International Telecommunication Union

ITU-T G.1050

(07/2016)

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Multimedia Quality of Service and performance – Generic and user-related aspects

Network model for evaluating multimedia transmission performance over Internet Protocol

Recommendation ITU-T G.1050

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

International telephone connections and circuits	G.100–G.199
General characteristics common to all analogue carrier-transmission systems	G.200–G.299
Individual characteristics of international carrier telephone systems on metallic lines	G.300–G.399
General characteristics of international carrier telephone systems on radio-relay or satellite links and interconnection with metallic lines	G.400–G.449
Coordination of radiotelephony and line telephony	G.450–G.499
Transmission media and optical systems characteristics	G.600–G.699
Digital terminal equipments	G.700–G.799
Digital networks	G.800–G.899
Digital sections and digital line system	G.900–G.999
Multimedia quality of service and performance – generic and user-related aspects	**G.1000–G.1999**
Transmission media characteristics	G.6000–G.6999
Data over transport – generic aspects	G.7000–G.7999
Packet over transport aspects	G.8000–G.8999
Access networks	G.9000–G.9999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T G.1050

Network model for evaluating multimedia transmission performance over Internet Protocol

Summary

Recommendation ITU-T G.1050 describes an Internet protocol (IP) network model that can be used for evaluating the performance of IP streams. The focus is on packet delay, delay variation, and loss. IP streams from any type of network device can be evaluated using this model.

The following are possible uses for Recommendation ITU-T G.1050:

– simulation of real-world IP network impairments (packet delay variation and packet loss characteristics);

– testing of any type of IP stream(s) under simulated network conditions using pcap files. The IP stream(s) can be evaluated using standard test cases or user-defined simulated network conditions;

– testing of any type of IP stream using hardware emulation of simulated network models using standard test cases or user-defined simulated network conditions.

This revision of Recommendation ITU-T G.1050 replaces Recommendation ITU-T G.1050 (2011) in its entirety.

Technical changes from Recommendation ITU-T G.1050 (2011) include:

1) The previous version of this Recommendation, ITU-T G.1050 (2011), uses pre-recorded pcap files of unmanaged best-effort streams (e.g., HTTP, P2P) that do not react to network congestion and therefore do not adapt to the available network capacity as would be expected in an actual network. The revised version of this Recommendation replaces these pcap files with an iPerf transmission control protocol (TCP) stream. Because the iPerf stream is carried by TCP, it automatically adapts to the simulated network conditions. In addition to this, the iPerf stream accurately measures the residual capacity of the network while carrying other managed services. Therefore, this enhancement provides more realism than the previous version.

2) The previous version of this Recommendation uses a custom discrete event simulator written in C++. The revised version of this Recommendation implements the same network topologies and test cases using the publicly available ns3 discrete event simulator. The new approach uses a direct code execution (DCE) virtual environment to run an actual iPerf application with an actual Linux network stack in each network node that uses TCP, making the results more realistic. For the simulations in this Recommendation, the Linux network stacks are configured to use the "cubic" TCP congestion control algorithms.

This Recommendation includes an electronic attachment containing the discrete event simulator source code, input packet capture files of interfering traffic, standard test cases and the simulator output.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T G.1050	2005-11-29	12	11.1002/1000/8674
2.0	ITU-T G.1050	2007-11-13	12	11.1002/1000/9272
3.0	ITU-T G.1050	2011-03-01	12	11.1002/1000/11074
4.0	ITU-T G.1050	2016-07-29	12	11.1002/1000/12968

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser,followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Electronic attachment: Discrete event simulator source code, input packet capture files of interfering traffic, standard test cases and the simulator output.

Introduction

Recommendation ITU-T G.1050 describes an IP network model that can be used for evaluating the performance of IP streams. The focus is on packet delay, delay variation, and loss. IP streams from any type of network device can be evaluated using this model.

Emphasis is given to the fact that manufacturers of communications equipment and service providers are interested in a specification that accurately models the IP network characteristics that determine performance. Evaluators desire a definitive set of simple tests that properly measure the performance of communications devices from various manufacturers. Therefore, the objective of this Recommendation is to define an application-independent model (e.g., data, voice, voiceband data, and video) that is representative of IP networks, that can be simulated at reasonable complexity, and that facilitates practical evaluation times. The IP network model presented herein represents a snapshot of actual network data provided by anonymous IP service providers and IP network equipment manufacturers in the 2010 timeframe, and will continue to evolve as more statistical information becomes available and as the IP network evolves.

Recommendation ITU-T G.1050

Network model for evaluating multimedia transmission performance over Internet Protocol

1 Scope

This revision to the Recommendation[1] defines managed Internet Protocol television (using UDP) (IPTV), IPTV and VoIP stream for well managed, partially managed and un-managed network conditions. It also defines the remaining residual unmanaged bandwidth (Internet services) which are typically used for TCP applications. The main difference between this revision and the previous revision is that this version is a Layer 4 (TCP) aware network model that can be used to evaluate TCP traffic.

This revision of the Recommendation utilizes the publicly available ns-3 network simulator which incorporates layer 4 TCP models. It allows more accurate characterization of bandwidth and delay for networks that carry TCP traffic because congestion causes TCP streams to back off and use less network bandwidth. The model is limited in that it is based on a single (Cubic) TCP flow. Different TCP stacks have different behaviours. However, the macroscopic behaviour of TCP flows should be similar.

This Recommendation is broadly applicable to the evaluation of any equipment that terminates or routes traffic using the Internet Protocol. This Recommendation can also be used to evaluate media streams or other protocols carried over IP networks. Examples of the types of equipment that can be evaluated using this model include:

– IP-connected endpoints:
  - IP network devices (such as: user agents, call agents, media servers, media gateways, application servers, routers, switches, etc.);
  - IP video (IPTV, video conferencing, telepresence, etc.);
  - IP phones (including soft phones);
  - IAF (Internet-aware fax).
– IP/TCP connected endpoints:
  - Peer-to-peer;
  - hypertext transfer protocol (HTTP);
  - Adaptive bit-rate video.
– PSTN-connected devices through IP gateways:
  - Plain old telephone service (POTS) through voice-over-IP (VoIP) gateways;
  - ITU-T T.38 facsimile devices and gateways;
  - ITU-T V.150.1 and ITU-T V.152 (voiceband data, VBD) modem-over-IP gateways;
  - TIA-1001 and ITU-T V.151 textphone-over-IP gateways.

The IP network model can be used in two ways:

– to test an IP stream under simulated network conditions;
– to test an IP stream in real time using hardware emulation of the network model.

---

[1] This Recommendation includes an electronic attachment containing the discrete event simulator source code, input packet capture files of interfering traffic, standard test cases and the simulator output.

The IP network model can be used to study and to understand:

– the interaction of different traffic mixes;

– the effects of QoS and queuing on different types of traffic;

– packet delay variation and packet loss.

Whether in software simulation or real-time hardware emulation, users can select from several test cases specified in this Recommendation. Users can optionally define their own test cases.

This model has the following limitations:

– Some VoIP networks may utilize public switched telephone network (PSTN) at one or both ends of the connection through a media gateway. This model only addresses the IP portion of the network and does not address the PSTN portion of the end-to-end connection.

– The network model represented in this Recommendation does not model all possible connections that can be encountered between devices.

– This Recommendation includes gigabit-capable passive optical network (GPON) and digital subscriber line (DSL) access technologies. Characteristics of other access technologies such as cable television (CATV) and wireless are for further study.

– Abnormal events such as link failures and route flaps (and the packet reordering that such events can cause) are not included in this Recommendation.

– The standard test cases use streams of interfering traffic that were captured on live networks. While realistic, they are still just examples; users could substitute their own files of interfering traffic.

– The LAN-to-LAN test cases of ITU-T G.1050 are now modelled as two cascaded ITU-T G.1050 core-to-LAN segments. See clause 6.3.

– The IP network model presented herein is based on an informal survey of anonymous IP service providers and IP network equipment manufacturers in the 2010 timeframe and will continue to evolve as more statistical information becomes available and as the IP network evolves.

The most significant limitation of the previous edition of the model was that the best effort traffic (e.g., HTTP) is injected from a pcap file and does not react to congestion (as TCP normally would). As a result, the packet loss and delay statistics were somewhat higher than expected.

To address this limitation, the best effort traffic that was previously injected from a pcap file has been replaced by a single TCP flow using iPerf. The specific characteristics of the iPerf flow depend on the specific TCP congestion control algorithm, but generally TCP will tend to use as much network capacity as is available to best effort traffic. Therefore, the iPerf TCP flow characterizes the available network capacity and delay while competing with other managed flows.

In order to implement the iPerf TCP flows, the simulator described in this revised Recommendation has been completely rewritten to use the ns3 discrete event simulator.

The ns3 simulator has a direct code execution (DCE) virtual environment in which it is possible to run an actual iPerf application with an actual Linux network stack in each TCP network node. This makes the results more realistic. For the simulations in this Recommendation, the Linux network stacks are configured to use the "cubic" TCP congestion control algorithms.

The network topologies and test cases are the same as was implemented as in the previous revision of the simulator, with the exception that all best effort traffic injected from pcap files has been removed and replaced by a single iPerf TCP flow.

It should be noted that although most networks of interest carry multiple concurrent TCP flows, the aggregate goodput of the flows tends to converge to the same value as a single iPerf flow would. Therefore, it is not necessary to characterize more than one parallel iPerf TCP flow in this simulator.

Finally, as a result of interactions between downstream and upstream iPerf TCP flows, each test case is run three ways:

– iPerf downstream only;

– iPerf upstream only;

– iPerf bidi (runs downstream and upstream together, which interfere with each other).

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T T.38] Recommendation ITU-T T.38 (2007), *Procedures for real-time Group 3 facsimile communication over IP networks*.

[ITU-T V.150.1] Recommendation ITU-T V.150.1 (2003), *Modem-over-IP networks: Procedures for the end-to-end connection of V-series DCEs*.

[ITU-T V.152] Recommendation ITU-T V.152 (2010), *Procedures for supporting voice-band data over IP networks*.

[ITU-T Y.1541] Recommendation ITU-T Y.1541 (2011), *Network performance objectives for IP-based services*.

3 Definitions

This Recommendation defines the following terms:

3.1 burst loss: A high density of packet loss over time, or loss of sequential packets, due to congestion, bandwidth limitation, line errors, or rerouting (delay translated into loss due to implementation) on the network.

3.2 delay: The time required for a packet to traverse the network or a segment of the network (see latency).

3.3 downstream: A transmission from a service provider toward an end user.

3.4 gateway: A network device that acts as an entrance to another network. One function is to convert media provided in one type of network to the format required in another type of network. For example, a gateway could terminate bearer channels from a switched circuit network (e.g., DS0s) and media streams from a packet network (e.g., RTP streams in an IP network).

3.5 goodput: Application level throughput.

3.6 interferer: A packet stream that contends with the test stream of interest for a limited network resource, such as a link buffer.

3.7 IP network: A network based on the Internet Protocol.

3.8 jitter: Variation in packet delay.

3.9 latency: An expression of how much time it takes for a packet of data to get from one designated point to another (see delay).

3.10 microburst: A packet traffic pattern characterized by short periods of high activity, and where the bursts are not readily detectable when measuring average traffic rate over a period of one second or longer.

3.11 MTU size: The largest size data-link packet or frame (specified in octets) that can be sent in a packet- or frame-based network such as the Internet.

3.12 packet loss: The failure of a packet to traverse the network to its destination. Typically, packet loss is caused by packet discards due to buffer overflow. This model does not take into account packet loss due to discards in the terminal jitter buffer.

3.13 peak jitter: The maximum variation of delay from the mean delay.

3.14 peak-to-peak jitter: The full range of packet delay from the maximum amount to the minimum amount.

3.15 peer-to-peer: A distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application.

3.16 QoS edge routing: Routing that typically takes place between the customer premises network and the service provider network based on quality of service classification values.

3.17 reordered packets: A packet that arrives at the destination with a packet sequence number that is smaller than the previous packet is deemed a reordered packet.

3.18 route flap: Repeated changes in a path due to updates to a routing table. The network model simulates the effect of route flaps by making incremental changes in the delay values of the core segment.

3.19 sequential packet loss: Two or more consecutive lost packets.

3.20 upstream: A transmission from an end user toward a service provider.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

BER	Bit Error Ratio
CATV	Cable Television
CBR	Constant Bit Rate
CSV	Comma-Separated Values (file format)
DCE	Direct Code Execution
DSL	Digital Subscriber Line
DSLAM	DSL Access Multiplexer
GPON	Gigabit-capable Passive Optical Network
HD	High-Definition video
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IPTV	Internet Protocol Television (using UDP)
LAN	Local Area Network
MTU	Maximum Transmission Unit
OLT	Optical Line Termination

ONT	Optical Network Termination
OTT	Over-the-Top (TCP streaming video)
PBS	Peak Burst Size (pcap generator)
pcap	packet capture (file format)
PIR	Peak Information Rate (pcap generator)
POTS	Plain Old Telephone Service
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RTT	Round-Trip-Times
SD	Standard-Definition video
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VoIP	Voice over Internet Protocol

5 Conventions

This Recommendation uses capitalized nouns when referring to objects in the C++ simulator source code of the packet scheduling algorithm.

6 Description of the model

6.1 Model overview

The new IP network model of this Recommendation is embodied in a discrete event software simulator. In a real sense, the simulator is the model. Annex A contains a detailed description of the simulator code used to implement the model. Other implementations are possible, including real-time hardware network emulators for test lab use, but their behaviour must match that of the simulator presented here.

The IP network is modelled as a network of basic elements. Figure 1 shows this basic network element, called a "node."

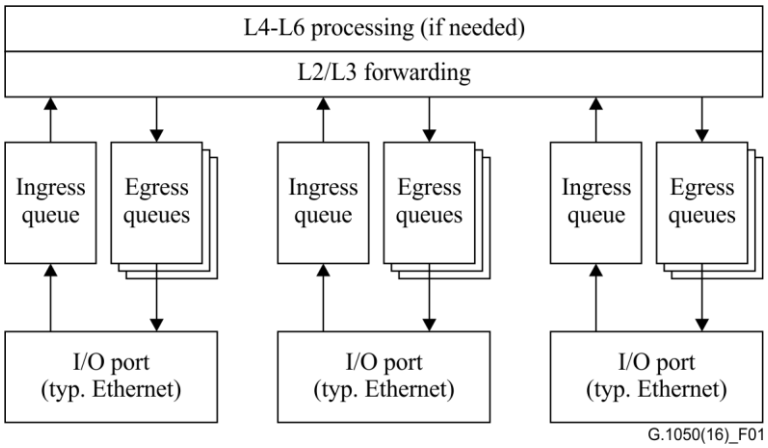


Figure 1 – Node – Basic model network element

These basic network elements are wired in series into a specific network topology as described in clause 6.2.

This is an outline of the simulator processing in one direction; both directions are included in the model:

1) A packet generator drives packets into the simulator. The arrival times and sizes of the test stream packets and the interfering stream packets are read from pcap files.
2) A node receives packets on its ingress ports, and determines where packets should go next.
3) A node schedules each packet for transmission out one of its egress ports.
4) Wires connect the egress port of one node to the ingress port of another node.
5) The process repeats for all packets through all nodes and wires.
6) Packet arrival and departure times are stored in a file for analysis.

The clauses that follow explain the components of the model in more detail:

– network topology;
– models of network elements;
– interfering stream files;
– simulation inputs;
– simulation outputs;
– packet scheduling algorithm.

6.2 Network topology

The following is a high-level description of the elements of the model. A more detailed description is available in Annex A.

Figure 2 shows the IP service provider's portion of the network, called the core, represented by a cloud symbol. Basic network elements within the core are interconnected to carry IP traffic between ports.

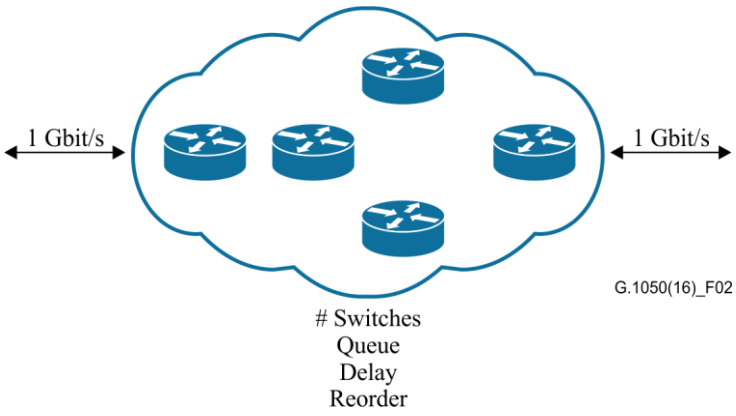


Figure 2 – Core network portion

Figure 3 shows the access portion of the IP network. In the downstream direction, from the core to the customer premises, a series of nodes and wires are connected: edge router, DSLAM (or OLT for GPON), DSL modem (or ONT for GPON), firewall, and router. The model is bidirectional, so upstream traffic traverses the same elements in reverse order.

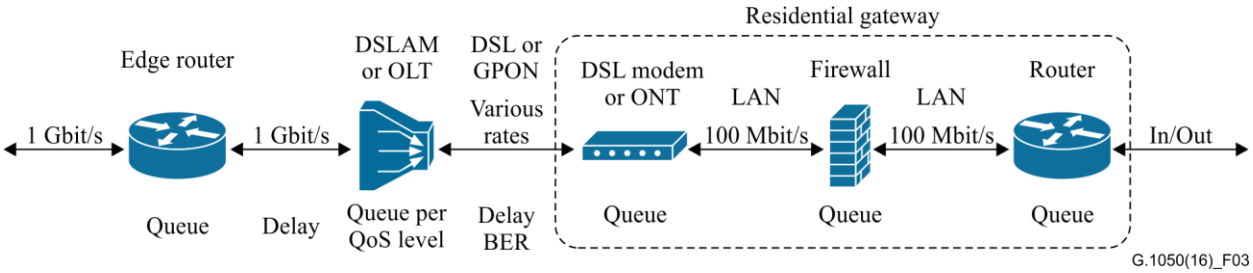


Figure 3 – Access network portion

Although the basic node of Figure 1 allows interfering streams to be inserted and to exit at each stage, the network topologies of this Recommendation are simplified. Interferers in the models only enter and exit at the points shown. All of the test case simulations of this Recommendation use the core-to-LAN network model.

Figure 4 (core-to-LAN) illustrates traffic flow for the managed server to client applications such as IPTV or streaming server and unmanaged TCP client and server traffic using iPerf.

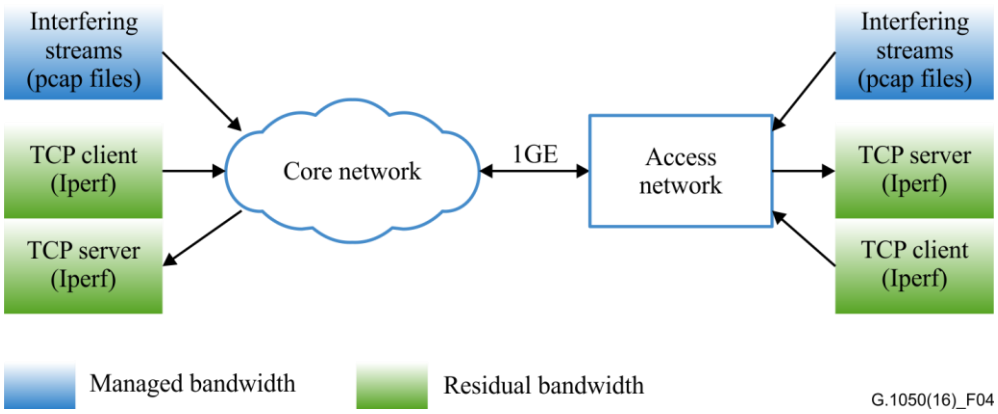


Figure 4 – Core-to-LAN network model

Figure 5 (LAN-to-LAN) illustrates an end-to-end network with LAN and access links on each side of the core as would occur in a client-to-client application such as voice over Internet Protocol (VoIP). It includes traffic flow for the managed applications or streaming server and unmanaged TCP client and server traffic using IPerf. It is modelled as two core-to-LAN networks concatenated. Some considerations for using the LAN-to-LAN model are given in clause 8.3.

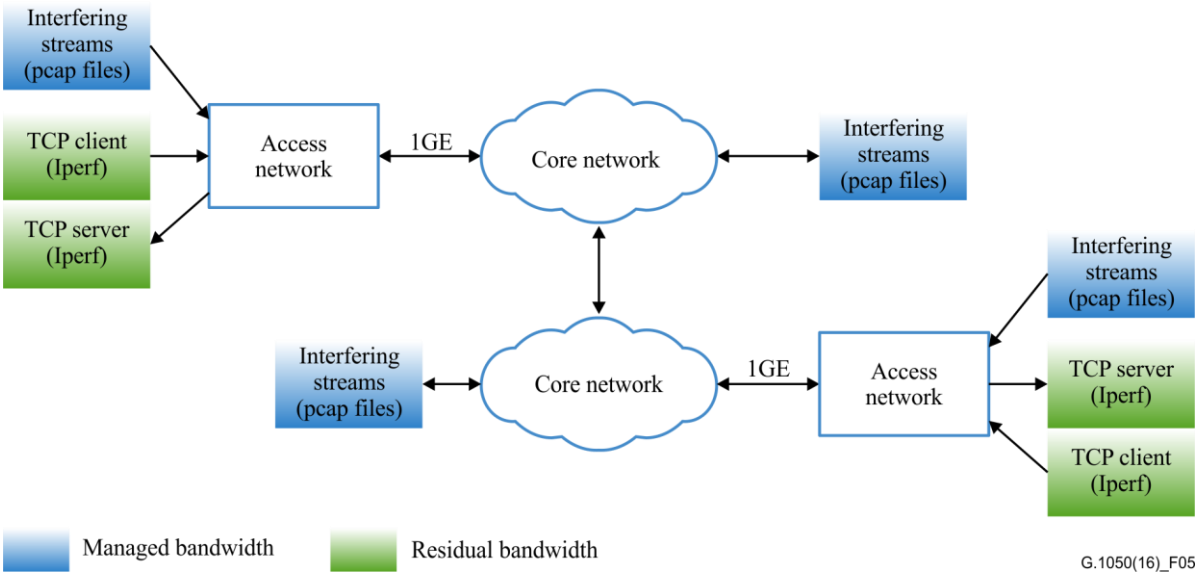


Figure 5 – LAN-to-LAN network model

6.3 Models of network elements

The various types of network elements considered in this model are listed as columns in Table 1. These are:

– core switches;
– edge router;
– access head-end device:
  • DSLAM;
  • GPON OLT.
– access subscriber end device:
  • DSL modem;
  • GPON ONT.
– firewall (as part of a residential gateway, for example);
– LAN;
– wires between devices.

Each of the network elements is modelled identically in the simulator: as the basic node element shown in Figure 1. All egress ports have multiple queues, one per quality of service (QoS) priority. Each queue is 65 × 1518 = 98'670 bytes per priority level.

The rows in Table 1 list the following attributes of each element in the network model:

– # switches: for the core section only, there are between 3 and 15 cascaded gigabit Ethernet switches;
– downstream link: refers to the direction from the core toward the premises;
– upstream link: refers to the direction from the premises toward the core;

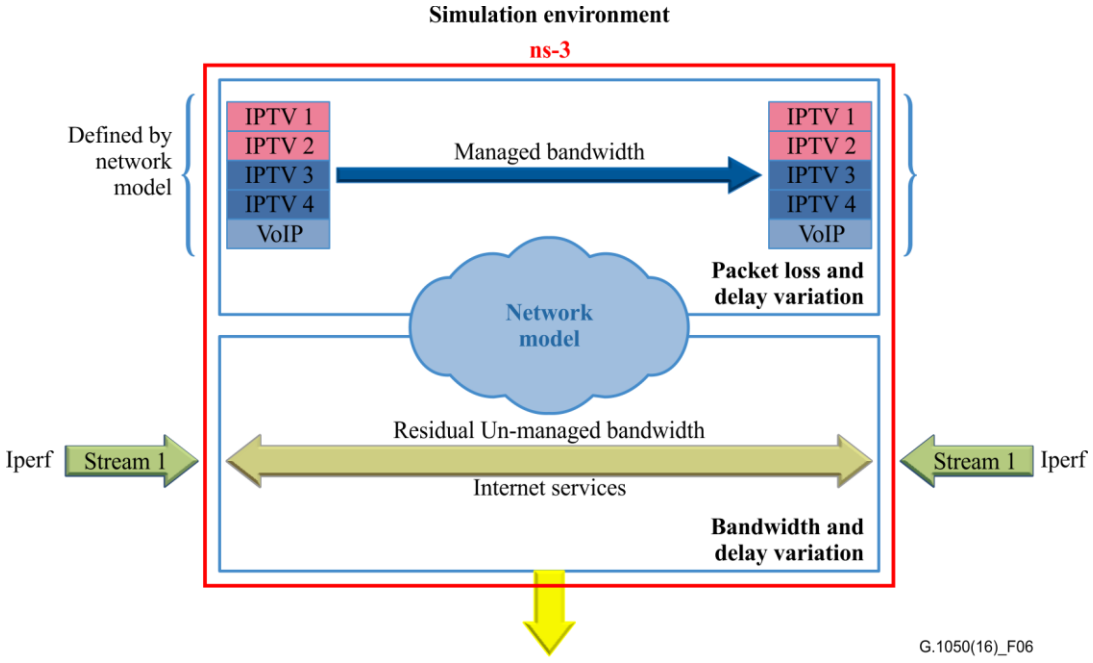
– delay: the one-way flat delay of the element;
– QoS: indicates that the element implements QoS priority scheduling;
– BER: the bit error ratio (BER) of the physical access link.

