering (Cisco)

URLs

[11] NS-2 Network Simulator – http://www.isi.edu/nsnam/ns/

RFCs

[12] RFC 2018 – TCP selective Acknowledgment Options (Mathis, et al.)

[13] RFC 2309 – Recommendations on Queue Management and Congestion Avoidance in the Internet (Braden, et al.)

[14] RFC 2474 – Definition of the DS Field (Nichols, et al.)

[15] RFC 2475 – An Architecture for Differentiated Services (Blake, et al.)

[16] RFC 2581 – TCP Congestion Control (Allman, et al.)

[17] RFC 2582 – NewReno modification to TCP's Fast Recovery algorithm (Floyd & Henderson)

[18] RFC 2597 – Assured Forwarding PHB Group (Heinanen)

[19] RFC 2697 – A Single Rate Three Color Marker (Heinanen & Guerin)

[20] RFC 2698 – A Two Rate Three Color Marker (Heinanen & Guerin)

[21] RFC 2702 – Requirements for Traffic Engineering over MPLS (Awduche, et al.)

[22] RFC 3031 – MPLS Architecture (Rosen, et al.)

[23] RFC 3086 – Definition of Differentiated Services Per Domain Behaviors (Nichols & Carpenter)

[24] RFC 3246 – An Expedited Forwarding PHB (Davie, et al.)

[25] RFC 3260 – New Terminology and Clarifications for Diffserv (Grossman)

[26] RFC 3270 – MPLS Support of Differentiated Services (Le Faucheur, et al.)

[27] RFC 3272 – Overview and Principles of Internet Traffic Engineering (Awduche, et al.)

[28] RFC 3290 – An Informal Management Model for Diffserv Routers (Bernet, et al.)

[29] RFC 3564 – Requirements for Support of Diffserv Aware MPLS Traffic Engineering (Le Faucheur & Lai)

Books

[30] "TCP/IP Illustrated, Volume 1: The Protocols" W. Richard Stevens

Glossary

Frame – A frame is the unit of transmission in a link layer protocol, and consists of a link-layer header followed by a packet.

IP Datagram – An IP datagram is the unit of end-to-end transmission in the IP protocol. An IP datagram consists of an IP header followed by transport layer data.

Packet – A packet is the unit of data passed across the interface between the internet layer and the link layer. It includes an IP header and data. A packet may be a complete IP datagram or a fragment of an IP datagram.

Segment – a segment is the unit of end-to-end transmission in the TCP protocol. A segment consists of a TCP header followed by application data. A segment is transmitted by encapsulation inside an IP datagram.

Traffic Trunk (TT) – an aggregation of traffic flows of the same class which are placed inside an LSP.

DS Field – the 6 most significant bits of the (former) IPV4 TOS octet or the (former) IPV6 Traffic Class octet.

DS code point (DSCP) – a value which is encoded in the DS field, and which each DS Node MUST use to select the PHB which is to be experienced by each packet it forwards.

Microflow – a single instance of an application-to-application flow of packets which is identified by source address, destination address, protocol id, and source port, destination port (where applicable).

Behavior Aggregate (BA) – a collection of packets with the same DS code point crossing a link in a particular direction.

Per Hop Behavior (PHB) – the externally observable forwarding behavior applied at a DS-compliant node to a behavior aggregate.

Traffic Aggregate (TA) – a collection of packets with a codepoint that maps to the same PHB, usually in a DS domain or some subset of a DS domain. A traffic aggregate marked for the foo PHB is referred to as the "foo traffic aggregate" or "foo aggregate" interchangeably. This generalizes the concept of Behavior Aggregate from a link to a network.

Per-Domain Behavior (PDB) – the expected treatment that an identifiable or target group of packets will receive from "edge-to-edge" of a DS domain. A particular PHB (or, if applicable, list of PHBs) and traffic conditioning requirements are associated with each PDB.

Ordered Aggregate (OA) – a set of Behavior Aggregates that share an ordering constraint. The set of PHBs that are applied to this set of Behavior Aggregates constitutes a PHB scheduling class.

PHB Group – a set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy.

PHB Scheduling Class (PSC) – a PHB group for which a common constraint is that, ordering of at least those packets belonging to the same microflow must be preserved.

DS Domain – a contiguous portion of the Internet over which a consistent set of differentiated services policies are administered in a coordinated fashion. A DS domain can represent different administrative domains or autonomous systems, different trust regions, different network technologies (e.g., cell/frame), hosts and routers, etc.

Service Level Specification (SLS) – a set of parameters and their values which together define the service offered to a traffic stream by a DS domain.

Traffic Conditioning Specification (TCS) – a set of parameters and their values which together specify a set of classifier rules and a traffic profile; a TCS is an integral element of an SLS.

BA classifier – a classifier that selects packets based only on the contents of the DS field.

MF Classifier – a multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.

Traffic conditioner – an entity which performs traffic conditioning functions and which may contain meters, markers, droppers, and shapers. Traffic conditioners are typically deployed in DS boundary nodes only. A traffic conditioner may re-mark a traffic stream or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile.

Traffic profile – a description of the temporal properties of a traffic stream such as rate and burst size.

Metering – the process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, or dropper, and/or may be used for accounting and measurement purposes.

Marking – the process of setting the DS codepoint in a packet based on defined rules; pre- marking, re-marking.

Shaping – the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.

Policing – the process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.

Assured Forwarding (AF) – PHB group which is a means for a provider DS domain to offer different levels of forwarding assurances. Within each AF class (an instance of the AF PHB group) IP packets are marked with one of three possible drop precedence values.

Expedited Forwarding (EF) – PHB intended to provide a building block for low delay, low jitter and low loss services by ensuring that the EF aggregate is served at a certain configured rate.

Class-Type (CT) – The set of traffic trunks crossing a link, that is governed by a specific set of bandwidth constraints. CT is used for the purposes of link bandwidth allocation, constraint based routing and admission control. A given traffic trunk belongs to the same CT on all links.