Table 1 – Network model element attributes

Attribute	Core Switches	Wire	Edge router	Wire	DSLAM	DSL access	DSL modem	GPON OLT	GPON access	GPON ONT	Wire	Firewall	LAN
# switches	3 to 15												
Downstream link rate (Mbit/s)		1000		1000		3 to 33			5 to 50		100		100
Upstream link rate (Mbit/s)		1000		1000		1 to 3			2 to 35		100		100
Delay		10 to 300 ms		100 ns		1 ms			1 ms				
QoS	1 to 7		1 to 7		1 to 7		1 to 7	1 to 7		1 to 7		1 to 7	
BER						10^{-8} to 10^{-6}			10^{-12} to 10^{-9}				

6.4 Interfering stream files

The network traffic files used by the simulator are all of a standard packet capture format commonly known as pcap, as defined in file pcap.h (version 2.4) of libpcap. See http://wiki.wireshark.org/Development/LibpcapFileFormat. Usually the files have been captured at the endpoints of the network at client devices. These files are used as interfering traffic input to the simulation model, as shown in Figure 6.



Annex F Table 2: Network characteristic files for residual Un-managed bandwidth (bandwidth and delay)

Figure 6 – Revised network model

The following types of streams are included in the standard test cases:

– IPTV: SD and HD, CBR and VBR;

– VoIP;

– FoIP (fax over IP).

These files have been anonymously captured by a variety of researchers and from a variety of sources as real examples of representative types of packet traffic. In general, the files are a reflection of the traffic patterns observed. They have had the payloads stripped to protect the privacy of the end user and ownership rights of the actual content as well as to keep the overall file size small. Note that even though the traffic source files omit the payloads, size information is maintained so that the full traffic is modelled by the simulation. Pcap files with payloads can be used as test streams or interfering streams, since the simulator does pass payload data.

In addition to payload stripping, the pcap files have had a number of processing steps:

– Flow separation: The captured files are split into upstream (toward the core) and downstream (toward the client endpoint) files for simulation.

– Bandwidth analysis: The files are analysed as to their overall and average bandwidth over finite time periods. As reference files, the microscopic and macroscopic bandwidth characteristics must be understood so that they can be applied reasonably to the various simulation test cases. In particular, the bandwidth characteristics of the files are highly dependent on factors such as the overall bandwidth capabilities and rate limitations of the network path, and the behaviour and rate adaptation of higher layer network protocols.

– Smoothing: At a microscopic level, bandwidth analysis of some pcap files reveals that there are microbursts. A potential problem with these microbursts is that the instantaneous bandwidth requirement can overwhelm link or memory capacities of the scenario under test. This can be alleviated either through external smoothing of the file or by the rate shaping parameters in the simulation configuration file.

– Bandwidth scaling: The standard pcap files are specific captures intended to be representative of network usage. Since they are generic in nature, playback into the simulator is made by bandwidth scaling. This is accomplished by temporal dilation; playback of packets by a constant scale factor, faster or slower, achieves the desired network usage.

Figure 7 shows the bit rate of each pcap stream averaged over 1-second and 5-second non-overlapping windows. See Annex C for a complete set of bandwidth plots.

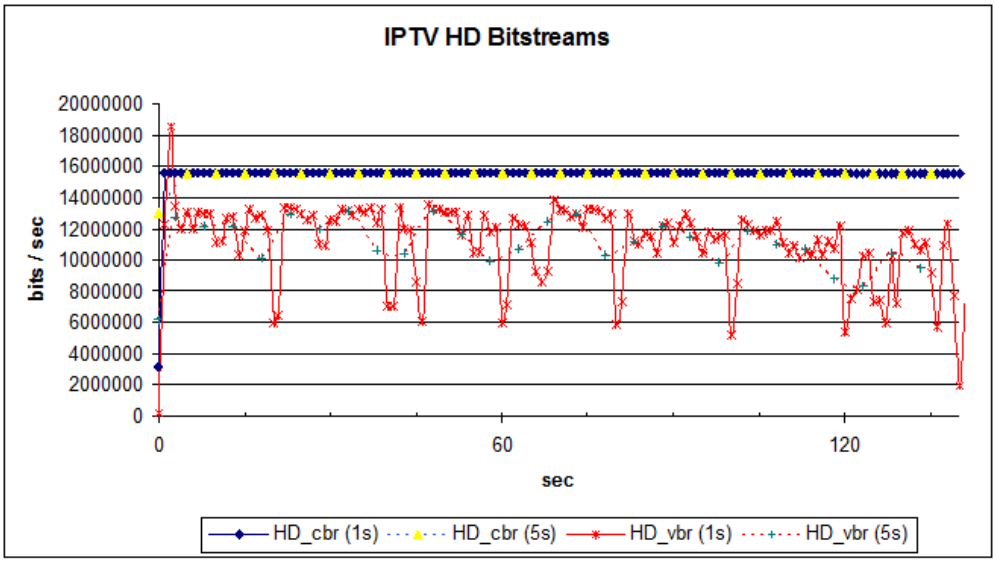


Figure 7 – Sample HD downstream flow

6.5 Simulation inputs

Table 2 lists the inputs to the simulation. The core-to-LAN network topology is assumed. One access technology (DSL or GPON) is selected. There are parameters for each network element and for each pcap file used as an interfering traffic stream.

Traffic flows are assigned a priority that is essential in practical access networks to ensure that higher priority traffic from managed services is carried in preference to lower priority traffic. The simulator has seven priority levels and no per-class bandwidth reservation. The simulation test cases are set as follows:

– 1: (highest priority) managed voice traffic;

– 2: managed IPTV-type video traffic;

– 3-6: unused;

– 7: (lowest priority) best effort iPerf TCP load.

Table 2 – Network model simulation input parameters

Network element	Impairment/Interferer	Parameter range
Core		
	Number of switches	3 to 15
	Total core link delay (ms)	10 to 250
Access (pick one technology)		
GPON	Access rate down (Mbit/s)	5 to 50
	Access rate up (Mbit/s)	2 to 35
	Residual BER	10^{-12} to 10^{-9}
	Delay (ms)	1
DSL	Access rate down (Mbit/s)	3 to 33
	Access rate up (Mbit/s)	1 to 3
	Residual BER	10^{-8} to 10^{-6}
	Delay (ms)	1
	Managed bandwidth	
	IPTV HD Stream 1 – CBR (qty)	0 to 1 (10 core-only)
	Downstream rate (Mbit/s)	8
	IPTV HD Stream 2 – VBR (qty)	0 to 1 (10 core-only)
	Downstream rate (Mbit/s)	8
	IPTV SD Stream 1 – CBR (qty)	0 to 1 (10 core-only)
	Downstream rate (Mbit/s)	2
	IPTV SD Stream 2 – VBR (qty)	0 to 1 (10 core-only)
	Downstream rate (Mbit/s)	2
	VoIP/Fax (qty)	0 to 1 (10 core-only)
	Rate down/up (Mbit/s)	0.064/0.064
Internet services		
	Residual Un-managed bandwidth	
Upstream, downstream, bidirectional	iPerf (TCP)	

6.6 Simulation outputs

Output from the simulator for the managed part of the network is in the form of pcap files and ".out" files. The pcap files are the output packets along with their timestamps. The ".out" files for the managed network cases contain the packet loss and delay values in a matrix. This is shown in Table 3. Comparisons of input pcap and output pcap files can be performed to determine the effect of the impairment on, for example, video/audio quality or voice quality. Test stream pcap files with payloads can be input into the simulation and the test stream output pcaps can be analysed for video/audio quality or voice quality under different network conditions.

The ".out" output files contain delay and packet loss information (in ASCII CSV format) about the simulation result for the managed traffic. This information is also implicitly contained in the pcap file, but because it is not possible directly to indicate packet delay or loss in a pcap file, the ".out" files are helpful for post analysis. For example, because the ".out" files are in CSV format, they can be read into spread sheet programs for analysis. The ".out" output files also can be replayed in a real-time emulator as described in clause 8.1.

The examples in Table 3, Figures 8 and 9 are for the managed part of the network. Table 3 shows part of the ".out" file resulting from a few packets.

The output shown in Figure 10 for the residual part of the network is in the form of a ".out" files and is created by using synthetic TCP loads in the uplink, downlink, as well as bi-directional test cases produces the bandwidth and delay in a matrix. This is shown in Table 4. These values are derived from the iPerf characterization of the residual part of the bandwidth

The examples in Table 4, Figures 10 and 11 are for the residual part of the network and result from the test case DW4.

Table 3 – Example managed bandwidth delay and drop data (Dw4, stream HD_cbr downstream)

Source:	TIA TR30.3: TIA-921C	
Content-encoding :	ASCII	
Delay unit:	ms	
Time	Delay	Drop
14.5956974	77.32	0
14.5970787	77.47	0
14.5984581	77.08	0
14.5998375	77.28	0
14.6012169	76.95	0
14.6039776	76.95	1
14.6039776	77.25	0
14.6053569	76.91	0
14.6067363	77.12	0
14.6081157	77.33	0
14.6094970	77.43	0
14.6108764	77.09	0
14.6122558	77.29	0
14.6136352	76.95	0

Table 3 – Example managed bandwidth delay and drop data (Dw4, stream HD_cbr downstream)

14.6150165	77.11	0
14.6163958	77.26	0
14.6177752	76.92	0
14.6190943	77.22	0
14.6204114	76.89	0
14.6217305	77.15	0

Figure 8 is a plot derived from the same csv file from which Table 3 is a tiny excerpt. Note the dropped packet at around 14.604 seconds.

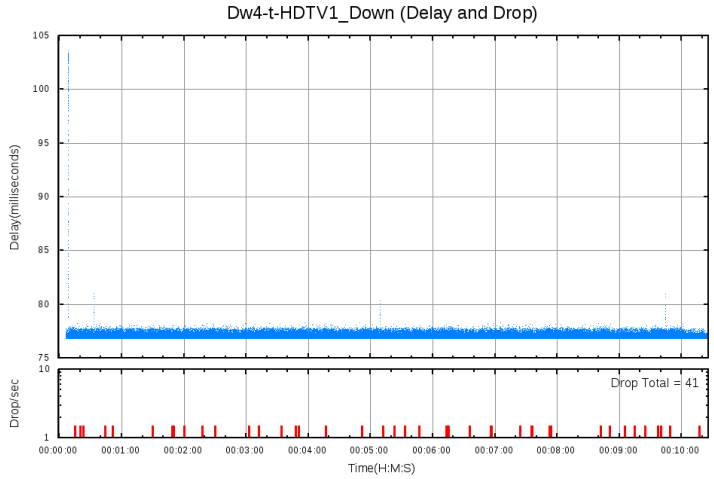


Figure 8 – Time series plot of packet delay and loss

The CSV file can be plotted and analysed to show a time series of the delay patterns and loss bursts. Furthermore, packet delay variation (PDV) histograms and cumulative distribution functions (CDF) can be generated for these files to analyse network characteristics. Figure 9 is the PDV histogram from the same Dw4 simulation.

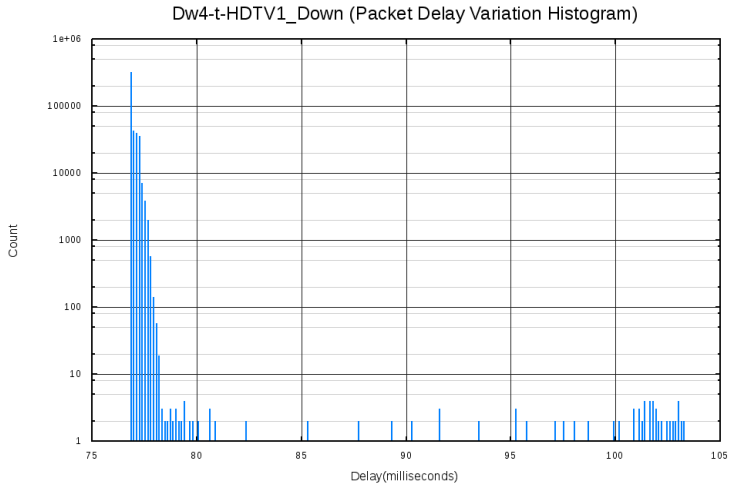


Figure 9 – PDV histogram

Description Table 4 columns:

t – time bin

Bandwidth – average bandwidth of time bin.

Packet – number of packets

mean – mean delay

m^2 – average squared packet delay. This can be used to introduce jitter for network emulation.

Table 4 – Example residual unmanaged bandwidth and delay data

t	Bandwith	Packet	Mean	m^2
1.4	485760	40	77.3	5976.6
1.6	728640	60	78.4	6147.6
1.8	1457280	120	80.0	6409.9
2.0	2064480	170	83.7	7027.3
2.2	2064480	170	82.2	6770.0
2.4	2064480	170	82.2	6770.0
2.6	4189680	345	83.3	6955.0
2.8	2307360	190	83.2	6939.6
3.0	2185920	180	82.3	6789.3
3.2	2185920	180	83.7	7019.0
3.4	2246640	185	83.3	6953.1
3.6	3036000	250	82.4	6813.1
3.8	3521760	290	84.7	7198.1
4.0	2307360	190	83.8	7043.5
4.2	2368080	195	83.6	7014.0
4.4	2428800	200	84.2	7116.2
4.6	2489520	205	84.1	7090.5
4.8	2550240	210	85.1	7271.9
5.0	2732400	225	85.4	7316.8
5.2	2793120	230	85.9	7417.6
5.4	3036000	250	86.1	7443.1
5.6	303600	25	77.2	5959.3
5.8	3218160	265	86.7	7564.3
6.0	3461040	285	95.4	9271.7
6.2	303600	25	77.5	6005.3
6.4	3764640	310	99.8	10208.1
6.6	2914560	240	95.0	9169.8

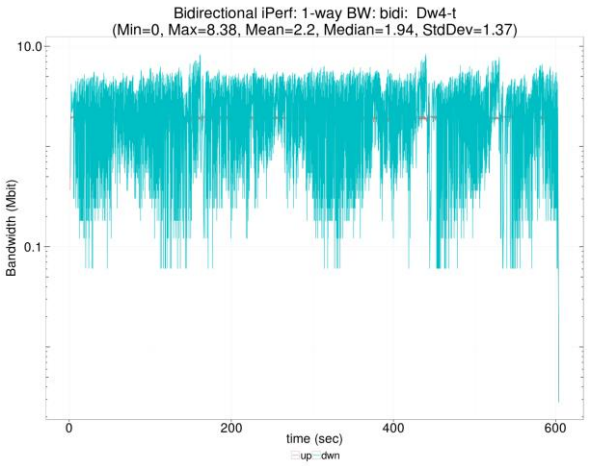


Figure 10 – Residual unmanaged bandwidth

Figure 11 shows example ns-3 simulation of a 6 Mbit link with the following streams: a 2 Mbit CBR IPTV stream, a 2 Mbit VBR stream and a best effort iPerf stream. The synthetic TCP traffic created by iPerf will saturate the link. Green is the resulting residual un-managed bandwidth for available for best effort traffic.

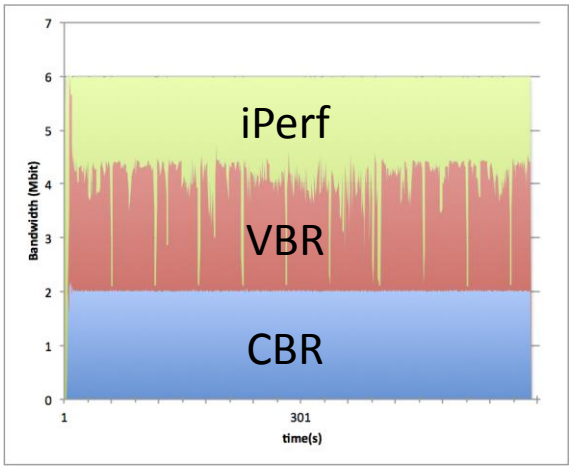


Figure 11 – Example of managed bandwidth with unmanaged synthetic TCP traffic (iPerf) – Stacked bandwidth chart

Addition example plots are shown in the following annexes:

– Annex D – Example output plots for test case Dw4-t;

– Annex E.1 to E.6 – Output merged plots for DSL and GPON.

7 IP network impairment-level requirements

An implementation of the network model shall create impairments that conform to the impairment levels specified in this clause.

7.1 Service test profiles

Table 5 represents service test profiles and the applications, node mechanisms and network techniques associated with them. [ITU-T Y.1541] uses a similar approach, but a one-to-one mapping to these service profiles may not be possible.

Table 5 – Service test profiles

Service test profiles	Applications (examples)	Node mechanisms	Network techniques
Well-managed IP network (Profile A)	High quality video and VoIP, video conferencing (real-time applications, loss sensitive, jitter sensitive, high interaction)	Strict QoS, guaranteed no over subscription on links	Constrained routing and distance
Partially-managed IP network (Profile B)	VoIP, video conferencing (Real-time applications, jitter sensitive, interactive)	Separate queue with preferential servicing, traffic grooming	Less constrained routing and distances
Unmanaged IP network, Internet (Profile C)	Lower quality video and VoIP, signalling, transaction data (highly interactive)	Separate queue, drop priority	Constrained routing and distance
	Transaction data, interactive		Less constrained routing and distances
	Short transactions, bulk data (low loss)	Long queue, drop priority	Any route/path
	Traditional Internet applications	Separate queue (lowest priority)	Any route/path

Table 6 shows industry-accepted impairment levels, including LAN and access that correspond to the service test profiles. The total packet loss shown is the sum of the sequential packet loss and random packet loss. Note that service provider service level agreements (SLAs) only guarantee characteristics of the core section of the network.

Table 6 – Impairment levels per service test profile

Impairment type	Units	Profile A, well-managed range (min to max)	Profile B, partially-managed range (min to max)	Profile C, unmanaged range (min to max)
One-way latency	ms	20 to 100 (regional) 90 to 300 (intercontinental)	20 to 100 (regional) 90 to 400 (intercontinental)	20 to 500
Jitter (peak-to-peak)	ms	0 to 50	0 to 150	0 to 500
Sequential packet loss	ms	Random loss only	40 to 200	40 to 10'000
Rate of sequential loss	s^{-1}	Random loss only	< 10–3	$< 10^{-1}$
Random packet loss	%	0 to 0.05	0 to 2	0 to 20
Reordered packets	%	0 to 0.001	0 to 0.01	0 to 0.1

7.2 Impairment combination standard test cases

The standard test cases of this IP network model have the following characteristics for each service test profile:

– Well-managed network (Profile A) – a network with no over-committed links that employs QoS edge routing. The well-managed test cases include managed voice and video services.

– Partially managed network (Profile B) – a network that minimizes over-committed links and has one or more links without QoS edge routing. The partially managed test cases include a mixture of managed voice and video services, and unmanaged/best-effort data and over-the-top video services.

– Unmanaged network (Profile C) – an unmanaged network such as the Internet that includes over-committed links and has one or more links without QoS edge routing. The unmanaged test cases include unmanaged data and over-the-top video services.

The specific core network parameters, interferers, and access network parameters for each test case were selected so that the simulation results align with the impairment level requirements of clause 7.1, with an adjustment to account for the fact that the simulations only cover the core-to-LAN topology. The interferers represent a realistic mix of typical traffic. The test cases span a range of impairment severities from mild to severe within each service test profile.

The test cases are specified in Tables 7, 8 and 9. Each test case is labelled in Table E.1 as follows:

– The first character is either D for DSL or G for GPON. The access rates and impairments chosen represent typical values in 2010. (The single core-only test case, lacking an access network portion, is labelled simply "Core.").

– The second character is one of the following:

  • **w** for well-managed profile A, highlighted in green in Table 7.

  • **p** for partially managed profile B, highlighted in yellow in Table 8.

  • **u** for unmanaged profile C, highlighted in red in Table 9.

The third character is an ordinal digit, where a higher value generally corresponds to a higher severity (more difficult) test case.

For example, test case Dw5 uses DSL access in a well-managed network with impairment severity 5. Note that all simulations are for the core-to-LAN network topology of Figure 4.

A given test case column uses the identical interfering streams and parameters between the DSL and GPON versions. Only the access technology parameters differ.

The column "PCAP Avg BW" shows the long-term average bit rate of each pcap file after smoothing, but before scaling. The percentages shown for each pcap file under each test case are the time scale factors applied to the interferers by the packet generator. A higher percentage decreases the inter-packet time, which increases the effective bit rate of the interferer.

Table 7 – Well-managed service test profile A test cases

Network Element	Impairment / Interferer	PCAP Avg BW (Mb/s)	Well Managed							
			w1	w2	w3	w4	w5	w6	w7	w8
Core										
	Number of Switches		3	4	5	6	7	8	9	10
	Total Core Link Delay (ms)		10	25	50	75	85	100	125	150
Access (pick one technology)										
GPON	Access Rate Down (Mbit/s)		50	50	35	35	25	25	25	15
	Access Rate Up (Mbit/s)		25	25	25	25	25	25	25	5
	Residual BER		1.E-12	1.E-12	1.E-11	1.E-10	1.E-10	1.E-09	1.E-09	1.E-09
DSL	Access Rate Down (Mbit/s)		33	33	33	22	16	16	16	10
	Access Rate Up (Mbit/s)		3	3	3	2	1	1	1	1
	Residual BER		1.E-08	1.E-08	1.E-08	1.E-08	1.E-07	1.E-07	1.E-07	1.E-07
	Managed Bandwidth	**(QoS Voice=1 Video=2)**								
	IPTV HD Stream 1 - CBR (qty)	8	1	1	1	1	1	1	1	1
	IPTV HD Stream 2 - VBR (qty)	8			1					
	IPTV SD Stream 1 - CBR (qty)	2		1		1	1	1	1	
	IPTV SD Stream 2 - VBR (qty)	2				1		1	1	
	VoIP/Fax (qty)	0.064/0.064	1	1	1	1	1	1	1	1
	Residual Bandwidth (Internet Services)	**QoS = 7**								
	Bandwidth and Delay Variation									

Table 8 – Partially-managed service test profile B test cases

Network Element	Impairment / Interferer	PCAP Avg BW (Mb/s)	Unmanaged						
			u1	u2	u3	u4	u5	u6	u7
Core									
	Number of Switches		3	4	5	8	10	12	15
	Total Core Link Delay (ms)		10	25	50	75	100	200	250
Access (pick one technology)									
GPON	Access Rate Down (Mbit/s)		35	35	25	15	5	15	5
	Access Rate Up (Mbit/s)		35	15	25	5	2	5	2
	Residual BER		1.E-12	1.E-12	1.E-11	1.E-09	1.E-09	1.E-09	1.E-09
DSL	Access Rate Down (Mbit/s)		24	18	12	6	3	6	3
	Access Rate Up (Mbit/s)		3	1.5	1.5	1	1	1	1
	Residual BER		1.E-08	1.E-07	1.E-07	1.E-06	1.E-06	1.E-06	1.E-06
	Managed Bandwidth	**(QoS Voice=1 Video=2)**							
	IPTV HD Stream 1 - CBR (qty)	8	There are no managed steams in the Unmanaged service test profile C						
	IPTV HD Stream 2 - VBR (qty)	8							
	IPTV SD Stream 1 - CBR (qty)	2							
	IPTV SD Stream 2 - VBR (qty)	2							
	VoIP/Fax (qty)	0.064/0.064							
	Residual Bandwidth	**QoS = 7**							
	Bandwidth and Delay Variation								

Table 9 – Unmanaged service test profile C cases

Network Element	Impairment / Interferer	PCAP Avg BW (Mb/s)	Partially Managed					
			p1	p2	p3	p4	p5	p6
Core								
	Number of Switches		3	4	5	8	10	12
	Total Core Link Delay (ms)		10	25	50	75	100	200
Access (pick one technology)								
GPON	Access Rate Down (Mbit/s)		35	35	25	15	5	15
	Access Rate Up (Mbit/s)		35	15	25	5	2	5
	Residual BER		1.E-12	1.E-12	1.E-11	1.E-09	1.E-09	1.E-09
DSL	Access Rate Down (Mbit/s)		24	18	12	6	3	6
	Access Rate Up (Mbit/s)		3	1.5	1.5	1	1	1
	Residual BER		1.E-08	1.E-07	1.E-07	1.E-06	1.E-06	1.E-06
	Managed Bandwidth	**(QoS Voice=1 Video=2)**						
	IPTV HD Stream 1 - CBR (qty)	8						
	IPTV HD Stream 2 - VBR (qty)	8						
	IPTV SD Stream 1 - CBR (qty)	2	1	1	1	1	0	1
	IPTV SD Stream 2 - VBR (qty)	2	1	1	1	1	1	1
	VoIP/Fax (qty)	0.064/0.064	1	1	1	1	1	1
	Residual Bandwidth (Internet Services)	**QoS = 7**						
	Bandwidth and Delay Variation							

Figure 12 shows the aggregate bit rate over time for the DSL well-managed profile. Test profile A has the following characteristics:

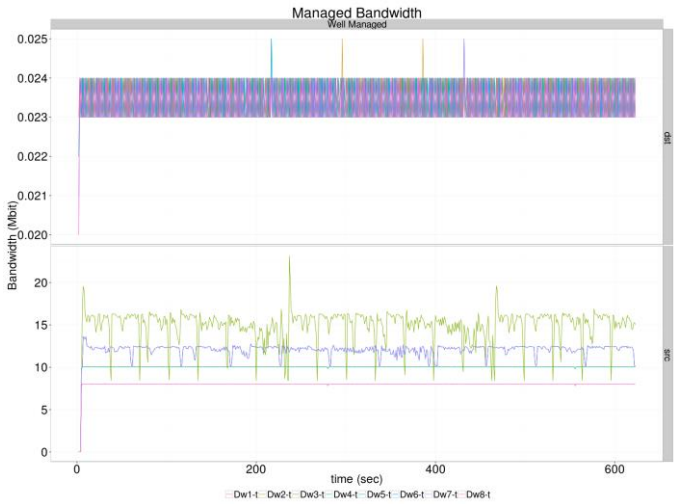


Figure 12 – Managed bandwidth for DSL well-managed profile A both directions

Figure 13 shows the aggregate bit rate over time for the DSL partially-managed profile. Test profile B has the following characteristics:

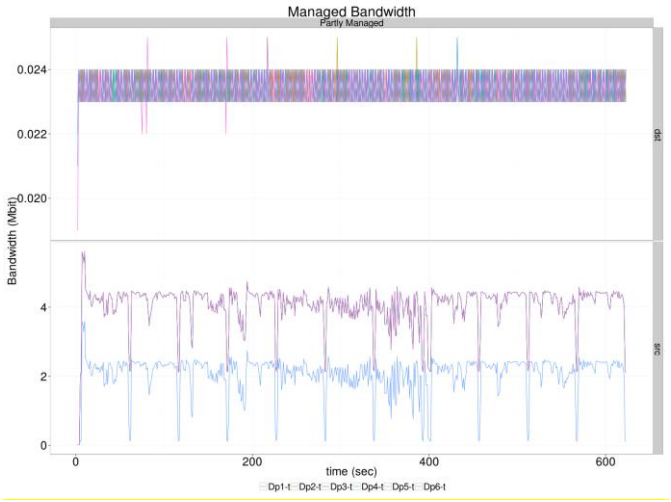


Figure 13 – Managed bandwidth for DSL partially-managed profile B both directions

NOTE – The Un-managed profile has no managed traffic, therefore there are no managed traffic plots. All of the bandwidth is available for iPerf.

8 Using the network model

The network model described by this Recommendation is implemented as a software simulator. For each test case, the simulator models the network behaviour while carrying a mixture of user datagram protocol (UDP) and TCP traffic. The UDP packets are taken from pcap files that represent real world managed services, such as IPTV or VoIP. The TCP packets are generated by an iPerf client and server that run within the simulator. The priority levels assigned to the UDP packets varies according to the specific test case. For all test cases, the priority level assigned to the TCP stream is best effort.

For the UDP flows, the simulator records the latency for each packet, or if the packet is discarded during the simulation, the simulator records that it was dropped. These results are saved in an output file and can be used directly in an inline network impairment emulator.

For the TCP stream, the simulator records the packets to a pcap file. At the conclusion of the simulation, the saved pcap file is analysed to determine the time-varying capacity that it used. This is taken to represent the network capacity available for best effort services. For each 200-millisecond time slot in the pcap file, the packet delay (latency), bandwidth and loss, are measured and these results are saved to an output file.

The output files for all the test cases are published in electronic form as Annex H of this Recommendation. The output files can then be used in an in-line network impairment emulator to evaluate the performance of a service carried over a best-effort TCP stream.

Alternatively, the simulator can also be used as a stand-alone evaluation platform to determine the residual best-effort capacity of a network under customized test cases that are different from the predefined test cases in this Recommendation. For this, a user may select different network parameters (link speeds, bit error rates or priorities), or select different interference traffic patterns. If desired, the simulator output for such customized test cases may also be used in an in-line network impairment emulator.

8.1 Using an in-line network impairment emulator

Figure 14 illustrates user defined test streams being replayed through the residual unmanaged bandwidth of a network model. Figure 15 shows an example of a lab test set-up where data is played between a device under test A (data source) and device under test B (data sink) through an in-line network impairment emulator.

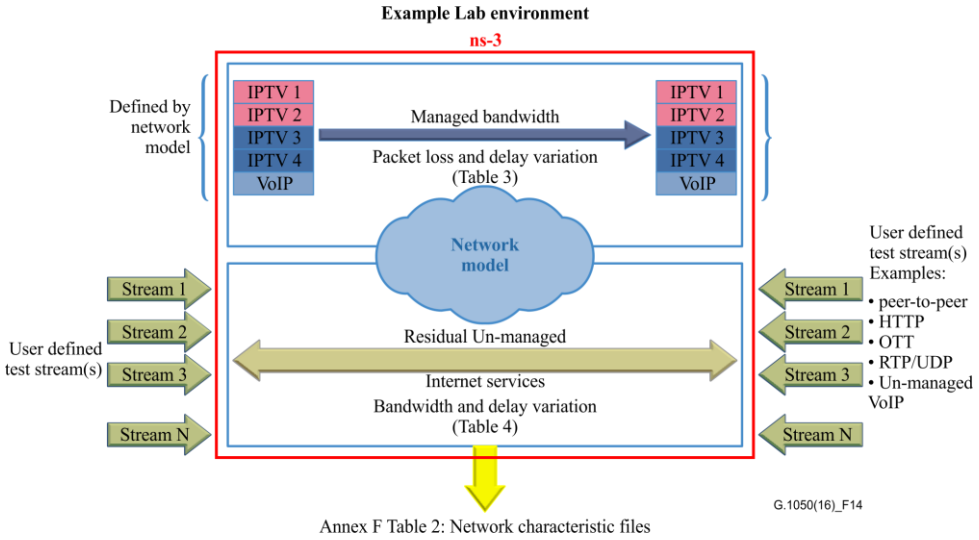


Figure 14 – Test user applications with network characteristic files

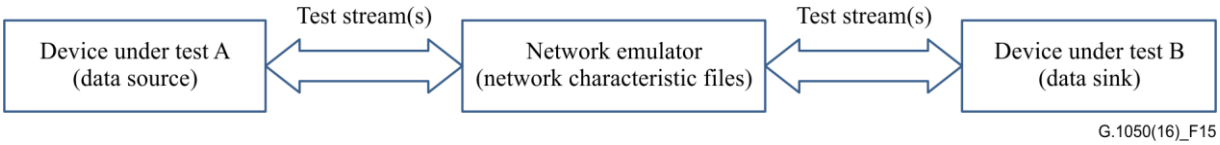


Figure 15 – Hardware emulator test set-up

The in-line network impairment emulator could be used as follows:

– network characteristic files can be replayed by a network emulator to test user defined stream(s) or applications;

– select a test case that includes a technology similar to the test stream;

– configure the emulator with the standard .out file from Annex F Table 2 for the selected test case:

  • Table 3 – Managed bandwidth – replay delay and drop;

  • Table 4 – Residual bandwidth – replay bandwidth and delay.

– run the test stream(s) through the emulator, and evaluate the effect of the network impairments on the stream.

8.2 Using the simulator to create custom test cases

Modifications to existing test cases can be made as follows:

– Select a standard test case as a starting point. See Annex G which provides a summary of all the test cases. This can be used as a roadmap to select the test case of interest.

– Replace or change settings in the parameter file to correspond to a desired test scenario. For example, link bit rates, error rates and latencies can be changed, as well as the number of core switches, or competing impairment streams. See clause A.9.

– Run the simulation, which results in a .out file, and optionally pcap captures for analysis.

– Evaluate performance using the simulator results.

– Optionally run the results on an in-line network impairment emulator.

8.3 Hardware emulator considerations

For a specified test case, the simulation produces bandwidth, delay and loss characteristics that can be used in a real-time emulator. The emulator accepts a packet stream under test, and plays out the same stream with impairments that match the corresponding simulation.

The simulator creates .out files to represent the delay and loss experienced by each packet in a stream. These .out files can be used directly to control packet delay and loss in a real-time inline impairment emulator.

In cases where the packet stream in the simulator is bursty, the delay and loss values in the .out file are non-uniform in time. Two methods to handle this situation are described in the following paragraphs. An impairment emulator shall provide Method 1 and may provide Method 2.

Method 1: Resample the delay and loss values in the .out file so that they are uniformly sampled in time, and apply these in the inline impairment emulator using the resampled rate. For example, if the test stream is bursty with an average packet rate of 1000 packets per second, the .out file may be resampled using a 1 ms resample rate, and the resulting impairments could then be applied at a 1 ms update rate. This approach ensures that the resulting test always lasts the same amount of time, but some samples may apply to more than one packet, and other samples may not be applied to any packets, depending on the arrival time of packets at the inline impairment emulator.

Method 2: Apply the delay and loss values from the .out files on a packet-by-packet basis. Each line in the .out file is applied to one packet without regard to the time value in the first column. This approach ensures that each delay and loss value is applied exactly once, but the test duration will vary inversely with the rate of packets applied to the inline impairment emulator. In other words, a higher packet rate applied to the emulator will consume the .out file samples more quickly, and thus the test finishes sooner.

If a long duration test is required to last for more than the 10-minute length of the .out files, the files should be looped.

An impairment emulator shall shape the stream under test to account for the "self-interference" effect by traffic shaping using the token bucket method. Packets that arrive close together in time contend for the available bandwidth in the emulated network. The amount of shaping depends on the effective bandwidth available to the stream under test in the given test case. This available capacity is the rate of the access link (the lowest-rate link in the network) less the sum of the rates of the interfering streams at QoS priority the same or higher than the stream under test.

The test cases described in this Recommendation cover primarily CORE-TO-LAN configurations, and this is well suited for emulation of IPTV service and newer over-the-top streaming video services. Test cases for LAN-TO-LAN configurations shall be constructed by concatenating two CORE-TO-LAN test cases. For example, a LAN-TO-LAN test case could be constructed by running Dw1 in series with Gp3. Such a test case is called Dw1-Gp3. The delay and loss values for a LAN-TO-LAN test case are calculated by first performing Method 1 on the individual test cases so that they have a common sample rate and then summing the delay values and logically OR-ing the loss values for the resulting individual test cases. The bandwidth limitation is calculated as the minimum of the effective bandwidth (EB) for the individual test cases.

$$Delay_{Dw1\text{-}Gp2} = Delay_{Dw1} + Delay_{Gp2}$$

$$Loss_{Dw1\text{-}Gp2} = Loss_{Dw1} \text{ "or" } Loss_{Gp2}$$

$$BW_{Dw1\text{-}Gp2} = Min[EB_{Dw1}, EB_{Gp2}]$$

8.4 Advanced uses of the network model

User-defined test cases: the user can define custom test cases by any of these methods:

– replace any of the standard interferer pcap files;

– change the traffic mix;

– change network conditions: number of switches, bit error ratio, access technology parameters, etc.

Network load generation: the output pcap files can be used as the basis of synthetic or dummy loads to inject onto networks to simulate different traffic mixes.

Annex A

Description of simulator

(This annex forms an integral part of this Recommendation.)

A.1 Simulator overview

The ITU-T G.1050-2016 impairment model is implemented with the ns-3 network simulator. ns-3 is a discrete-event network simulator for Internet systems. It is an open-source project written mainly in the C++ language using object oriented techniques and is licensed under the GNU Public License (v2).

ns-3 is available to the public for research, development and use. The ns-3 project is hosted at the website www.nsnam.org, where source code, documentation and project announcements can be found. There are several email discussion forums that allow project members to communicate with each other and to publicly announce events such as new releases and upcoming conferences.

ns-3 contains models of network elements, such as routers and communication links, as well as many popular network protocols such as TCP/IP. Events in the simulation consist mainly of packet arrivals and departures from the different network elements, but other types of events are also possible.

An important reason for choosing ns-3 as the basis for the ITU-T G.1050-2016 model is that it contains a working TCP/IP network protocol stack. Attempts to add TCP/IP to the previous revision of the ITU-T G.1050 model proved difficult. Furthermore, ns-3 is able to run actual network code in the DCE virtual environment, which ensures the realism of the simulation. Specifically, the simulations use the Linux 2.6.36 network stack, but the DCE simulation environment will support other network stacks for future extensibility.

The ITU-T G.1050-2016 impairment model is built on top of the base ns-3 code and has two main parts: patches and additions. The patches modify existing parts of ns-3 to extend functionality or fix minor errors. During the ITU-T G.1050 build process, the patches are applied to the base ns-3 code. Some or all of these patches may be incorporated in a future release of the base ns-3 code. The additions are new files that add features or functionality beyond what is available in the base ns-3 code. During the ITU-T G.1050 build process, these additional files are overlaid onto the base ns-3 directory structure and compiled with it. ns-3 release 3.24.1 was used as the basis for the simulations described in this Recommendation.

The Core-to-LAN network topology described in clause 6.2 has been implemented in ns-3, and during simulations various traffic streams are transferred over this topology. The base traffic load on the model consists of various managed streams such as IPTV and VoIP. The residual capacity of the network beyond the base load is available for unmanaged streams, such as *Over-the-top* (OTT) video.

The managed traffic streams in this model are represented by real-world packet captures contained in PCAP files. These managed streams are played into the model by reading the PCAP file, injecting the packets into the network from a simulated source node and addressing them to the simulated destination node. These packets form the background load carried by the Core-to-LAN topology.

The unmanaged streams in the model are represented by TCP sessions between a simulated server and a simulated client. The server and client run a program called iPerf. iPerf sends data from the client to the server using a TCP session, and it sends data as fast as allowed by TCP. The packets carrying the iPerf session are given *best effort* priority. The iPerf session does a good job of measuring network capacity available to TCP-based best-effort services while competing with other managed services such as IPTV and VoIP.

As the simulation runs, the iPerf packets are captured into PCAP files at the client and server nodes within the simulation. After the simulation finishes, the PCAP files are analysed to determine how the network capacity (or bandwidth) and packet delay vary over time during the simulation. The average bandwidth and delay are calculated once every 200 milliseconds and recorded in a file that can then be used to test other TCP-based best-effort services, for example with a network impairment emulator.

As mentioned earlier, ns-3 is able to directly execute actual software for network protocols and applications. This is called direct code execution (DCE*)* and is a virtual execution environment for real-world network code. A simulated network node in ns-3 can run a virtualized instance of an actual TCP/IP network stack implementation, such as running the actual Linux kernel code for TCP/IP. A simulated network node can also run a virtualized instance of a network application, such as running the actual code for iPerf. Using DCE is better than running a separately implemented model because it uses the same source code of real systems, and has the same functionality. However, using DCE also slows down the overall simulation.

For ITU-T G.1050-2016, the iPerf program and the supporting Linux kernel TCP/IP stack running on the client and server nodes are implemented as DCE virtual environments. The iPerf instances are compiled from publicly available iPerf version 2.0.5 source code. The Linux kernel source code on the client and server nodes is Linux version 2.6.36. Because of this, various TCP congestion control algorithms that exist in that version of Linux (e.g., reno, bic, cubic) can be selected and run. For this Recommendation, the cubic congestion control algorithm is used. Because TCP is not used on the other nodes of the simulated Core-to-LAN network and because using DCE slows down the simulation, the remaining network nodes use the regular ns-3 protocol models. The reason for this is to improve the runtime of the simulation.

A.2 Directory structure

`./`	top level directory
`analysis/`	directory of scripts for post analysis
`dce/`	contains ns-3 and dce source code
`ns-3.24.1-base.diff`	contains ITU-T G.1050 patches to ns-3
`overlay/`	contains ITU-T G.1050 overlay code
`setenv.sh`	script to set environment variables
`setup.sh`	script that calls download, patch, overlay & build
`download.sh`	script to download ns-3 source code
`patch.sh`	script to apply ITU-T G.1050 patches to ns-3
`overlay.sh`	script to overlay ITU-T G.1050 files to ns-3
`build.sh`	script to build the modified ITU-T G.1050 simulator
`rebuild.sh`	script to rebuild the simulator
`dce/source/ns-3-dce/myscripts/core2lan`	contains ITU-T G.1050 core2lan model
`dce/source/ns-3-dce/tc`	contains ITU-T G.1050 test case files
`dce/source/ns-3-dce/pcap`	contains ITU-T G.1050 PCAP input files
`dce/source/ns-3-dce/out`	contains ITU-T G.1050 output files

A.3 Building the simulator

The source code for the simulation is written in the C++ language and has been tested in Ubuntu 14.04.

The following is a list of Linux packages that are required in order to build and run the simulator:

```
build-essential, pkg-config, subversion, wireshark, flex, bison, gnuplot, tcl, tcl-dev,
libboost-dev, libboost-doc, libboost-all-dev, python-pygccxml, python-pygraphviz, mercurial,
python-pygoocanvas, bzr, cmake, cvs, git, unrar, p7zip-full, autoconf, python, qt4, qt4-qmake,
qt4-dev-tools, libpcap-dev, libc6-dev, libc6-dbg, mercurial, libdb-dev, aptitude, libssl-dev
```

To build the simulator, first run the script setenv.sh to set the necessary environment variables.

```
$ source ./setenv.sh
```

Then, run the script setup.sh

```
$ ./setup.sh
```

This script has four phases:

1) Downloads the base ns-3 simulator code (ns-3.24.1);

2) Applies the ITU-T G.1050-2016 patches that are part of this annex;

3) Adds the ITU-T G.1050-2016 overlay files;

4) Builds the modified simulator.

It should be noted that the setup.sh script performs a full download and build, and if it runs successfully, no other commands are needed to prepare the simulator.

A.4 Download phase

The following shows a simplified version of the messages that are printed during the download phase of the set-up. The download phase is based on a script published on the nsnam.org website for getting started with DCE.

```
$ ./setup.sh
+ source ./setenv.sh
+ hg clone http://code.nsnam.org/bake bake
requesting all changes
adding changesets
adding manifests
adding file changes
added 331 changesets with 787 changes to 63 files
updating to branch default
45 files updated, 0 files merged, 0 files removed, 0 files unresolved
+ mkdir dce
+ cd dce
+ bake.py configure -e dce-linux-1.4
+ bake.py download -vvv
 >> Download pybindgen-0.17.0.876 - OK
 >> Search g++ - OK
 >> Search qt4 - OK
 >> Download bash - OK
 >> Search python-dev - OK
 >> Search pygraphviz - OK
 >> Search pygoocanvas - OK
 >> Search libpcap-dev - OK
 >> Search libexpat-dev - OK
 >> Download iperf - OK
 >> Search libdb-dev - OK
 >> Search bison - OK
 >> Search flex - OK
 >> Search libssl-dev - OK
 >> Download net-next-sim-2.6.36 - OK
 >> Download wget - OK
 >> Download thttpd - OK
 >> Download elf-loader - OK
 >> Download netanim-3.105 - OK
 >> Download ccnx - OK
 >> Download iproute-2.6.38-fix-01 - OK
 >> Download ns-3.24.1 - OK
 >> Download dce-meta-1.4 - OK
 >> Download dce-linux-1.4 - OK
```

A.5 Patch phase to apply ITU-T G.1050-2016 patches

Once all of the base files for ns-3 have been downloaded, some of the files are patched using the "patch" utility. The patches that are applied are maintained in a file called ns-3.24.1-base.diffs. The patches are organized in three groups according to the part of the base code that is patched.

The first group of files that are patched belong to the core of the NS3 simulator.

source/ns-3.24.1/src/internet/model/global-router-interface.cc
source/ns-3.24.1/src/internet/model/global-router-interface.h
source/ns-3.24.1/src/network/utils/pcap-file.cc
source/ns-3.24.1/src/network/utils/pcap-file.h
source/ns-3.24.1/src/network/utils/pcap-file-wrapper.cc
source/ns-3.24.1/src/network/utils/pcap-file-wrapper.h
source/ns-3.24.1/src/network/helper/trace-helper.h
source/ns-3.24.1/src/internet/helper/ipv4-routing-helper.cc

– The patch to the global-router-interface code allows Ethernet switches to be connected together within a single layer 2 network.

– The patch to the pcap-file code extends the capabilities of the existing code to support nanosecond resolution time stamps, reading from PCAP files, rewinding PCAP files and fixes an error in the declaration of the magic number for nanosecond timestamp PCAP files.

– The patch to trace-helper fixes an error in the default value of arguments that control the default capture length of packets to PCAP files.

The second group of files that are patched belong to the Linux kernel code that is run within the DCE environment of NS3.

source/net-next-sim-2.6.36/net/core/sock.c
source/net-next-sim-2.6.36/net/socket.c
source/net-next-sim-2.6.36/net/ipv4/ip_input.c

– The patch to the ip_input.c code fixes a problem that occurs in ns-3 simulations in which some nodes use the Linux TCP/IP stack while other nodes use the native ns-3 TCP/IP stack. The problem occurs because the default behavior for Linux is to discard packets that have an incorrect IP header checksum, but the default behaviour of ns-3 is for these checksums to be disabled. Because calculating checksums is time consuming and slows down the simulation, instead of enabling checksum calculation in ns-3, the patch to ip_input.h allows the Linux TCP/IP stack to recognize and process packets containing IP checksums that are set to zero (instead of discarding them). This behaviour is modelled upon the existing and well-known behaviour for receiving UDP packets with UDP checksums that are zero.

The following shows a simplified version of the messages printed during the patch phase of the setup:

```
+ patch --verbose -p1
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c -r ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/model/global-router-interface.cc
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/model/global-router-interface.cc
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/internet/model/global-router-interface.cc
--------------------------
patching file source/ns-3.24.1/src/internet/model/global-router-interface.cc
Using Plan A...
Hunk #1 succeeded at 1242.
Hunk #2 succeeded at 1256.
Hunk #3 succeeded at 1289.
Hunk #4 succeeded at 1331.
Hunk #5 succeeded at 1355.
Hunk #6 succeeded at 1406.
Hunk #7 succeeded at 1419.
Hunk #8 succeeded at 1460.
Hunk #9 succeeded at 1485.
Hunk #10 succeeded at 1744.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
```

```
--------------------------
|diff -c -r ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/model/global-router-interface.h
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/model/global-router-interface.h
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/internet/model/global-router-interface.h
--------------------------
patching file source/ns-3.24.1/src/internet/model/global-router-interface.h
Using Plan A...
Hunk #1 succeeded at 769 (offset 1 line).
Hunk #2 succeeded at 783 (offset 1 line).
Hunk #3 succeeded at 848 (offset 1 line).
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c -r ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/drop-tail-queue.cc
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/drop-tail-queue.cc
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/utils/drop-tail-queue.cc
--------------------------
patching file source/ns-3.24.1/src/network/utils/drop-tail-queue.cc
Using Plan A...
Hunk #1 succeeded at 35.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.24.1.ORIG/source/ns-3.24.1/src/network/helper/trace-helper.h
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/helper/trace-helper.h
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/helper/trace-helper.h
--------------------------
patching file source/ns-3.24.1/src/network/helper/trace-helper.h
Using Plan A...
Hunk #1 succeeded at 100 with fuzz 1 (offset 19 lines).
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file.cc
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file.cc
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/utils/pcap-file.cc
--------------------------
patching file source/ns-3.24.1/src/network/utils/pcap-file.cc
Using Plan A...
Hunk #1 succeeded at 40.
Hunk #2 succeeded at 48 (offset -1 lines).
Hunk #3 succeeded at 85 (offset -1 lines).
Hunk #4 succeeded at 161 (offset -1 lines).
Hunk #5 succeeded at 307 (offset -1 lines).
Hunk #6 succeeded at 331 (offset -1 lines).
Hunk #7 succeeded at 346 (offset -1 lines).
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file.h
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file.h
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/utils/pcap-file.h
--------------------------
patching file source/ns-3.24.1/src/network/utils/pcap-file.h
Using Plan A...
Hunk #1 succeeded at 116.
Hunk #2 succeeded at 172.
Hunk #3 succeeded at 199.
Hunk #4 succeeded at 366.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.cc
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.cc
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.cc
--------------------------
patching file source/ns-3.24.1/src/network/utils/pcap-file-wrapper.cc
Using Plan A...
Hunk #1 succeeded at 17.
Hunk #2 succeeded at 40.
Hunk #3 succeeded at 58.
Hunk #4 succeeded at 110.
```

```
Hunk #5 succeeded at 124.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -c ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.h
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.h
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/network/utils/pcap-file-wrapper.h
--------------------------
patching file source/ns-3.24.1/src/network/utils/pcap-file-wrapper.h
Using Plan A...
Hunk #1 succeeded at 48.
Hunk #2 succeeded at 107.
Hunk #3 succeeded at 144.
Hunk #4 succeeded at 230.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.18.ORIG/source/net-next-sim-2.6.36/arch/sim/sim-socket.c
|*** ns-3.18.ORIG/source/net-next-sim-2.6.36/arch/sim/sim-socket.c
|--- ns-3.18.TIA/source/net-next-sim-2.6.36/arch/sim/sim-socket.c
--------------------------
patching file source/net-next-sim-2.6.36/arch/sim/sim-socket.c
Using Plan A...
Hunk #1 succeeded at 174 (offset 13 lines).
Hunk #2 succeeded at 193 (offset 13 lines).
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.18.ORIG/source/net-next-sim-2.6.36/net/core/sock.c
|*** ns-3.18.ORIG/source/net-next-sim-2.6.36/net/core/sock.c
|--- ns-3.18.TIA/source/net-next-sim-2.6.36/net/core/sock.c
--------------------------
patching file source/net-next-sim-2.6.36/net/core/sock.c
Using Plan A...
Hunk #1 succeeded at 486.
Hunk #2 succeeded at 533.
Hunk #3 succeeded at 546.
Hunk #4 succeeded at 568.
Hunk #5 succeeded at 596.
Hunk #6 succeeded at 782.
Hunk #7 succeeded at 817.
Hunk #8 succeeded at 1311.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.18.ORIG/source/net-next-sim-2.6.36/net/socket.c.
|*** ns-3.18.ORIG/source/net-next-sim-2.6.36/net/socket.c
|--- ns-3.18.TIA/source/net-next-sim-2.6.36/net/socket.c
--------------------------
patching file source/net-next-sim-2.6.36/net/socket.c
Using Plan A...
Hunk #1 succeeded at 1756.
Hunk #2 succeeded at 1790.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.18.ORIG/source/net-next-sim-2.6.36/net/ipv4/ip_input.c
|*** ns-3.18.ORIG/source/net-next-sim-2.6.36/net/ipv4/ip_input.c
|--- ns-3.18.TIA/source/net-next-sim-2.6.36/net/ipv4/ip_input.c
--------------------------
patching file source/net-next-sim-2.6.36/net/ipv4/ip_input.c
Using Plan A...
Hunk #1 succeeded at 415.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.24.1.ORIG/source/ns-3-dce/helper/linux-stack-helper.cc
|*** ns-3.24.1.ORIG/source/ns-3-dce/helper/linux-stack-helper.cc
|--- ns-3.24.1.TIA/source/ns-3-dce/helper/linux-stack-helper.cc
--------------------------
patching file source/ns-3-dce/helper/linux-stack-helper.cc
Using Plan A...
```

```
Hunk #1 succeeded at 64.
Hunk #2 succeeded at 117.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.24.1.ORIG/source/ns-3-dce/model/dce-fd.cc
|*** ns-3.24.1.ORIG/source/ns-3-dce/model/dce-fd.cc
|--- ns-3.24.1.TIA/source/ns-3-dce/model/dce-fd.cc
--------------------------
patching file source/ns-3-dce/model/dce-fd.cc
Using Plan A...
Hunk #1 succeeded at 524 (offset 2 lines).
Hunk #2 succeeded at 535 (offset 2 lines).
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.24.1.ORIG/source/ns-3-dce/model/unix-socket-fd.cc
|*** ns-3.24.1.ORIG/source/ns-3-dce/model/unix-socket-fd.cc
|--- ns-3.24.1.TIA/source/ns-3-dce/model/unix-socket-fd.cc
--------------------------
patching file source/ns-3-dce/model/unix-socket-fd.cc
Using Plan A...
Hunk #1 succeeded at 245.
Hunk #2 succeeded at 303.
Hunk #3 succeeded at 318.
Hunk #4 succeeded at 484.
Hunk #5 succeeded at 555.
Hunk #6 succeeded at 574.
Hmm...  The next patch looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|diff -cr ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/helper/ipv4-routing-helper.cc
|*** ns-3.24.1.ORIG/source/ns-3.24.1/src/internet/helper/ipv4-routing-helper.cc
|--- ns-3.24.1.TIA/source/ns-3.24.1/src/internet/helper/ipv4-routing-helper.cc
--------------------------
patching file source/ns-3.24.1/src/internet/helper/ipv4-routing-helper.cc
Using Plan A...
Hunk #1 succeeded at 66.
done
```

A.6 Overlay phase – adding new code to the ITU-T G.1050-2016 simulator

During the overlay phase, soft links are created to mirror the content of the files within the ./overlay directory tree into the ns-3 source code. This allows new files and modules to be added without modifying the existing distribution, while also allowing the overlay files to be maintained separately.

The following is a list of the overlay files and a brief explanation of what each does.

overlay/source/ns-3.24.1/src/tia921/model/asymmetric-link-channel.cc
overlay/source/ns-3.24.1/src/tia921/model/asymmetric-link-channel.h
overlay/source/ns-3.24.1/src/tia921/model/asymmetric-link-net-device.cc
overlay/source/ns-3.24.1/src/tia921/model/asymmetric-link-net-device.h
overlay/source/ns-3.24.1/src/tia921/helper/asymmetric-link-helper.cc
overlay/source/ns-3.24.1/src/tia921/helper/asymmetric-link-helper.h

The asymmetric-link source code implements an asymmetric communication channel similar to typical access link such as the remote access (RACC) or local Access (LACC) channel in the CORE2LAN model. It has configurable speed, latency and bit error rate for each direction of transmission.

overlay/source/ns-3.24.1/src/tia921/model/fdxeth-channel.cc
overlay/source/ns-3.24.1/src/tia921/model/fdxeth-channel.h
overlay/source/ns-3.24.1/src/tia921/model/fdxeth-net-device.cc
overlay/source/ns-3.24.1/src/tia921/model/fdxeth-net-device.h
overlay/source/ns-3.24.1/src/tia921/helper/fdxeth-helper.cc
overlay/source/ns-3.24.1/src/tia921/helper/fdxeth-helper.h

The fdxeth source code implements a full-duplex Ethernet communication model. ns-3 does not have a model for a full duplex Ethernet link. Therefore, this model is required. It has configurable speed and latency. It also implements packet transmission in a way that more accurately reflects the physical

coding sublayer of the Ethernet protocol, in particular with regard to the required inter-frame gap and also generates transmission events in an order that allows future expansion to model cut-through switching.

overlay/source/ns-3.24.1/src/tia921/model/ethernet-priority-queue.cc
overlay/source/ns-3.24.1/src/tia921/model/ethernet-priority-queue.h

The Ethernet priority queue code implements priority queuing similar to what is used as part of the virtual LAN tag.

overlay/source/ns-3.24.1/src/tia921/model/pcap-packet-receiver.cc
overlay/source/ns-3.24.1/src/tia921/model/pcap-packet-receiver.h
overlay/source/ns-3.24.1/src/tia921/model/pcap-packet-sender.cc
overlay/source/ns-3.24.1/src/tia921/model/pcap-packet-sender.h
overlay/source/ns-3.24.1/src/tia921/helper/pcap-sendrecv-helper.cc
overlay/source/ns-3.24.1/src/tia921/helper/pcap-sendrecv-helper.h

The PCAP packet sender and receiver files implement objects to send and receive packets through an ns-3 simulation by reading from and writing to PCAP files.

overlay/source/ns-3.24.1/src/tia921/helper/switch-helper.cc
overlay/source/ns-3.24.1/src/tia921/helper/switch-helper.h

The switch helper code implements a subclass of the existing BridgeHelper class that has the additional convenience method for installing the L2 forwarding behaviour on a switch.

overlay/source/ns-3.24.1/src/tia921/model/prio-tag.cc
overlay/source/ns-3.24.1/src/tia921/model/prio-tag.h
overlay/source/ns-3.24.1/src/tia921/model/seq-ts-tag.cc
overlay/source/ns-3.24.1/src/tia921/model/seq-ts-tag.h

The priority tag and sequence timestamp tag code implement priority tags that are used by the Ethernet priority queue, and sequence-timestamp tags that are used by the PCAP sender and receivers.

overlay/source/ns-3-dce/myscripts/core2lan/core2lan.cc
overlay/source/ns-3-dce/myscripts/core2lan/setup-pcap.cc
overlay/source/ns-3-dce/myscripts/core2lan/setup-pcap.h
overlay/source/ns-3-dce/myscripts/core2lan/sim-param.cc
overlay/source/ns-3-dce/myscripts/core2lan/sim-param.h

The CORE2LAN code, along with the setup-pcap helper, and sim-param helper implement the top level of the simulation model.

The process of overlaying is simply one of creating many soft links. The following is an excerpt of the first few lines that are printed during the overlay phase.

```
+ OVERLAYDIR=../overlay
+ xargs mkdir -p
+ grep -v '[.]svn'
+ cd ../overlay
+ find source -type d
+ read FILEPATH
+ grep -v '[.]svn'
+ cd ../overlay
+ find source -type f
++ dirname source/ns-3.24.1/src/tia921/helper/dash-server-helper.h
+ DIRNAME=source/ns-3.24.1/src/tia921/helper
++ sed -e 's:[^/]\+:..:g'
++ echo source/ns-3.24.1/src/tia921/helper
+ LINKPATH=../../../../../../../overlay/source/ns-3.24.1/src/tia921/helper/dash-server-helper.h
+   ln   -s   -f   ../../../../../../../overlay/source/ns-3.24.1/src/tia921/helper/dash-server-helper.h
source/ns-3.24.1/src/tia921/helper
[...]
+ LINKPATH=../../../overlay/source/ns-3-dce/run_core2lan.sh
+ ln -s -f ../../../overlay/source/ns-3-dce/run_core2lan.sh source/ns-3-dce
+ read FILEPATH
+ ln -s myscripts/core2lan/tc source/ns-3-dce/tc
+ ln -s myscripts/core2lan/pcap source/ns-3-dce/pcap
```

A.7 Build phase

During the build phase, the source code for the modified base simulator and the ITU-T G.1050-2016 additional files are compiled.

```
>> Building iperf - OK
>> Building net-next-sim-2.6.36 - OK
>> Building wget - OK
>> Building pybindgen-0.17.0.876 - OK
>> Building thttpd - OK
>> Building bash - OK
>> Building ccnx - OK
>> Building iproute-2.6.38-fix-01 - OK
>> Building elf-loader - OK
>> Building netanim-3.105 - OK
>> Building ns-3.24.1 - OK
>> Building dce-linux-1.4 - OK
```

A.8 The CORE2LAN model

After the setup.sh script is run and the simulator is compiled, a number of new directories are created. The important ones are under ./dce/source:

./dce/source/iperf	The iPerf TCP performance tester
./dce/source/net-next-sim-2.6.36	The Linux operating system
./dce/source/ns-3.24.1	The ns-3 source code
./dce/source/ns-3-dce	The DCE add-on for ns-3

Because the ITU-T G.1050 model uses DCE, it is added to the DCE subdirectory. Within the DCE directory, there are several new files and subdirectories:

myscripts/core2lan/	The core2lan test case
pcap@	The PCAP input files
run_core2lan.sh@	A script to run a test case
tc@	The test case files
out/	The simulator output files

A.9 Simulator input data

Input to the simulator is in the form of PCAP files. These files are driven into the simulation as specified by the timestamps in PCAP files. The payload of the PCAP files, if present, is carried along with the packets in the simulation and placed in the output files. The simulation parameter settings can adjust the timing of the packets driven into the simulation, for example, to speed up or slow down the playback, or to smooth out unintended burstiness. A full list of the adjustable parameters for the PCAP packet generator is given below.

The parameters for running the simulation are given in a TCL file. The advantage of using TCL as a parameter file format is that it is a well-known file format, and it is extensible so that users can further customize the behaviour of the simulation to suit their particular needs. It also allows reuse of the test case parameter files from the previous revision of this Recommendation. An example of how this helps is that it gives a convenient way to comment out portions of code and to print messages to the screen for information or debugging.

At the very top of a parameter file, the user must include the common support routines from the file tc/common.tcl.

```
source "tc/common.tcl"
```

The first section of the parameter file consists of variable settings using the TCL "set" command. It is no different from setting a normal variable in TCL, except that the names of the variables are specific to the simulator. The example below sets the simulation length to 10 minutes (600 seconds)

```
# - - - - - - - - - - - - - - - - - -
# Simulation length
# - - - - - - - - - - - - - - - - - -
set SimLengthSeconds [expr 10*60]
```

In most test cases, the above line is commented out so that a default simulation length from common.tcl can control the duration of the simulation.

Note also in this example that a TCL expression computes the number of seconds in ten minutes. While it is also possible to specify 600 seconds, using an expression in this way helps to make clear what is intended (10 minutes) and also serves as an example of how to use TCL expressions in the parameter file.

Next, set the access and core network parameters (this example is from test case Dp4):

```
# - - - - - - - - - - - - - - - - - -
# Network Parameters
# - - - - - - - - - - - - - - - - - -
Set_Access_Params dsl_p4_u4
Set_Core_Params p4_u4
```

The first line above sets parameters for the access network. The second line above sets parameters for the core network. The arguments to these two procedures is a text string that refers to a pre-defined combination of parameters that are given in common.tcl. Custom test cases can be created by setting the individual simulator control variables at this level.

Table A.1 shows the names of the predefined access parameters, and the corresponding access parameters:

Table A.1 – Predefined access parameters

Tech Param Name	Node	SpeedDn	SpeedUp	BER_Fwd	BER_Rev	LatcyDn	LatcyUp
dsl_w1	RACC	3.30E+07	3.00E+06	1.00E-08	1.00E-08	1.00E-03	1.00E-03
dsl_w2	RACC	3.30E+07	3.00E+06	1.00E-08	1.00E-08	1.00E-03	1.00E-03
dsl_w3	RACC	3.30E+07	3.00E+06	1.00E-08	1.00E-08	1.00E-03	1.00E-03
dsl_w4	RACC	2.20E+07	2.00E+06	1.00E-08	1.00E-08	1.00E-03	1.00E-03
dsl_w5	RACC	1.60E+07	1.00E+06	1.00E-07	1.00E-07	1.00E-03	1.00E-03
dsl_w6	RACC	1.60E+07	1.00E+06	1.00E-07	1.00E-07	1.00E-03	1.00E-03
dsl_w7	RACC	1.60E+07	1.00E+06	1.00E-07	1.00E-07	1.00E-03	1.00E-03
dsl_w8	RACC	1.00E+07	1.00E+06	1.00E-07	1.00E-07	1.00E-03	1.00E-03
pon_w1	RACC	5.00E+07	2.50E+07	1.00E-12	1.00E-12	1.00E-03	1.00E-03
pon_w2	RACC	5.00E+07	2.50E+07	1.00E-12	1.00E-12	1.00E-03	1.00E-03
pon_w3	RACC	3.50E+07	2.50E+07	1.00E-11	1.00E-11	1.00E-03	1.00E-03
pon_w4	RACC	3.50E+07	2.50E+07	1.00E-11	1.00E-11	1.00E-03	1.00E-03
pon_w5	RACC	2.50E+07	2.50E+07	1.00E-10	1.00E-10	1.00E-03	1.00E-03
pon_w6	RACC	2.50E+07	2.50E+07	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_w7	RACC	2.50E+07	2.50E+07	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_w8	RACC	1.50E+07	5.00E+06	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_p1_u1	RACC	3.50E+07	3.50E+07	1.00E-12	1.00E-12	1.00E-03	1.00E-03
pon_p2_u2	RACC	3.50E+07	1.50E+07	1.00E-12	1.00E-12	1.00E-03	1.00E-03
pon_p3_u3	RACC	2.50E+07	2.50E+07	1.00E-11	1.00E-11	1.00E-03	1.00E-03
pon_p4_u4	RACC	1.50E+07	5.00E+06	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_p5_u5	RACC	5.00E+06	2.00E+06	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_p6_u6	RACC	1.50E+07	5.00E+06	1.00E-09	1.00E-09	1.00E-03	1.00E-03
pon_p7_u7	RACC	5.00E+06	2.00E+06	1.00E-09	1.00E-09	1.00E-03	1.00E-03
dsl_p1_u1	RACC	2.40E+07	3.00E+06	1.00E-08	1.00E-08	1.00E-03	1.00E-03
dsl_p2_u2	RACC	1.80E+07	1.50E+06	1.00E-07	1.00E-07	1.00E-03	1.00E-03
dsl_p3_u3	RACC	1.20E+07	1.50E+06	1.00E-06	1.00E-06	1.00E-03	1.00E-03
dsl_p4_u4	RACC	6.00E+06	1.00E+06	1.00E-06	1.00E-06	1.00E-03	1.00E-03
dsl_p5_u5	RACC	3.00E+06	1.00E+06	1.00E-06	1.00E-06	1.00E-03	1.00E-03
dsl_p6_u6	RACC	6.00E+06	1.00E+06	1.00E-06	1.00E-06	1.00E-03	1.00E-03
dsl_p7_u7	RACC	3.00E+06	1.00E+06	1.00E-06	1.00E-06	1.00E-03	1.00E-03

Table A.2 shows the names of the predefined access parameters, and the corresponding access parameters:

Table A.2 – Predefined core parameters

Name	NumCoreSW	Latency	Speed
w1	3	10.0E-3	1.00E+09
w2	4	25.0E-3	1.00E+09
w3	5	50.0E-3	1.00E+09
w4	6	75.0E-3	1.00E+09
w5	7	85.0E-3	1.00E+09
w6	8	100.0E-3	1.00E+09
w7	9	125.0E-3	1.00E+09
w8	10	150.0E-3	1.00E+09
p1_u1	3	10.0E-3	1.00E+09
p2_u2	4	25.0E-3	1.00E+09
p3_u3	5	50.0E-3	1.00E+09
p4_u4	8	75.0E-3	1.00E+09
p5_u5	10	100.0E-3	1.00E+09
p6_u6	12	200.0E-3	1.00E+09
p7_u7	15	250.0E-3	1.00E+09

The next section of the parameter file specifies the input and output packet streams. PacketGenerators read input PCAP files and drive them into the simulation. PacketMonitors receive packets from elements within the simulation and save them in PCAP format, and also save delay and packet loss information.

An example of this for test case Dp4-t is shown below:

```
# ================================================================================================================================
#                                  Managed Bandwidth PCAP Stream Generators and Monitors
#
#                Name                File              Start         BW   PPM Rand    Prio  Repeat   PIR PBS
# --------------------------------------------------------------------------------------------------------------------------------
PCAP_Generator   PktGenPCAPFwdI4     SD_cbr              5.0        0.260 0.0  0.0       2     -1            ;# 2.0Mb/s
PCAP_Generator   PktGenPCAPFwdI5     SD_vbr              5.0        0.361 0.0  0.0       2     -1            ;# 2.0Mb/s
PCAP_Generator   PktGenPCAPFwdI1     voip_g729_fwd       2.0          1.0 0.0  0.0       1     -1
PCAP_Generator   PktGenPCAPRevI1     voip_g729_rev       2.0          1.0 0.0  0.0       1     -1

PCAP_Monitor     PktMonPCAPFwdI1     VoIP1_Fwd
PCAP_Monitor     PktMonPCAPFwdI4     SDTV1_Down
PCAP_Monitor     PktMonPCAPFwdI5     SDTV2_Down
PCAP_Monitor     PktMonPCAPRevI1     VoIP1_Rev
```

The parameters for the input PCAP generators are specified in the columns as shown above. The meanings of the columns are described briefly below.

– **Instance name (Name)** – the instance name of the generator. This depends on the specific top level simulation file.

– **Filename (file)** – the file name of the input PCAP file. Assumes that the PCAP file is in the ./pcap subdirectory and has a .pcap suffix so neither of these needs to be specified in the parameter file.

– **Start delay (start)** – the relative delay, in seconds, before starting the generator.

– **Bandwidth scale factor (BW)** – a scale factor that divides the relative timestamps in the PCAP file. A value of 1.0 makes no change to the timing of the packets. A value of 2.0 causes the packets to be transmitted twice as fast by dividing the timestamps in the PCAP file by 2.0.

– **PPM offset (PPM)** – this is similar to the bandwidth scale factor, but is expressed in units of parts per million. A value of 0 ppm makes no change to the rate at which packets are generated. A value of 100 ppm reduces the timestamps in the PCAP file by 100 ppm, which causes the file to be transmitted 100 ppm faster than the nominal rate.
– **Randomness (Rand)** – this is a value from 0.0 to 1.0 that adds a percentage of randomness to the inter-packet times. A value of 0.5 represents 50% randomness for inter-packet timing. For example, if the time between two consecutive packets in the PCAP file is 1 millisecond, and the randomness is 0.5, then the actual time between those two consecutive packets could be in the range 500 microsecond to 1.5 millisecond.
– **Priority (Prio)** – this is the assigned priority of the packets.
– **Repeat count (Repeat)** – this is the number of times to repeat the PCAP file. A value of 0 or 1 means to play the file once, a value of two or more will play the file that many times, and a value of –1 will repeat the file forever.
– **Shaper PIR (PIR) and shaper PBS (PBS)** – these parameters control a shaper that is built into the PacketGeneratorPCAP object. The PIR is the peak information rate and is expressed in bits per second, so the value 10E6 represents 10 million bits per second. The PBS is the peak burst size and is expressed in bits. Normally the PIR should be set safely above (e.g., 2.5x) the bit rate of a VBR video stream, unless the goal is to smooth out microbursts.

The PacketMonitor outputs PCAP files and ".out" files. There are only two parameters for the PCAP monitor command:

– **Instance name** – the instance name of the monitor. This depends on the specific top level simulation file.
– **File name** – the base name of the output files. The output files will be placed into the same directory as the parameter file and will have suffixes .pcap and .out.

Note that if a PCAP_Monitor line is commented out or not present, the output files will not be written. This is helpful in situations where only a subset of the output files is needed so that disk space can be saved.

The simulation provides the following global variables for use by the TCL code in the parameter file:

– **$paramfile** – this is the name of the parameter file;
– **$paramdir** – this is the directory name of the parameter file;
– **$outdir** – this is the output directory name, and is set based on $paramfile in common.tcl.

A.10 Running a simulation case with a convenience script

To run one test case, there is a convenience script called run_core2lan.ss. An example of running the test case "Du6-t" is shown below:

```
./run_core2lan.sh Du6-t --pcapUp=1 --pcapDown=1 --iperfDown --verbose=2 --
capLen=70
```

A.11 Simulator output

All output files for test case *tcname* are placed in the ./out/*tcname* subdirectory. By placing the output results for each test case in their own directory, it is easier to keep them organized. By placing the outputs in a separate directory from the input parameter settings, the output results can be easily cleaned up, and it is easier to manage the parameter files.

As described in the section above on the parameter file, it is not necessary to save output for all of the streams being tested. This can be helpful in conserving disk space, as the output files can be quite large.

A.11.1 Output for UDP flows

For the managed UDP streams, such as IPTV and VoIP, the simulator output is in the form of PCAP files and ".out" files. The PCAP files record the managed stream output packets along with their timestamps. Comparisons of input PCAP and output PCAP files can be performed to determine the effect of the impairment on, for example, video quality.

The ".out" output files contain delay and packet loss information (in ASCII CSV format) about the simulation result. This information is also implicitly contained in the PCAP file, but because it is not possible to directly indicate packet delay or loss in a PCAP file, the ".out" files are helpful for post analysis. For example, because the ".out" files are in CSV format, they can be read into a spreadsheet program for analysis.

The columns contained in the .out file reflect the time the packet was transmitted into the network, the delay experienced by the packet and a flag indicating whether the packet was dropped by the network. Time is represented as the number of seconds since the beginning of the simulation and has nanosecond resolution (nine digits after the decimal point). Packet delay is represented by milliseconds and has nanosecond resolution (6 digits after the decimal point). Drop is a Boolean flag represented by a 0 or a 1 (1 means a packet was dropped, 0 means successful reception).

A.11.2 Output for TCP flows

The pcap files for the TCP flows generated by the iPerf client and server are analysed after the simulations finish. There is a pcap file captured at the server port, and a pcap file captured at the client port. That way, every packet at the server can be matched with the corresponding packet at the client so that one-way network delay can be calculated for each packet in each direction.

The post analysis is run by the top level script "analysis/postall." It processes the pcap files for each test case with a helper script called "postz." The postz script uses the "tstat" "tshark" and "bw" utilities to extract information from the pcap files. The result of the postz script is four files:

– iperf-[up|down]-client.csv;

– iperf-[up|down]-server.csv;

– bw-[up|down].csv;

– del-[up|down].csv.

NOTE – for upstream iPerf, select "up" and for downstream iPerf, select "down".

After postz is run on all of the test cases, "postall" then uses the "merge.pl" script to create an "R" language script to create summary plots where results for all test cases are merged onto a single page.

A.12 Plotting results

The results are plotted by the top level script "analysis/plotall." This script uses the "R" statistical language environment to make plots using the scripts created in the post analysis phase. The script also uses plotting utilities (based on gnuplot and perl) from an earlier revision of this Recommendation to create plots of stream flow latency and packet loss. The "R" based plotting utilities are used for the TCP streams, while the gnuplot utilities are used for the UDP streams.

A.13 PCAP file list

The simulator input PCAP files are in the dce/source/ns-3-dce/pcap/ subdirectory. Since these files are large, they are typically distributed separately from the model source code, and often the pcap subdirectory is a soft-link to a separate directory. Here is a list of typical input PCAP files.

```
pcap/HD_cbr.pcap
pcap/HD_vbr.pcap
pcap/SD_cbr.pcap
pcap/SD_vbr.pcap
```

```
pcap/voip_g729_fwd.pcap
pcap/voip_g729_rev.pcap
pcap/ipfax_v34_t38_fwd.pcap
pcap/ipfax_v34_t38_rev.pcap
```

A.14 Test case file list

The individual test case parameter files are in the sim/tc directory:

```
./tc/Dp1-t.param
./tc/Dp2-t.param
./tc/Dp3-t.param
./tc/Dp4-t.param
./tc/Dp5-t.param
./tc/Dp6-t.param
./tc/Dp7-t.param
./tc/Du1-t.param
./tc/Du2-t.param
./tc/Du3-t.param
./tc/Du4-t.param
./tc/Du5-t.param
./tc/Du6-t.param
./tc/Du7-t.param
./tc/Dw1-t.param
./tc/Dw2-t.param
./tc/Dw3-t.param
./tc/Dw4-t.param
./tc/Dw5-t.param
./tc/Dw6-t.param
./tc/Dw7-t.param
./tc/Dw8-t.param
./tc/Gp1-t.param
./tc/Gp2-t.param
./tc/Gp3-t.param
./tc/Gp4-t.param
./tc/Gp5-t.param
./tc/Gp6-t.param
./tc/Gp7-t.param
./tc/Gu1-t.param
./tc/Gu2-t.param
./tc/Gu3-t.param
./tc/Gu4-t.param
./tc/Gu5-t.param
./tc/Gu6-t.param
./tc/Gu7-t.param
./tc/Gw1-t.param
./tc/Gw2-t.param
./tc/Gw3-t.param
./tc/Gw4-t.param
./tc/Gw5-t.param
./tc/Gw6-t.param
./tc/Gw7-t.param
./tc/Gw8-t.param
```

The output results for each test case are placed in a subdirectory of the "./out" directory. For each PacketMonitorPCAP that is configured in the parameter file, there is an output PCAP file and an output "*.out*" file (described below).

```
./tc/common.tcl
```

There are four convenience scripts to help run the simulation and plot results:

```
run_core2lan.ss
runall
postall
plotall
```

A.15 Common TCL file

There is one TCL support file that helps with setting up simulation parameters:

```
./tc/common.tcl
```

The common TCL support file "common.tcl" helps simplify the process of setting variables for the PCAP generators and monitors.

The first thing that the common.tcl file does is create the output directory and set the variable $outdir:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Create the output directory
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
set outroot "out"
set outdir [file join $outroot [file rootname [file tail $paramfile]]]
file mkdir $outdir
```

The next routine "gset" is a general purpose routine for setting a global variable:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# gset -- set a variable at global scope
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc gset {name value} {
    upvar \#0 $name $name
    set $name $value
}
```

The next routine defines parameters for a PCAP generator:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Define parameters for a PCAP Generator
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc PCAP_Generator {name file {start ""} {bwscale ""} {ppm ""} \
                     {rand ""} {prio ""} {rept ""} {PIR ""} {PBS ""}} {
                          gset ${name}_FileName    "pcap/$file.pcap"
    if {$start   != ""} { gset ${name}_StartDelay  $start   }
    if {$bwscale != ""} { gset ${name}_BW_Scale    $bwscale }
    if {$ppm     != ""} { gset ${name}_PPM_Offset  $ppm     }
    if {$rand    != ""} { gset ${name}_Randomness  $rand    }
    if {$prio    != ""} { gset ${name}_Priority    $prio    }
    if {$rept    != ""} { gset ${name}_RepeatCnt   $rept    }
    if {$PIR     != ""} { gset ${name}_ShaperPIR   $PIR     }
    if {$PBS     != ""} { gset ${name}_ShaperPBS   $PBS     }
}
```

The next routine defines parameters for a PCAP monitor:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Define parameters for a PCAP Monitor
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc PCAP_Monitor {name file} {
    global outdir
    gset ${name}_OutFileName      "$outdir/$file.out"
    gset ${name}_PCAPFileName     "$outdir/$file.pcap"
}
```

The next routines help to set network parameters for the access link:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Set Global Vars for Access Link Parameters
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc _Access_Params {name
            {SpeedDown ""}
            {SpeedUp ""}
            {BER_Fwd ""}
            {BER_Rev ""}
            {LatencyDown ""}
            {LatencyUp ""}} {
    if {$SpeedDown   != ""} { gset ${name}_SpeedDown    $SpeedDown   }
    if {$SpeedUp     != ""} { gset ${name}_SpeedUp      $SpeedUp     }
    if {$BER_Fwd     != ""} { gset ${name}_BER_Fwd      $BER_Fwd     }
    if {$BER_Rev     != ""} { gset ${name}_BER_Rev      $BER_Rev     }
    if {$LatencyDown != ""} { gset ${name}_LatencyDown  $LatencyDown }
    if {$LatencyUp   != ""} { gset ${name}_LatencyUp    $LatencyUp   }
    if {$ReorderProb != ""} { gset RO_Fwd_Prob          $ReorderProb }
    if {$ReorderProb != ""} { gset RO_Rev_Prob          $ReorderProb }
}
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Select a set of access link parameters based on name
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc Set_Access_Params {paramset} {
                                    #            Node SpdDn  SpdUp BER_Fwd BER_Rev LatcyDn LatcyUp
       if {$paramset == "dsl_w1"}   { _Access_Params RACC  33e6   3e6   1e-8   1e-8   1e-3  1e-3
 } elseif {$paramset == "dsl_w2"}   { _Access_Params RACC  33e6   3e6   1e-8   1e-8   1e-3  1e-3
 } elseif {$paramset == "dsl_w3"}   { _Access_Params RACC  33e6   3e6   1e-8   1e-8   1e-3  1e-3
 } elseif {$paramset == "dsl_w4"}   { _Access_Params RACC  22e6   2e6   1e-8   1e-8   1e-3  1e-3
 } elseif {$paramset == "dsl_w5"}   { _Access_Params RACC  16e6   1e6   1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "dsl_w6"}   { _Access_Params RACC  16e6   1e6   1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "dsl_w7"}   { _Access_Params RACC  16e6   1e6   1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "dsl_w8"}   { _Access_Params RACC  10e6   1e6   1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "pon_w1"}   { _Access_Params RACC  50e6  25e6   1e-12  1e-12  1e-3  1e-3
 } elseif {$paramset == "pon_w2"}   { _Access_Params RACC  50e6  25e6   1e-12  1e-12  1e-3  1e-3
 } elseif {$paramset == "pon_w3"}   { _Access_Params RACC  35e6  25e6   1e-11  1e-11  1e-3  1e-3
 } elseif {$paramset == "pon_w4"}   { _Access_Params RACC  35e6  25e6   1e-11  1e-11  1e-3  1e-3
 } elseif {$paramset == "pon_w5"}   { _Access_Params RACC  25e6  25e6   1e-10  1e-10  1e-3  1e-3
 } elseif {$paramset == "pon_w6"}   { _Access_Params RACC  25e6  25e6   1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_w7"}   { _Access_Params RACC  25e6  25e6   1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_w8"}   { _Access_Params RACC  15e6  5e6    1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_p1_u1"} { _Access_Params RACC  35e6  35e6   1e-12  1e-12  1e-3  1e-3
 } elseif {$paramset == "pon_p2_u2"} { _Access_Params RACC  35e6  15e6   1e-12  1e-12  1e-3  1e-3
 } elseif {$paramset == "pon_p3_u3"} { _Access_Params RACC  25e6  25e6   1e-11  1e-11  1e-3  1e-3
 } elseif {$paramset == "pon_p4_u4"} { _Access_Params RACC  15e6  5e6    1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_p5_u5"} { _Access_Params RACC   5e6  2e6    1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_p6_u6"} { _Access_Params RACC  15e6  5e6    1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "pon_p7_u7"} { _Access_Params RACC   5e6  2e6    1e-9   1e-9   1e-3  1e-3
 } elseif {$paramset == "dsl_p1_u1"} { _Access_Params RACC  24e6  3.0e6  1e-8   1e-8   1e-3  1e-3
 } elseif {$paramset == "dsl_p2_u2"} { _Access_Params RACC  18e6  1.5e6  1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "dsl_p3_u3"} { _Access_Params RACC  12e6  1.5e6  1e-7   1e-7   1e-3  1e-3
 } elseif {$paramset == "dsl_p4_u4"} { _Access_Params RACC   6e6  1.0e6  1e-6   1e-6   1e-3  1e-3
 } elseif {$paramset == "dsl_p5_u5"} { _Access_Params RACC   3e6  1.0e6  1e-6   1e-6   1e-3  1e-3
 } elseif {$paramset == "dsl_p6_u6"} { _Access_Params RACC   6e6  1.0e6  1e-6   1e-6   1e-3  1e-3
 } elseif {$paramset == "dsl_p7_u7"} { _Access_Params RACC   3e6  1.0e6  1e-6   1e-6   1e-3  1e-3
 } elseif {$paramset == "core_only"} { _Access_Params RACC   1e9  1e9    1e-12  1e-12  0     0
 } else {
    error "ERROR: Invalid Net_Param name $paramset."
 }
}
```

The next routines help to set network parameters for the core:

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Set Global Vars for Access Link Parameters
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc _Core_Params {{numsw ""}
            {Latency ""}
            {Speed ""}} {
    if {$numsw     != ""} { gset NumCoreSW       $numsw   }
    if {$Latency   != ""} { gset CORE_Latency    $Latency   }
    if {$Speed     != ""} { gset CORE_Speed      $Speed     }
```

```
}
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Select a set of core parameters based on name
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
proc Set_Core_Params {paramset} {
                                        #       NumCoreSW Latency   Speed
          if {$paramset == "w1"}      { _Core_Params          3   10e-3    1e9
    } elseif {$paramset == "w2"}      { _Core_Params          4   25e-3    1e9
    } elseif {$paramset == "w3"}      { _Core_Params          5   50e-3    1e9
    } elseif {$paramset == "w4"}      { _Core_Params          6   75e-3    1e9
    } elseif {$paramset == "w5"}      { _Core_Params          7   85e-3    1e9
    } elseif {$paramset == "w6"}      { _Core_Params          8  100e-3    1e9
    } elseif {$paramset == "w7"}      { _Core_Params          9  125e-3    1e9
    } elseif {$paramset == "w8"}      { _Core_Params         10  150e-3    1e9
    } elseif {$paramset == "p1_u1"}   { _Core_Params          3   10e-3    1e9
    } elseif {$paramset == "p2_u2"}   { _Core_Params          4   25e-3    1e9
    } elseif {$paramset == "p3_u3"}   { _Core_Params          5   50e-3    1e9
    } elseif {$paramset == "p4_u4"}   { _Core_Params          8   75e-3    1e9
    } elseif {$paramset == "p5_u5"}   { _Core_Params         10  100e-3    1e9
    } elseif {$paramset == "p6_u6"}   { _Core_Params         12  200e-3    1e9
    } elseif {$paramset == "p7_u7"}   { _Core_Params         15  250e-3    1e9
    } elseif {$paramset == "core_only"} { _Core_Params        5  100e-3    1e9
    } else {
       error "ERROR: Invalid Net_Param name $paramset."
    }
}
# - - - - - - - - - - - - - - - - -
# Other Misc parameter settings
# - - - - - - - - - - - - - - - - -
set CORESW_BufSizeBytes       [expr 65*1518]
set DSLAM_BufSizeBytes        [expr 65*1518]
set Firewall_BufSizeBytes     [expr 65*1518]
set Modem_BufSizeBytes        [expr 65*1518]
set RLAN_Speed                100E6
# - - - - - - - - - - - - - - - - - -
# Set Default Simulation length
# - - - - - - - - - - - - - - - - - -
set SimLengthSeconds          [expr 10*60]
```

Annex B

C++ source code of the discrete event simulator

(This annex forms an integral part of this Recommendation.)

The C++ source code of the simulator is included in simulator.tar.gz of the electronic attachment (Annex H).

Annex C

Packet capture files of interfering traffic

(This annex forms an integral part of this Recommendation.)

Table C.1 lists the pcap files used in the standard test cases. These files are included in folder ./pcap in the electronic attachment (Annex H).

Table C.1 – PCAP files of interfering traffic

QoS	PCAP file	Data size	Packets	Duration	Avg Pkt size	Packet rate	Avg bit rate	5s peak	1s peak	Description
2	HD_cbr	275756340	201282	142	1370	1419	15555964	15558816	15574160	High definition IPTV flow, constant bit rate
2	HD_vbr	196918938	143737	142	1370	1014	11112049	13112544	18555280	High definition IPTV flow, variable bit rate
2	SD_cbr	274844350	200617	285	1370	703	7702130	7704880	7715840	Standard definition IPTV flow, constant bit rate
2	SD_vbr	197616080	144246	285	1370	507	5553410	6608880	8789920	Standard definition IPTV flow, variable bit rate
1	voip_g729_fwd	603120	7180	215	84	33	22406	22444	22848	[ITU-T G.729] voice-over-IP
1	voip_g729_rev	602784	7176	215	84	33	22403	22444	22848	

Figures C.1 to C.3 show the bit rate of each pcap stream averaged over 1-second and 5-second non-overlapping windows.

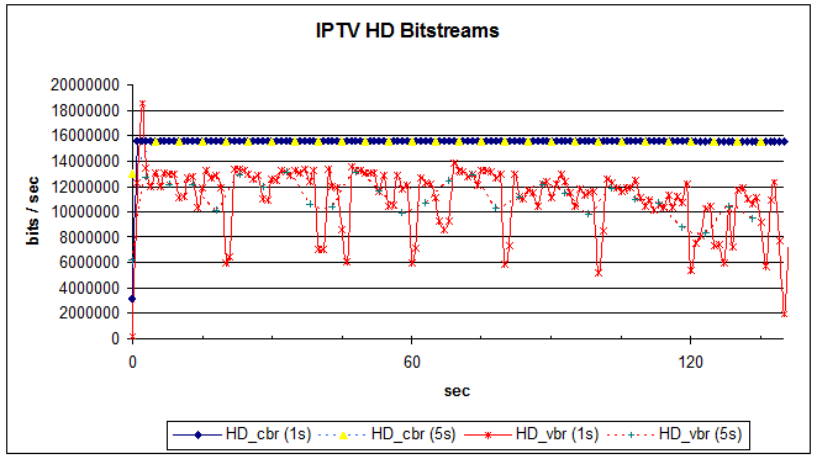


Figure C.1 – IPTV HD input PCAP bandwidth

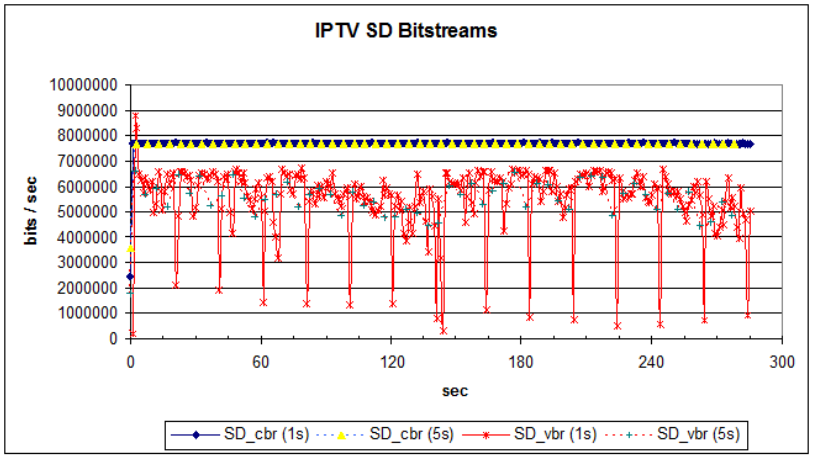


Figure C.2 – IPTV SD input PCAP bandwidth

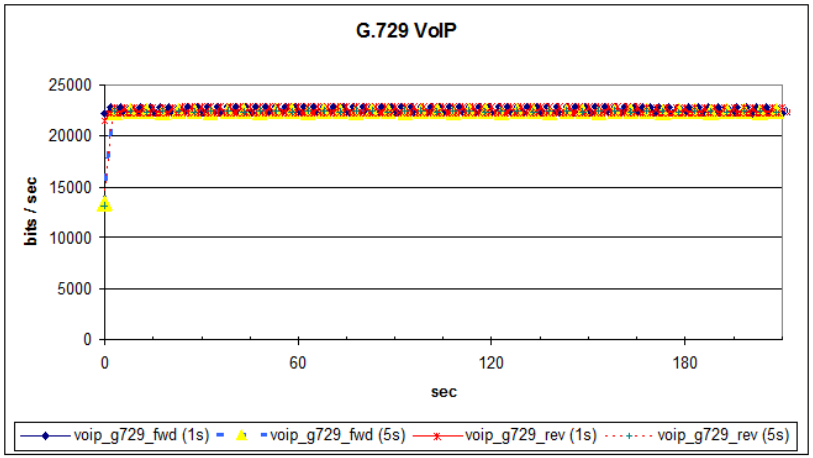


Figure C.3 – Voice over IP input PCAP bandwidth

Annex D

Example output plots for a typical test case

(This annex forms an integral part of this Recommendation.)

The baseline test cases of the previous edition of this Recommendation have been modified and run using the ns-3 simulator of Annex B. The goal of this simulator has been to more accurately characterize the residual best effort "goodput". To accomplish this, the simulator utilizes iPerf, a synthetic TCP load generation program, to replace the unmanaged best traffic of the original test cases.

The test cases are executed in three scenarios:

a) Down: The iPerf flow is run in the downstream direction and directly competes with managed traffic.

b) Up: The iPerf flow is run unencumbered in the upstream direction. While the data stream has no competition, the acknowledgement packets do compete since they must traverse the downstream path.

c) Bidi: iPerf flows are present in both the upstream and downstream directions.

This annex presents the plots that are available for each test case in the electronic attachment. For the purposes of this example, the plots are presented for the bidirectional scenario of the Dw4-t test case.

The generated plots fall into three categories:

– Overall flow summaries: Violin plots (Figures D.1.1 to D.1.8) allow for comparison of the distribution of one-way delay for the individual flows. Further plots are also provided for specific comparisons between the VoIP, DTV, and TCP (iPerf) flows. Bandwidth plots are also provided for the different flows in the test case with 1sec and 10sec bandwidth averaging to help visualize the burstiness of the traffic. These charts are also presented as line charts and stacked area charts to help visualize each stream's bandwidth contributions to the overall traffic mix.

– Legacy plots for unidirectional managed flows: This series of plots (Figures D.2.1 to D.2.10) were the same ones presented in the previous revision of this Recommendation. These plots capture the packet delay, drop, and distributions for the managed flows of the test case.

– Unmanaged TCP flow (iPerf) summaries: The TCP flows are characterized by a series of plots (Figures D.3.1 to D.3.7) showing the bandwidth, delay, and round-trip-time (RTT) of the flow.

D.1 Overall Flow Summaries

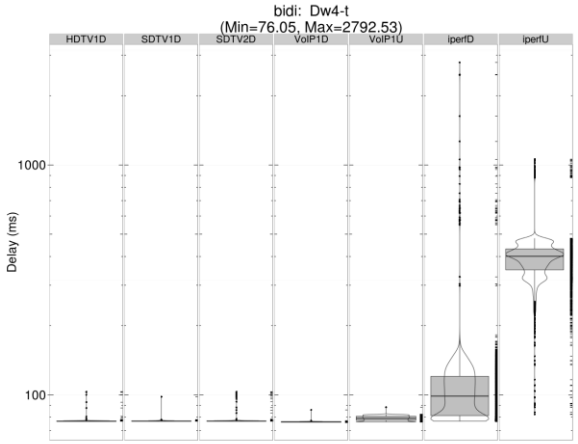


Figure D.1.1 – Delay distributions for all flows

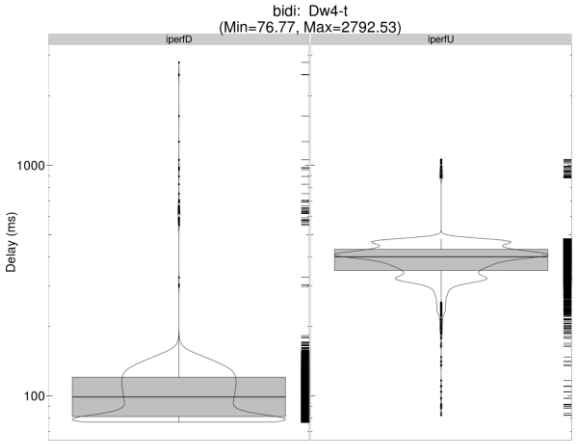


Figure D.1.2 – Delay distributions for iPerf flows

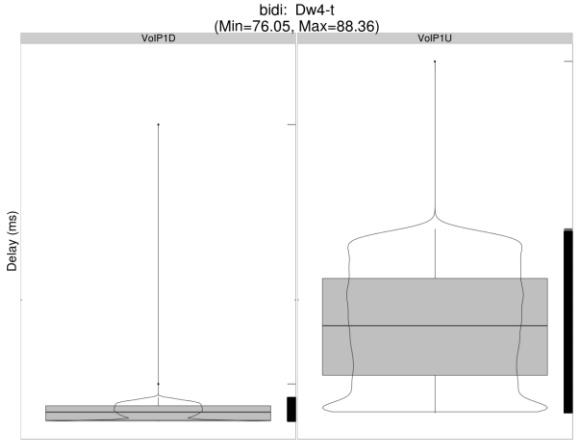


Figure D.1.3 – Delay distributions for VoIP flows

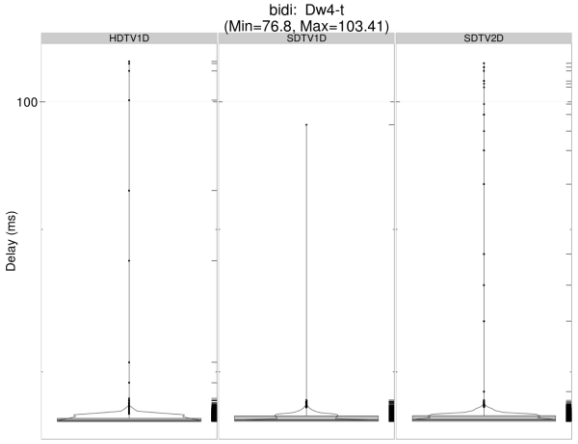


Figure D.1.4 – Delay distributions for DTV flows (if present)

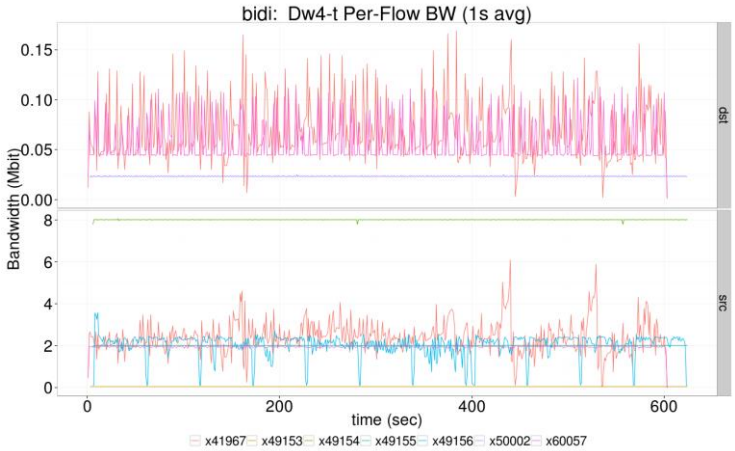


Figure D.1.5 – Bandwidth per flow/port (1s avg)

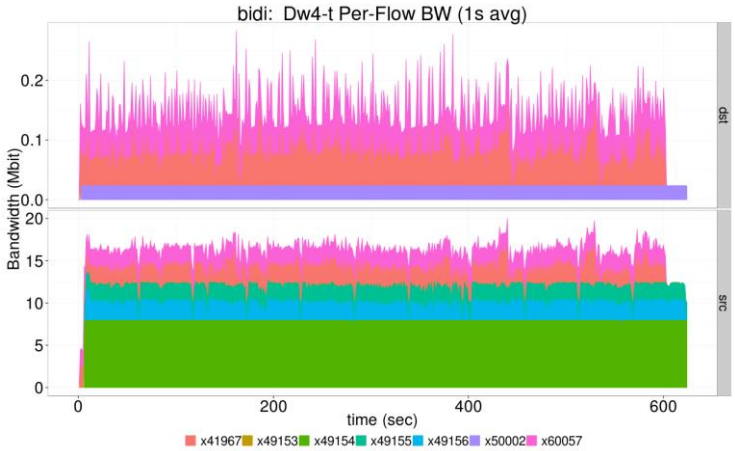


Figure D.1.6 – Bandwidth per flow/port stacked (1s avg)

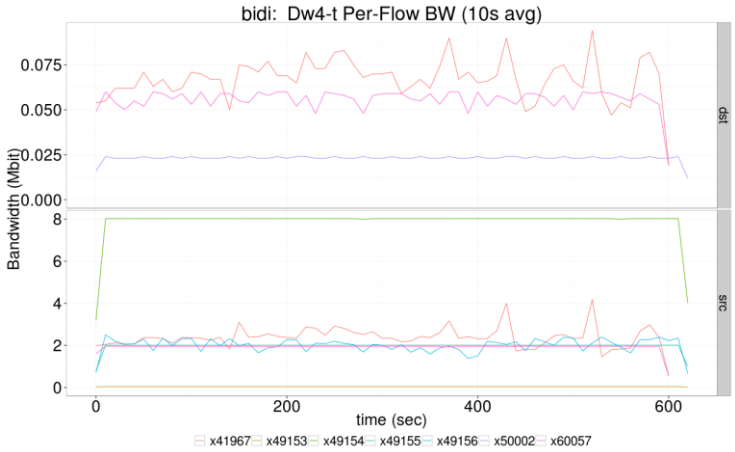


Figure D.1.7 – Bandwidth per flow/port (10s avg)

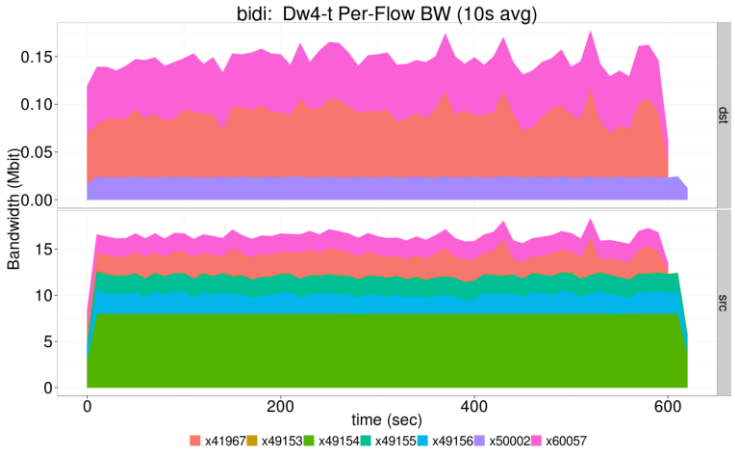


Figure D.1.8 – Bandwidth per flow/port stacked (10s avg)

D.2 Legacy plots for unidirectional managed flows (where present)

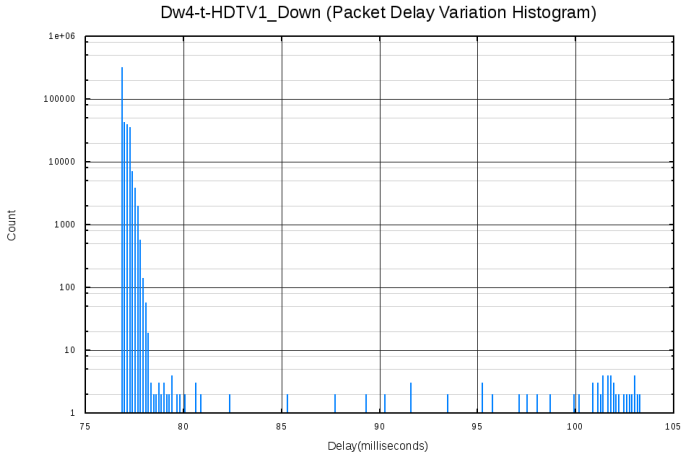


Figure D.2.1 – Downstream DTV flow packet delay variation

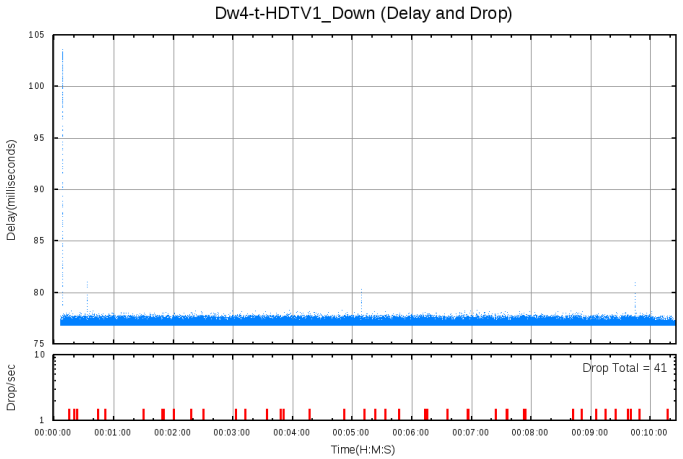


Figure D.2.2 – Downstream DTV flow packet delay-loss plot

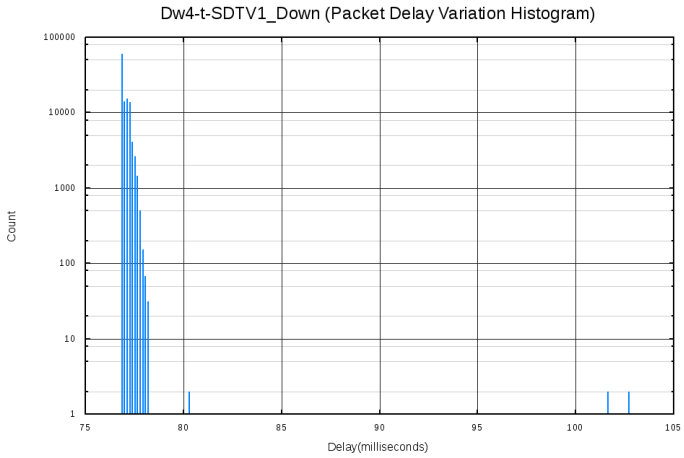


Figure D.2.3 – Downstream DTV flow packet delay variation

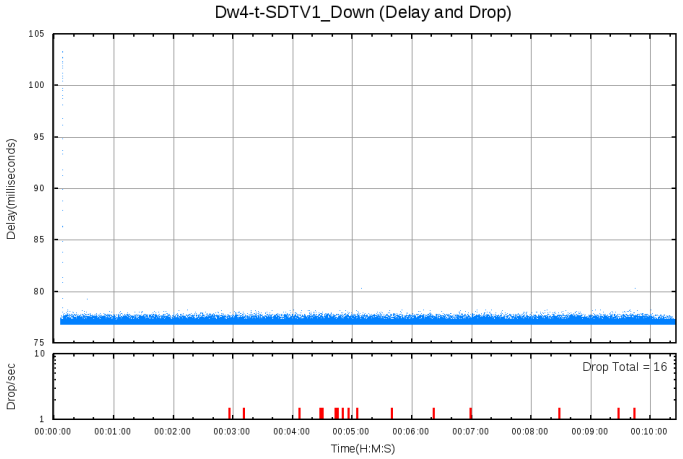


Figure D.2.4 – Downstream DTV flow packet delay-loss plot

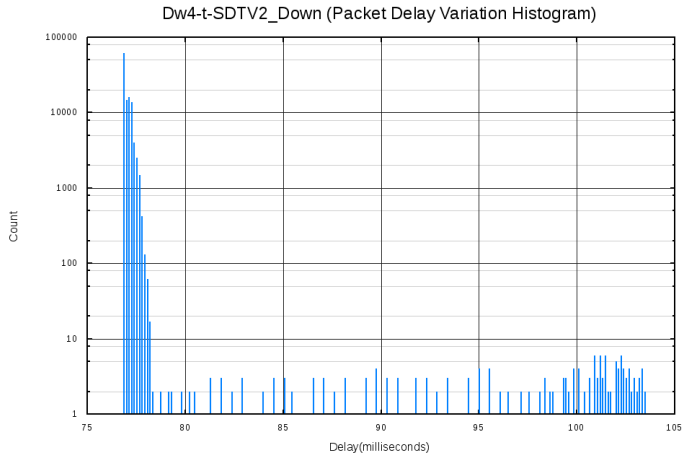


Figure D.2.5 – Downstream DTV flow packet delay variation

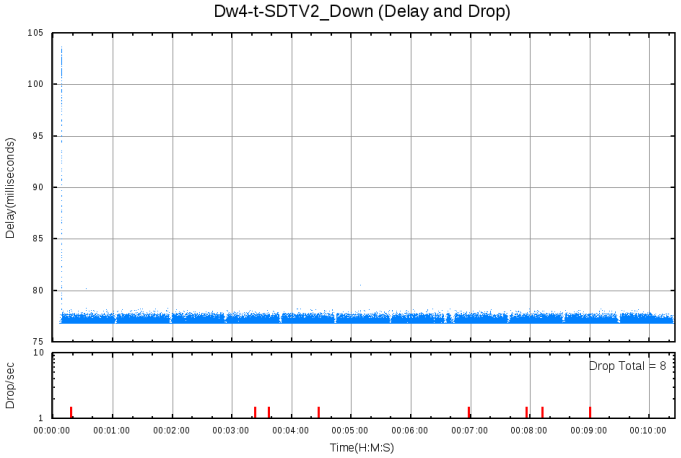


Figure D.2.6 – Downstream DTV flow packet delay-loss plot

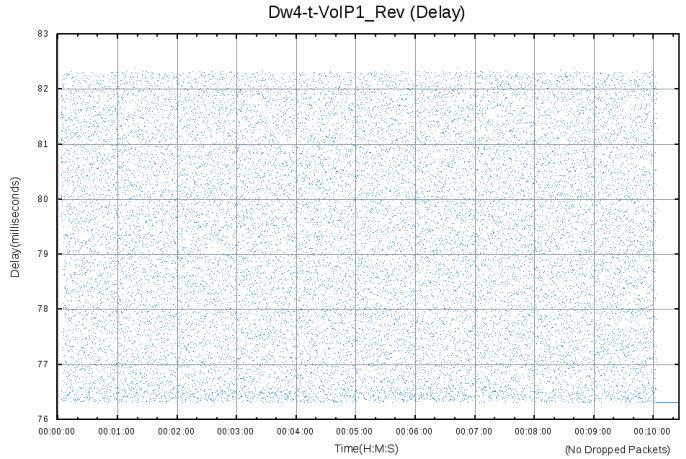


Figure D.2.7 – Downstream VoiP flow packet delay variation

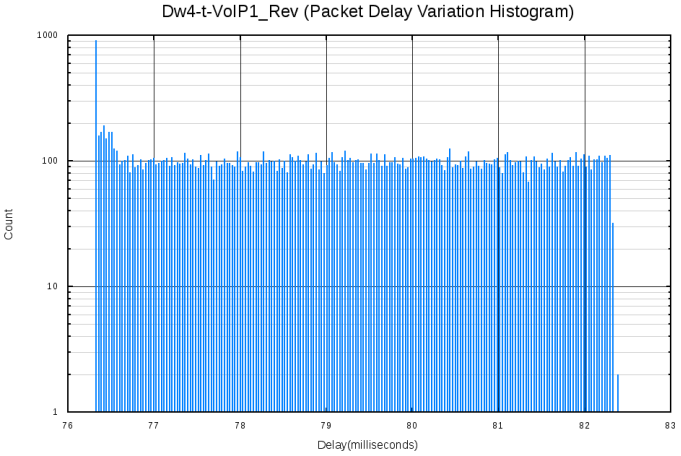


Figure D.2.8 – Downstream VoIP flow packet delay-loss plot

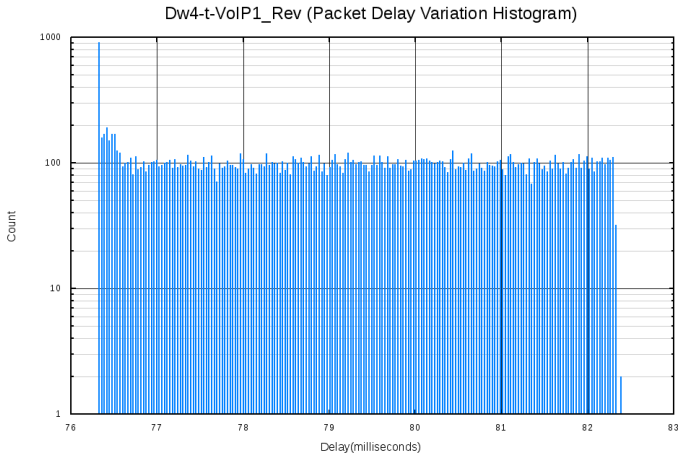


Figure D.2.9 – Upstream VoiP flow packet delay variation

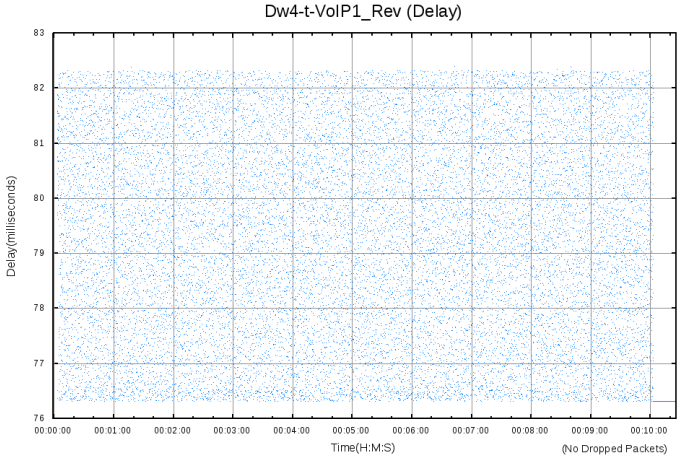


Figure D.2.10 – Upstream VoIP flow packet delay-loss plot

D.3 Unmanaged TCP flow (iPerf) summaries

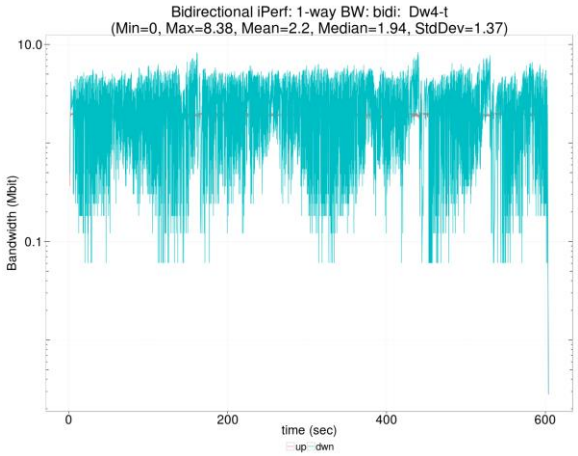


Figure D.3.1 – TCP bandwidth/delay/RTT

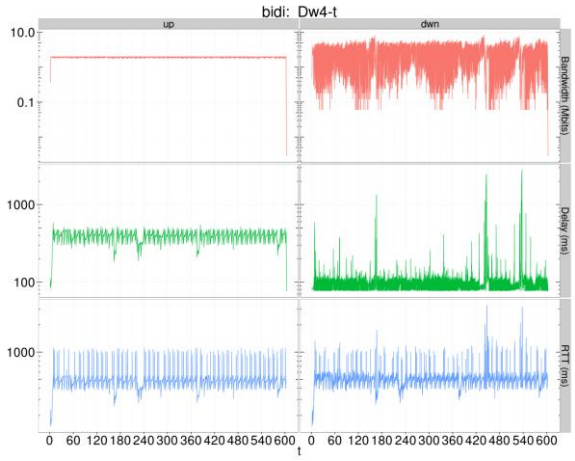


Figure D.3.2 – TCP bandwidth/delay/RTT (upstream/downstream for bidirectional case)

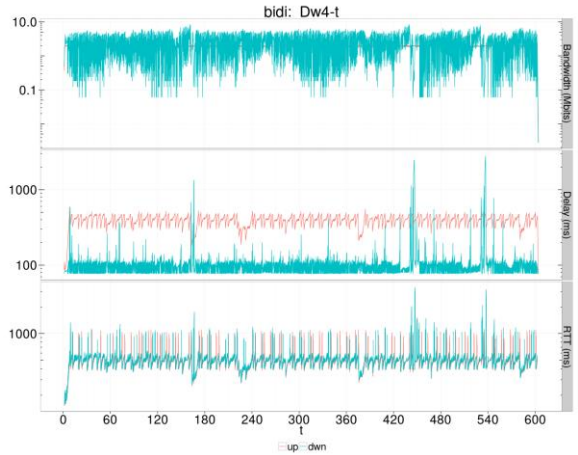


Figure D.3.3 – iPerf bandwidth

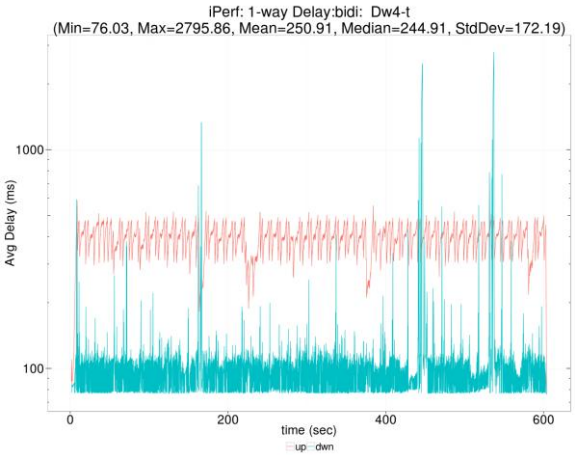


Figure D.3.4 – iPerf delay

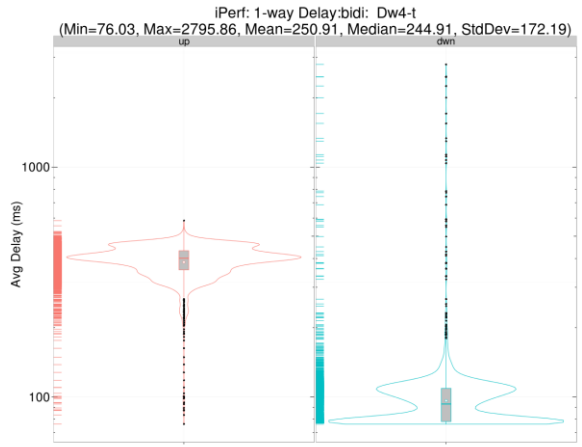


Figure D.3.5 – iPerf delay distribution (violin plot)

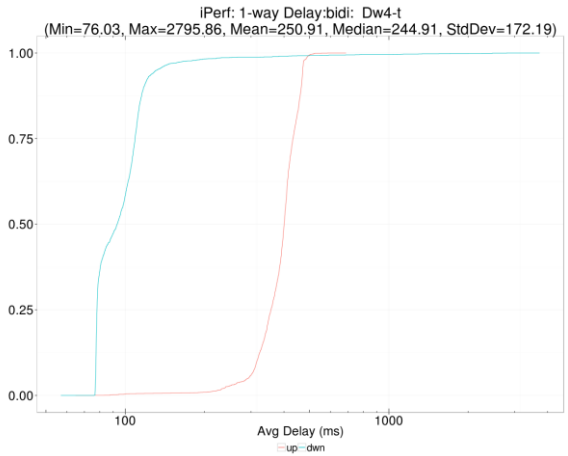


Figure D.3.6 – iPerf delay CDF

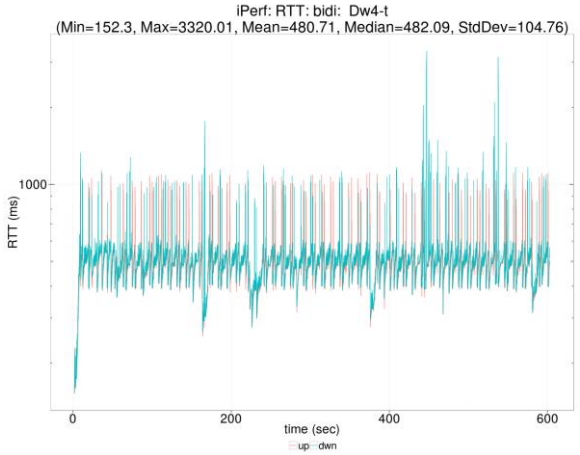


Figure D.3.7 – iPerf delay RTT

Annex E

Summary of simulation output plots

(This annex forms an integral part of this Recommendation.)

This annex describes a series of summary plots provided for the six technology-scenario combinations generated in this Recommendation. The purpose of these plots is to provide a quick reference to aid in the selection of specific test cases of interest. The plots are provided in their entirety in this annex as well as in the electronic attachment. The naming convention of plot files is as follows:

– $TECH – Technology (D=DSL, G=GPON);

– $SC – iPerf Scenario: (out.bidi, out.dwn, out.up).

Table E.1 – Summary output plot file naming convention

File Name	Description
/$TECH/$SC	Technology-scenario directory
summary.png	Test case median delay-bandwidth points
cdfplt-01.png	One-way iPerf delay CDF by direction and management type
merge-01.png	One-way iPerf bandwidth by direction and management type
merge-02.png	One-way iPerf bandwidth (unmanaged cases)
merge-03.png	One-way iPerf bandwidth (partly managed cases)
merge-04.png	One-way iPerf bandwidth (well managed cases)
merge-05.png	One-way iPerf delay distribution (unmanaged cases)
merge-06.png	One-way iPerf delay distribution (well managed cases)
merge-07.png	One-way iPerf delay distribution (partly managed cases)
mngplt-01.png	Aggregate managed bandwidth (well and partly managed cases)
/$TECH/$SC/PDF	Directory containing .pdf versions of test case plots
/$TECH/$SC/PCAP	Directory containing .pcap files of test case traffic (optional)
/$TECH/$SC/cdf.csv	Data file for cdfplt-01.png
/$TECH/$SC/stats.csv	Data file for summary.png
/$TECH/info.csv	Data file for Annex G Tables

E.1 DSL technology

E.1.1 DSL bidirectional iPerf scenario

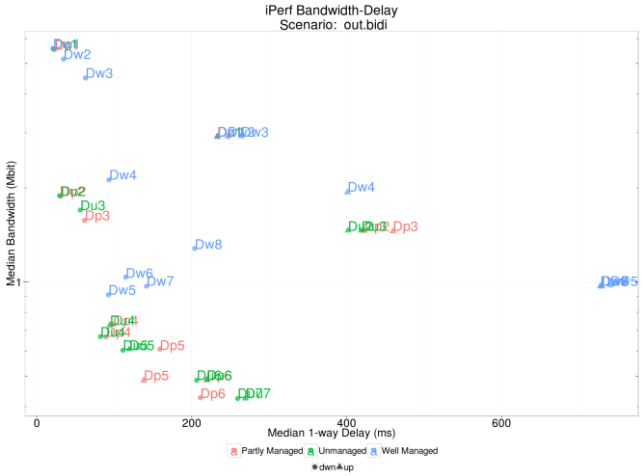


Figure E.1.1.1 – Test case median delay-bandwidth points

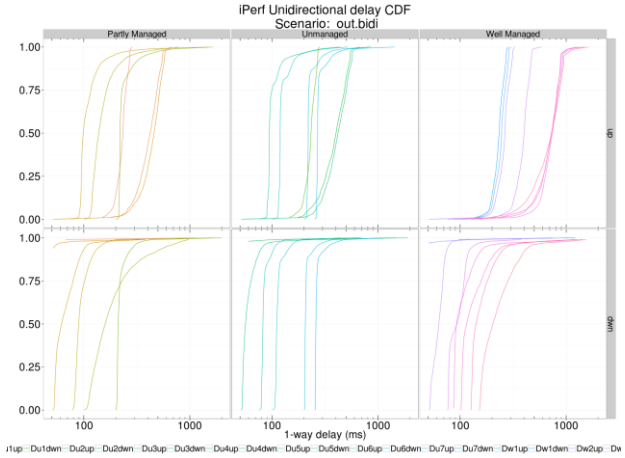


Figure E.1.1.2 – One-way iPerf delay CDF by direction and management type

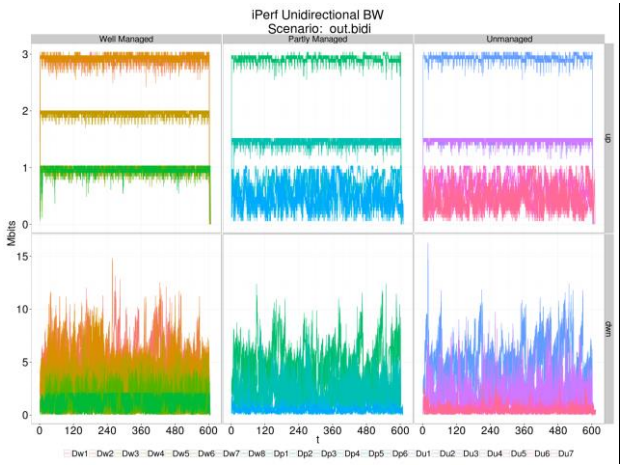


Figure E.1.1.3 – One-way iPerf bandwidth by direction and management type

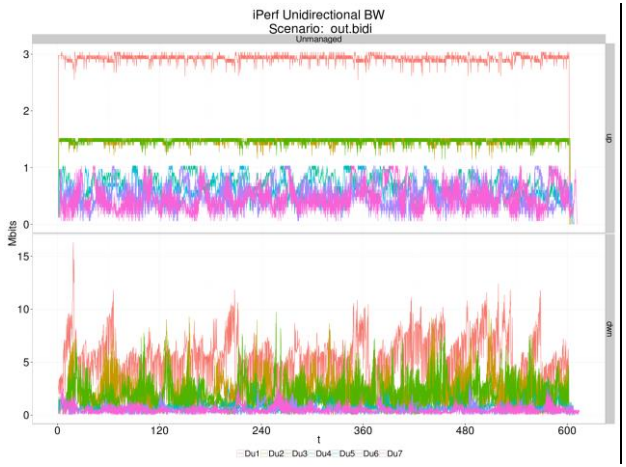


Figure E.1.1.4 – One-way iPerf bandwidth (unmanaged cases)

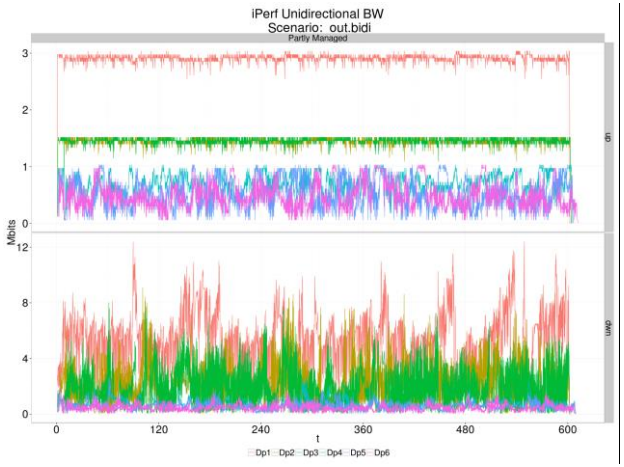


Figure E.1.1.5 – One-way iPerf bandwidth (partly managed cases)

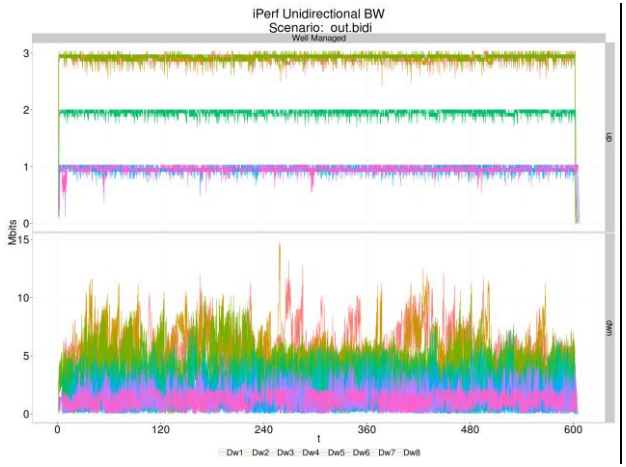


Figure E.1.1.6 – One-way iPerf bandwidth (well managed cases)

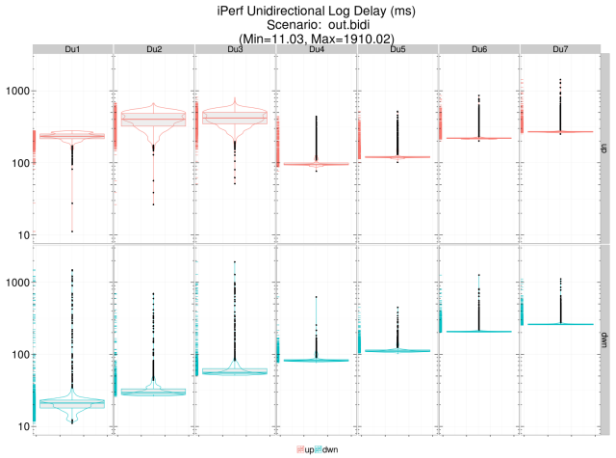


Figure E.1.1.7 – One-way iPerf delay distribution (unmanaged cases)

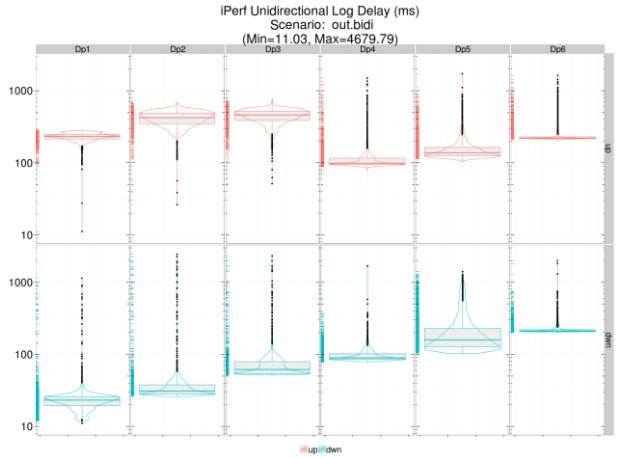


Figure E.1.1.8 – One-way iPerf delay distribution (well managed cases)

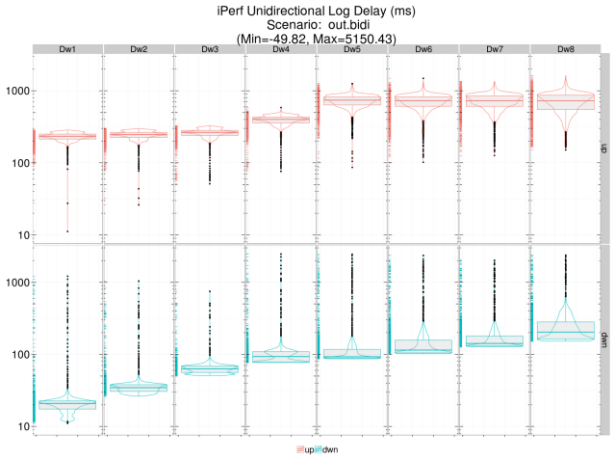


Figure E.1.1.9 – One-way iPerf delay distribution (partly managed cases)

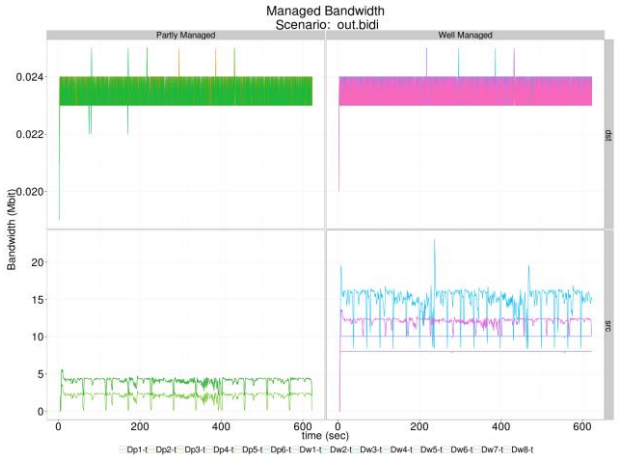


Figure E.1.1.10 – Aggregate managed bandwidth (well and partly managed cases)

E.1.2 DSL upstream iPerf scenario

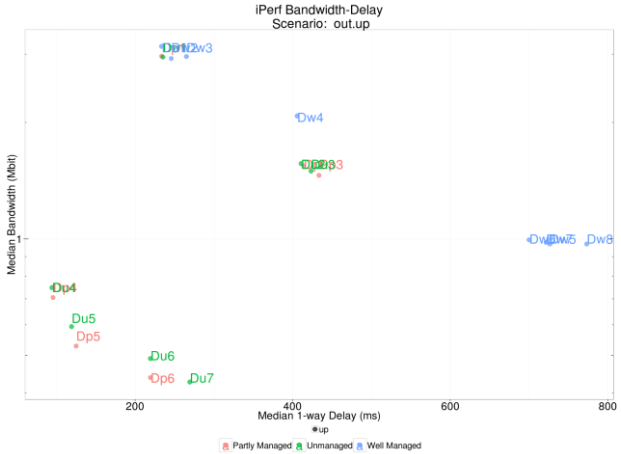


Figure E.1.2.1 – Test case median delay-bandwidth points

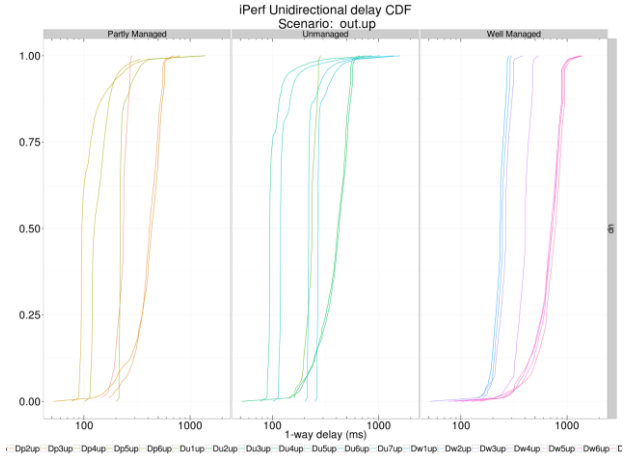


Figure E.1.2.2 – One-way iPerf delay CDF by direction and management type

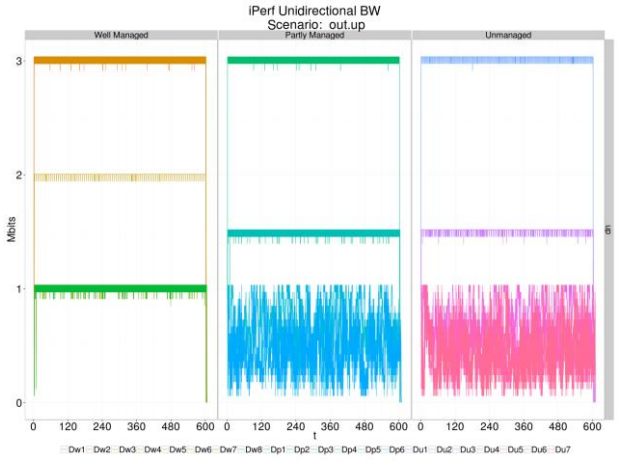


Figure E.1.2.3 – One-way iPerf bandwidth by direction and management type

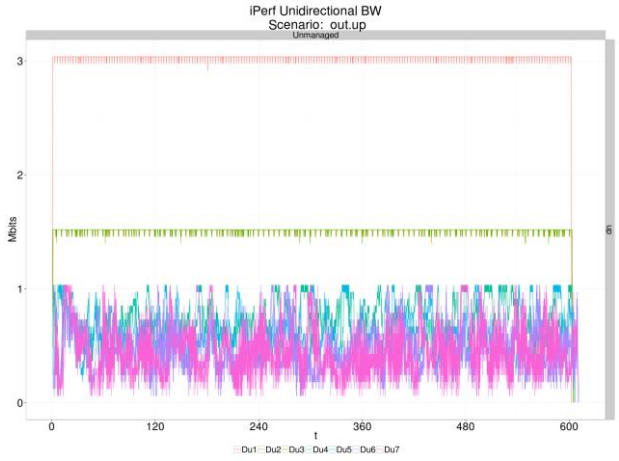


Figure E.1.2.4 – One-way iPerf bandwidth (unmanaged cases)

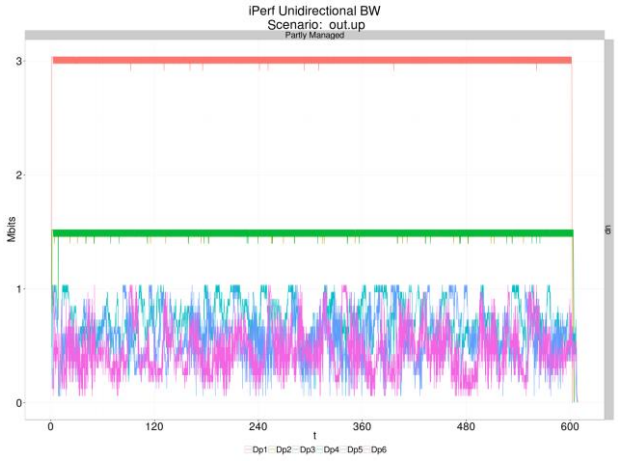


Figure E.1.2.5 – One-way iPerf bandwidth (partly managed cases)

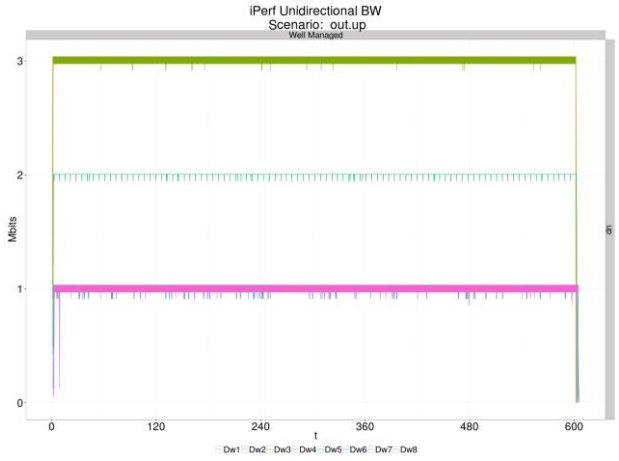


Figure E.1.2.6 – One-way iPerf bandwidth (well managed cases)

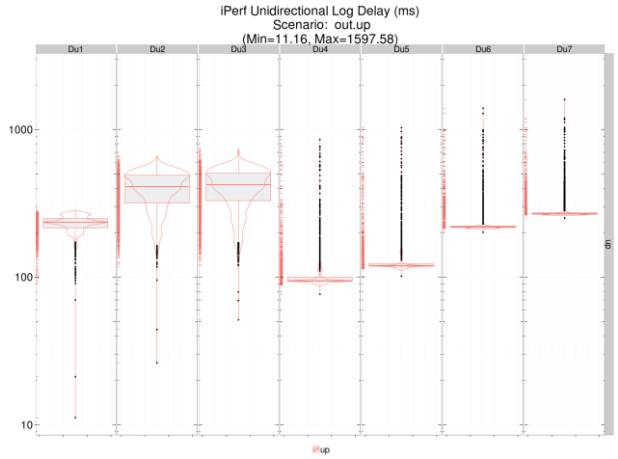


Figure E.1.2.7 – One-way iPerf delay distribution (unmanaged cases)

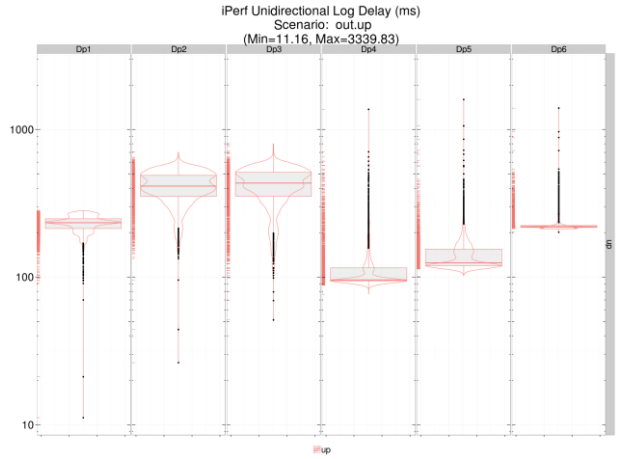


Figure E.1.2.8 – One-way iPerf delay distribution (well managed cases)

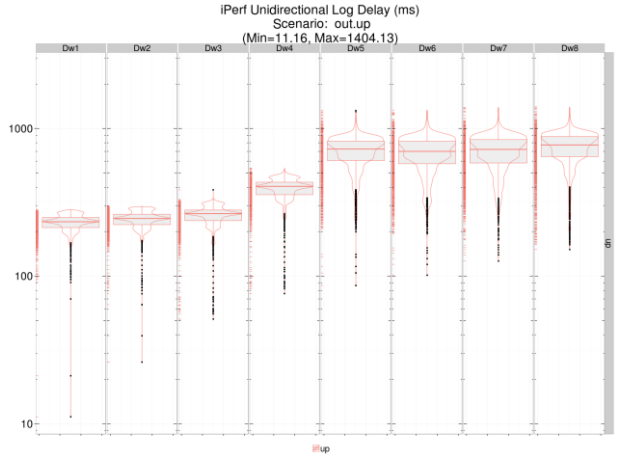


Figure E.1.2.9 – One-way iPerf delay distribution (unmanaged cases)

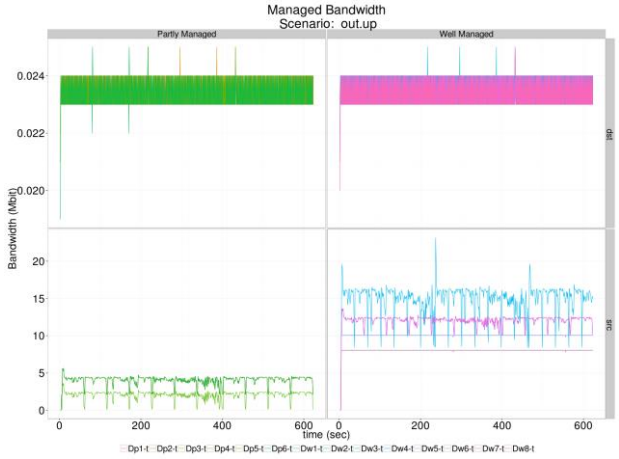


Figure E.1.2.10 – One-way iPerf delay distribution (well managed cases)

E.1.3 DSL downstream iPerf scenario

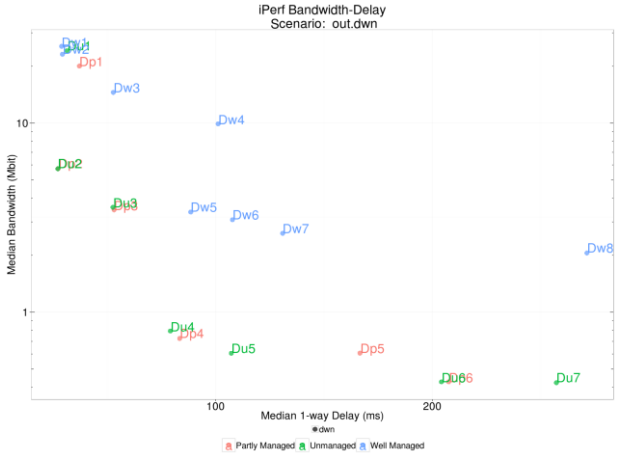


Figure E.1.3.1 – Test case median delay-bandwidth points

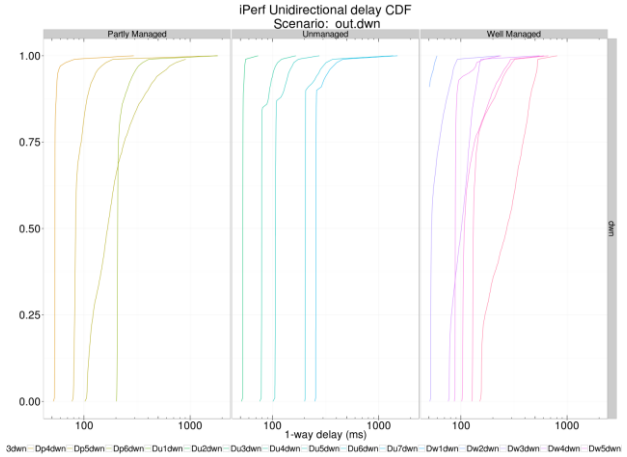


Figure E.1.3.2 – One-way iPerf delay CDF by direction and management type

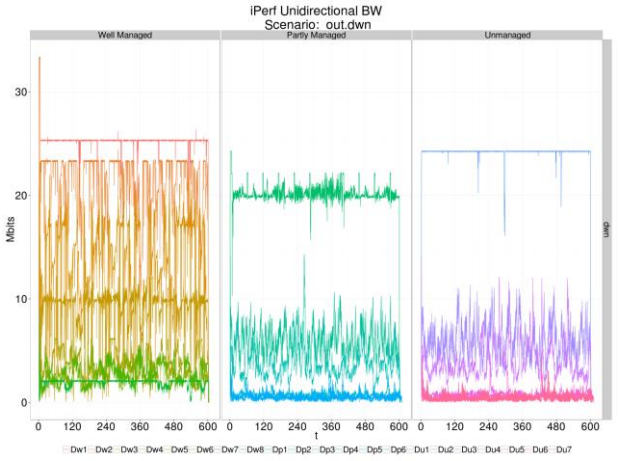


Figure E.1.3.3 – One-way iPerf bandwidth by direction and management type

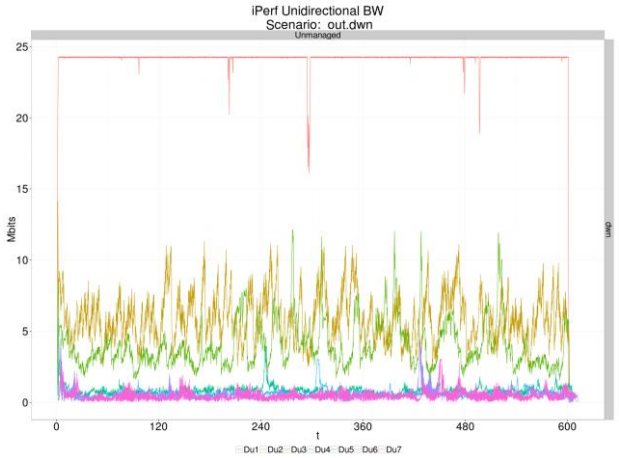


Figure E.1.3.4 – One-way iPerf bandwidth (nmanaged cases)

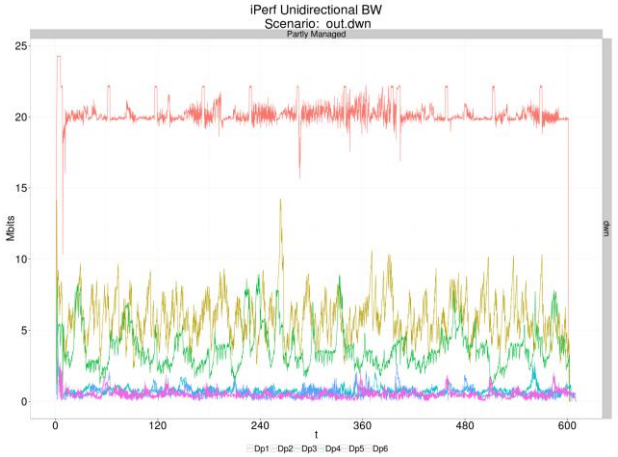


Figure E.1.3.5 – One-way iPerf bandwidth (partly managed cases)

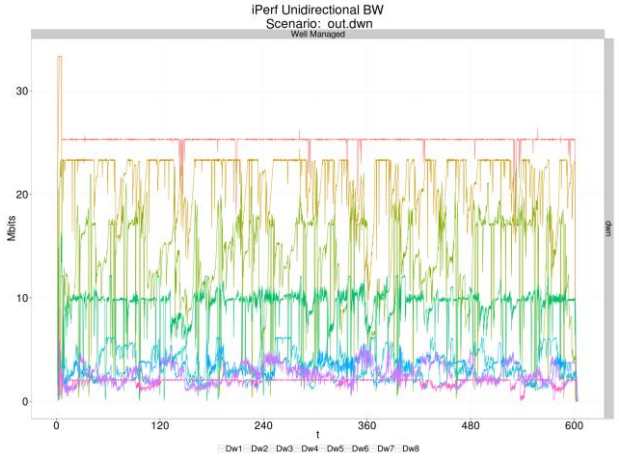


Figure E.1.3.6 – One-way iPerf bandwidth (well managed cases)

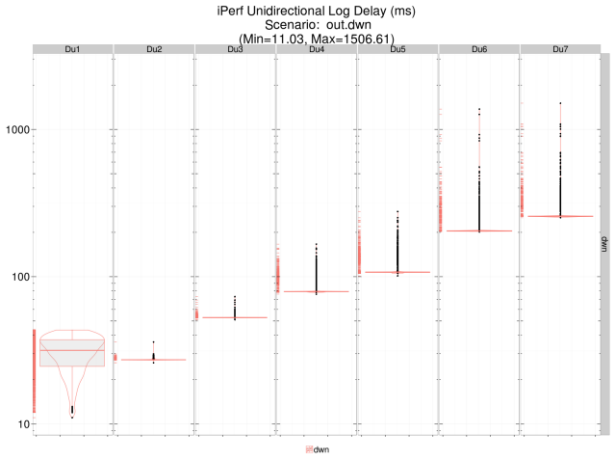


Figure E.1.3.7 – One-way iPerf delay distribution (unmanaged cases)

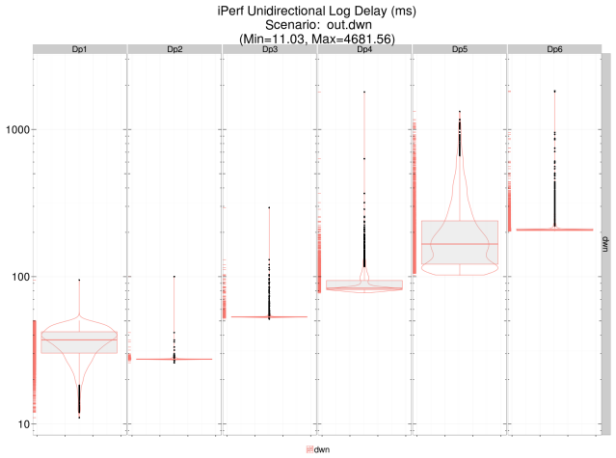


Figure E.1.3.8 – One-way iPerf delay distribution (well managed cases)

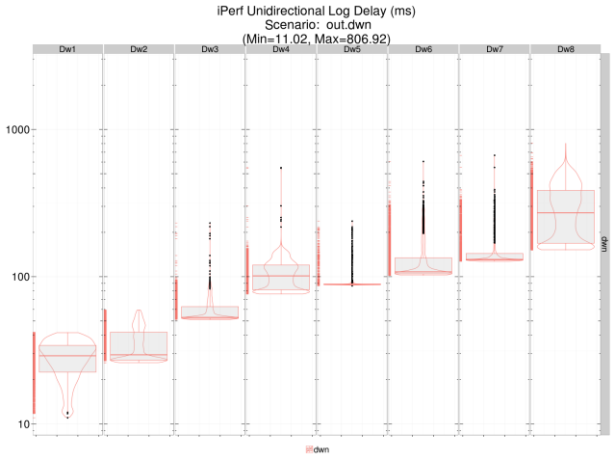


Figure E.1.3.9 – One-way iPerf delay distribution (partly managed cases)

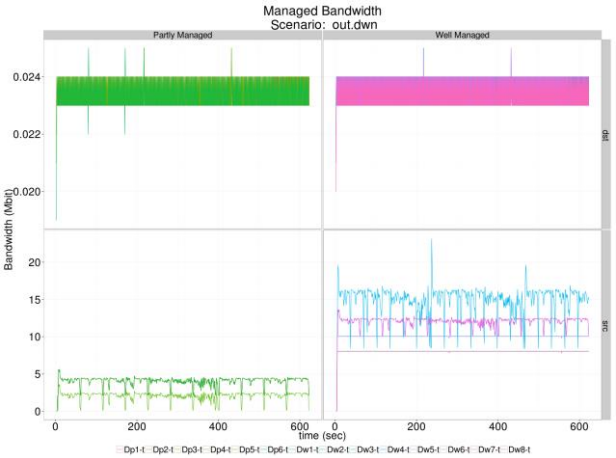


Figure E.1.3.10 – One-way iPerf delay distribution (well managed cases)

E.2 GPON technology

E.2.1 GPON bidirectional iPerf ccenario

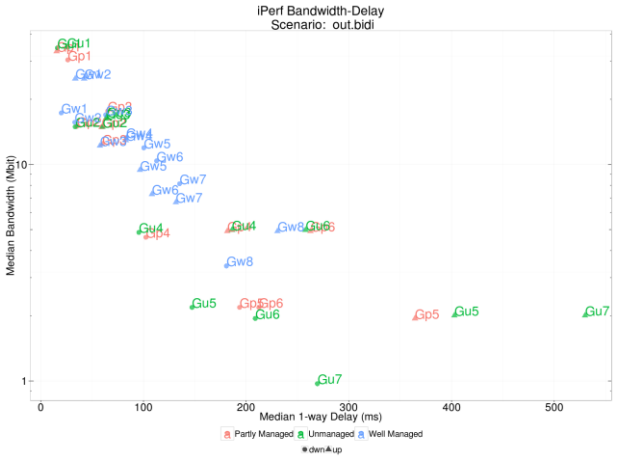


Figure E.2.1.1 – Test case median delay-bandwidth points

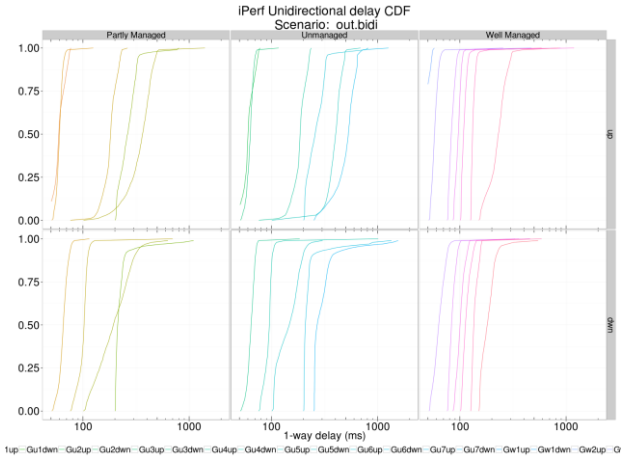


Figure E.2.1.2 – One-way iPerf delay CDF by direction and management type

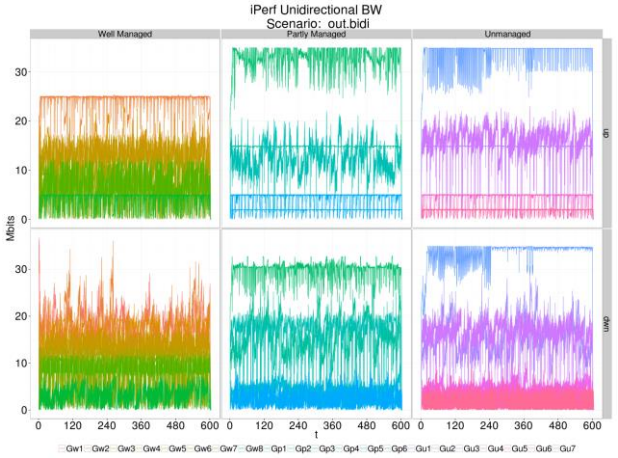


Figure E.2.1.3 – One-way iPerf bandwidth by direction and management type

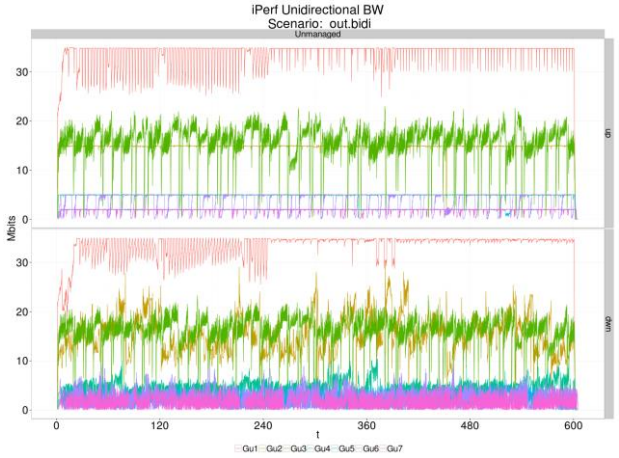


Figure E.2.1.4 – One-way iPerf bandwidth (unmanaged cases)

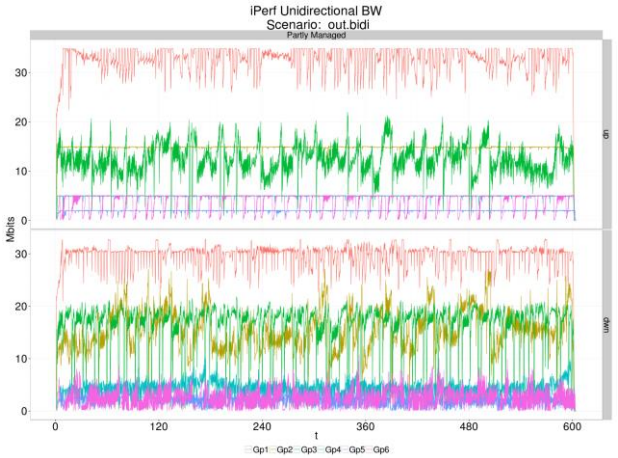


Figure E.2.1.5 – T one-way iPerf bandwidth (partly managed cases)

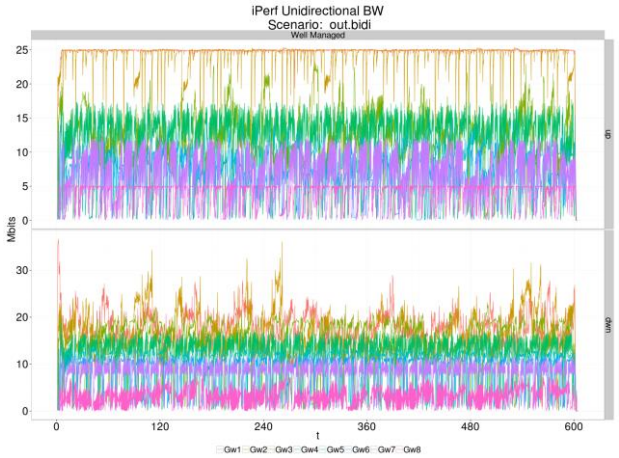


Figure E.2.1.6 – One-way iPerf bandwidth (well managed cases)

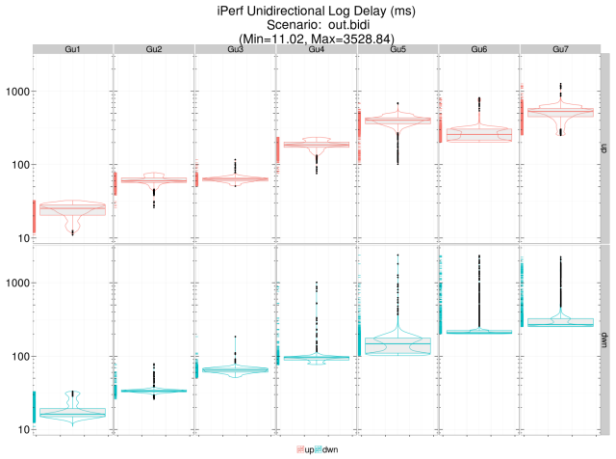


Figure E.2.1.7 – One-way iPerf delay distribution (unmanaged cases)

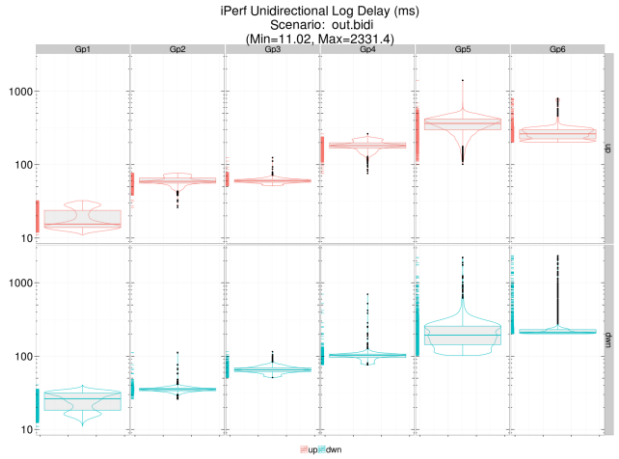


Figure E.2.1.8 – One-way iPerf delay distribution (well managed cases)

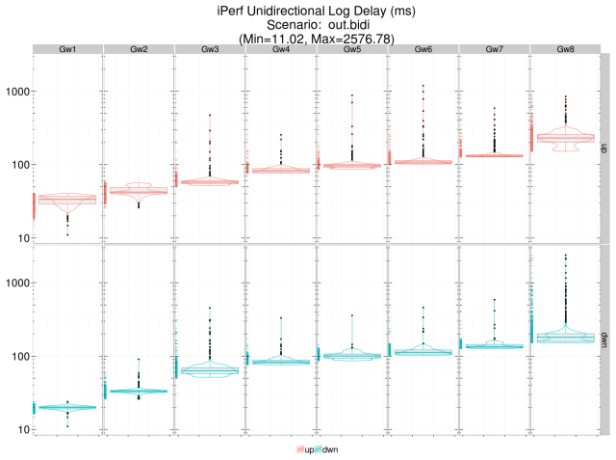


Figure E.2.1.9 – One-way iPerf delay distribution (partly managed cases)

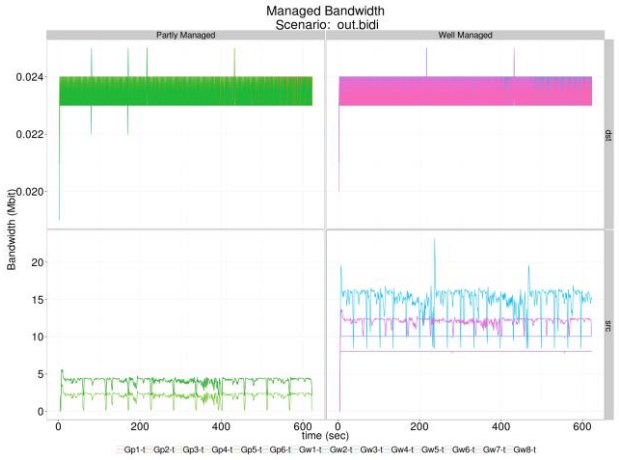


Figure E.2.1.10 – One-way iPerf delay distribution (well managed cases)

E.2.2 GPON upstream iPerf scenario

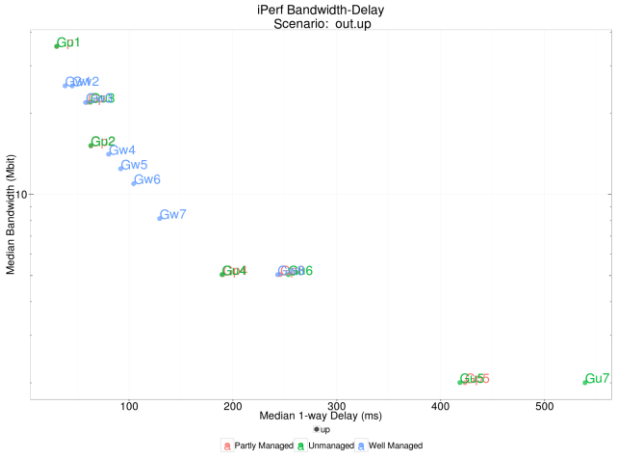


Figure E.2.2.1 – Test case median delay-bandwidth points

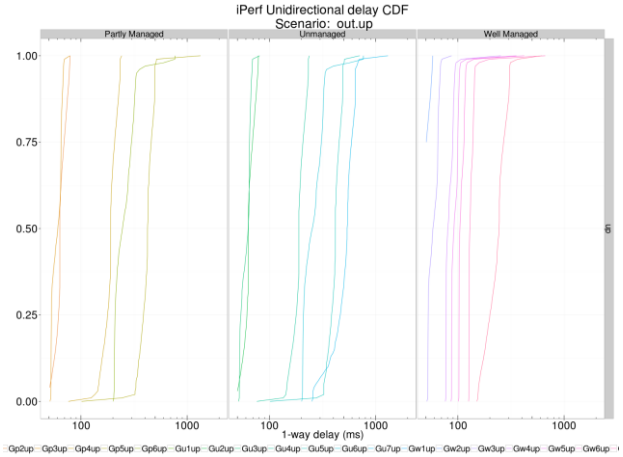


Figure E.2.2.2 – One-way iPerf delay CDF by direction and management type

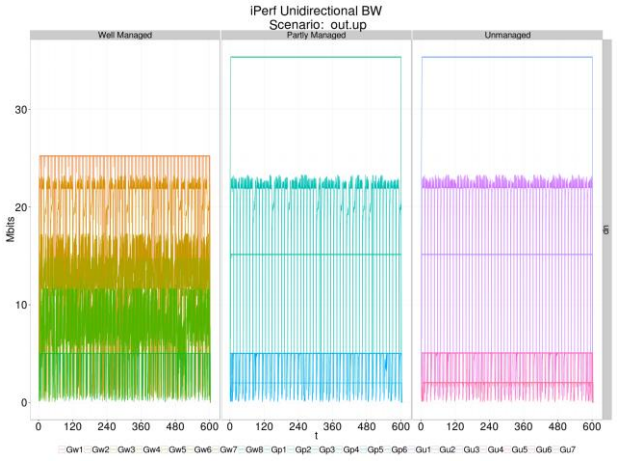


Figure E.2.2.3 – One-way iPerf bandwidth by direction and management type

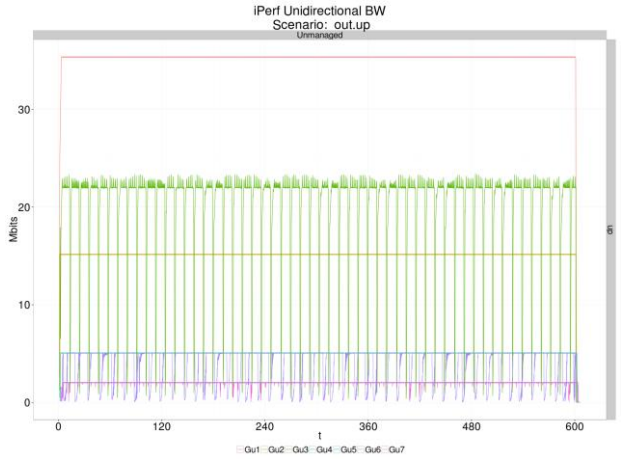


Figure E.2.2.4 – One-way iPerf bandwidth (unmanaged cases)

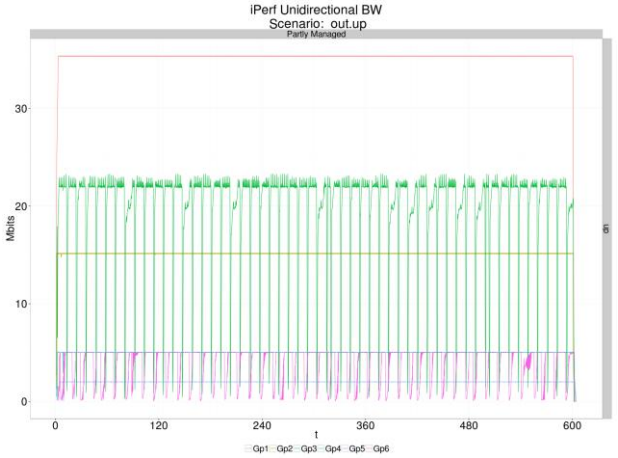


Figure E.2.2.5 – One-way iPerf bandwidth (partly managed cases)

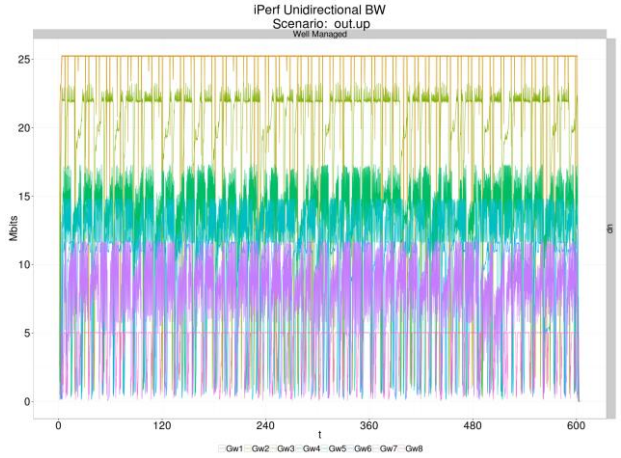


Figure E.2.2.6 – One-way iPerf bandwidth (well managed cases)

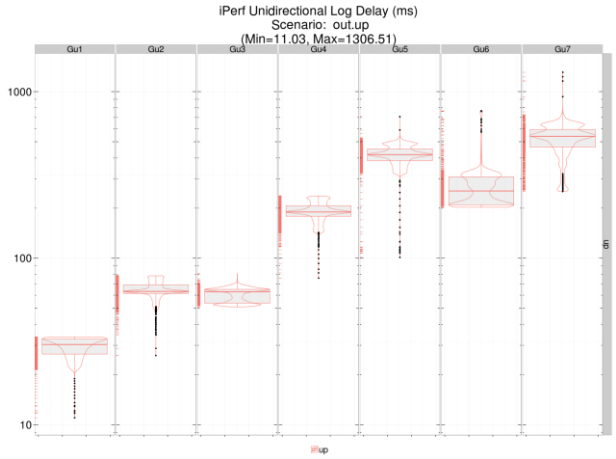


Figure E.2.2.7 – One-way iPerf delay distribution (unmanaged cases)

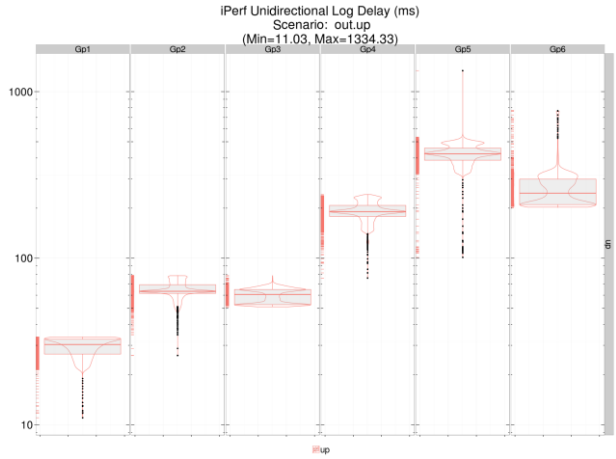


Figure E.2.2.8 – One-way iPerf delay distribution (well managed cases)

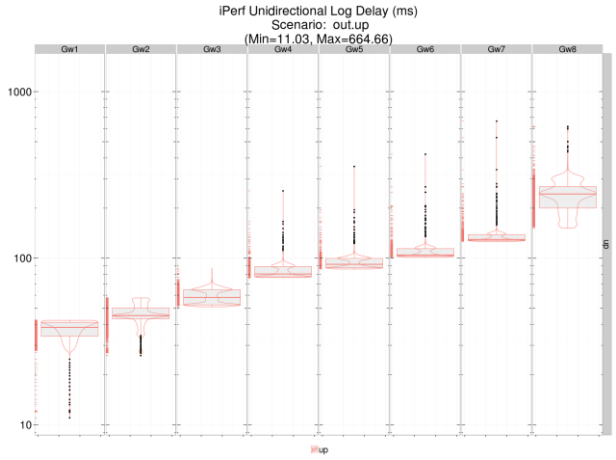


Figure E.2.2.9 – One-way iPerf delay distribution (partly managed cases)

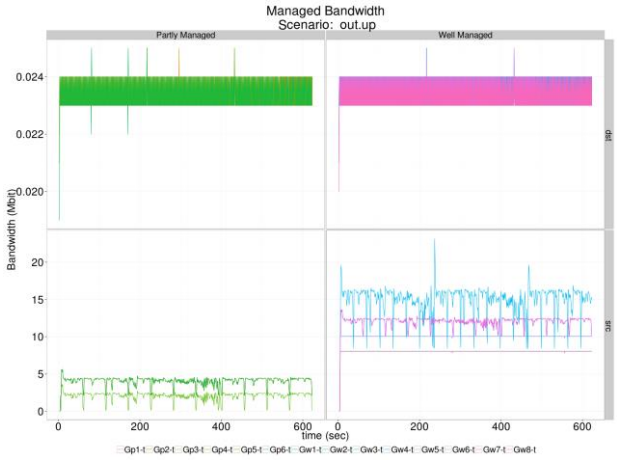


Figure E.2.2.10 – Aggregate managed bandwidth (well and partly managed cases)

E.2.3 GPON downstream iPerf scenario

Figure E.2.3.1 – Test case median delay-bandwidth points

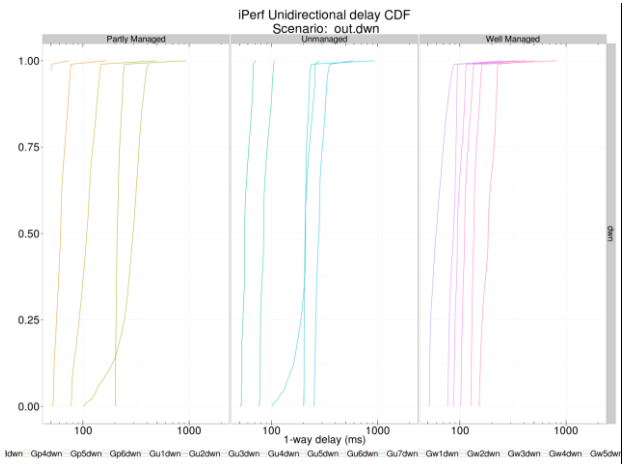


Figure E.2.3.2 – One-way iPerf delay CDF by direction and management type

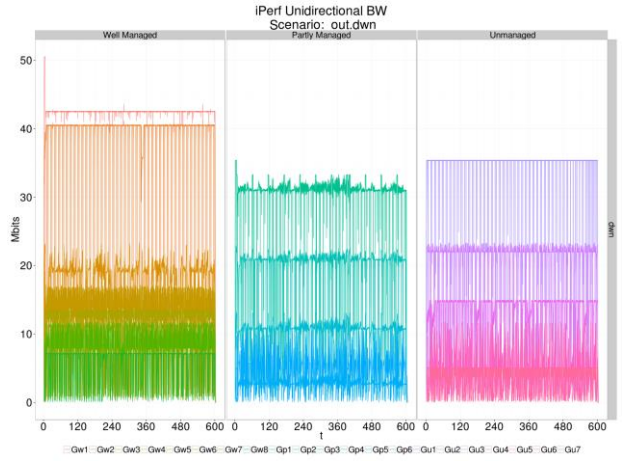


Figure E.2.3.3 – Test one-way iPerf bandwidth by direction and management type

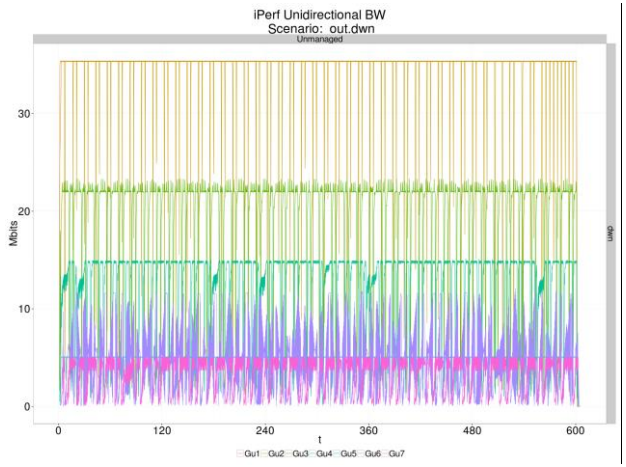


Figure E.2.3.4 – One-way iPerf bandwidth (unmanaged cases)

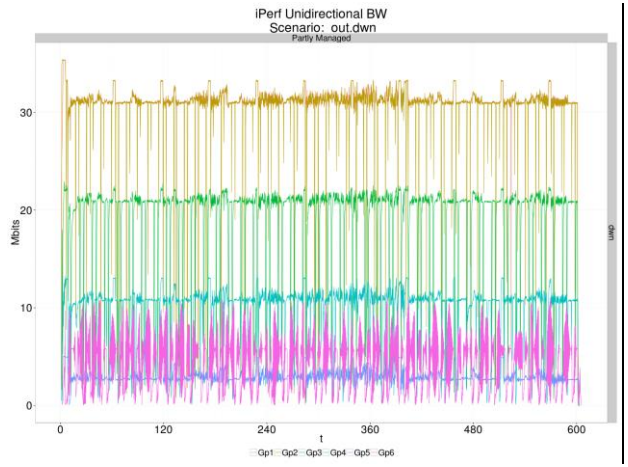


Figure E.2.3.5 – One-way iPerf bandwidth (partly managed cases)

Figure E.2.3.6 – One-way iPerf bandwidth (well managed cases)

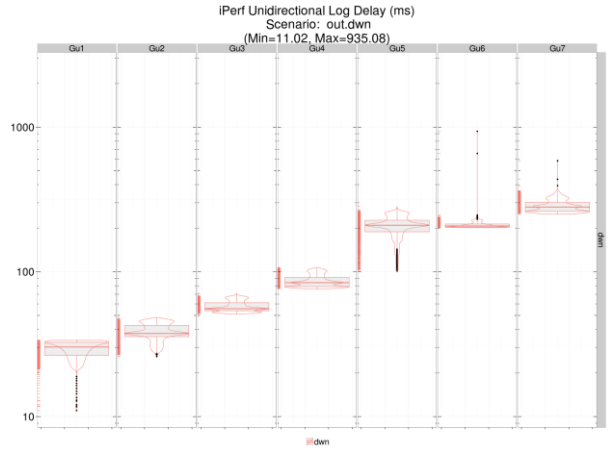


Figure E.2.3.7 – One-way iPerf delay distribution (unmanaged cases)

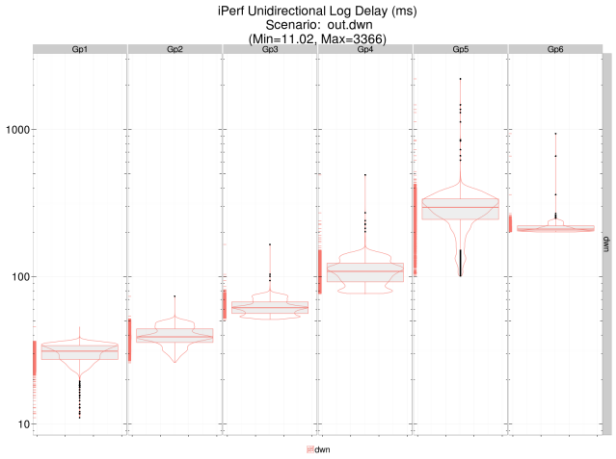


Figure E.2.3.8 – One-way iPerf delay distribution (well managed cases)

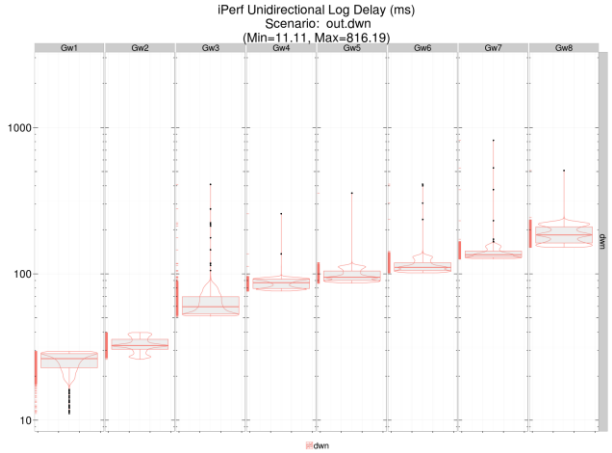


Figure E.2.3.9 – One-way iPerf delay distribution (partly managed cases)

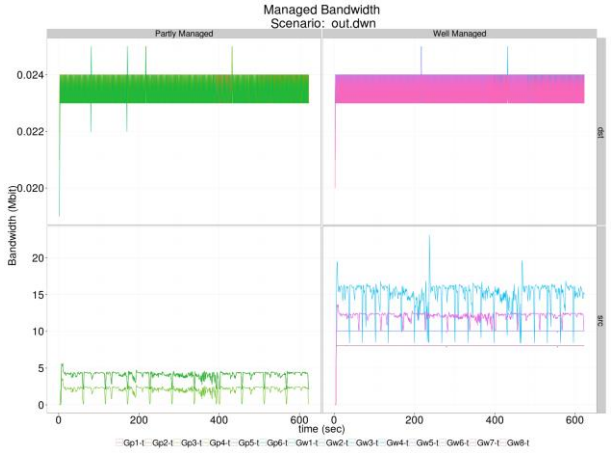


Figure E.2.3.10 – Aggregate managed bandwidth (well and partly managed cases)

Annex F

Simulator output

(This annex forms an integral part of this Recommendation.)

Simulation results are found in the ./D and ./G directories of the model directory of the electronic attachment (Annex F). The simulation control scripts generate result files with the following naming conventions:

– $TECH – Technology (D=DSL, G=GPON);

– $SC – iPerf Scenario: (out.bidi, out.dwn, out.up);

– $TC – test case name (./tc/$TC.param). These match the column headings in Tables 6, 7 and 8;

– $FLOW – name of the managed monitor/flow in test case. These match the PCAP file names listed in Annex E.

The results for each test case are found in a separate directory with the name of the test case. Each monitor specified in the test case represents a particular flow. Files are generated as shown in Table F.1:

Table F.1 – Simulator output file structure

Directory or file name	Description
./$TECH/$SC/cdf.csv	Summary iPerf CDF Table (Appendix E)
./$TECH/$SC/PDF/$TC.pdf	Directory containing .pdf files of merged plots of a scenario
./$TECH/$SC/$TC/	Directory containing test case outputs.
./$TECH/$SC/$TC/$FLOW.out	Output .out delay-loss schedule
./$TECH/$SC/$TC/$FLOW-PDV.csv	Packet delay variation (PDV) file
./$TECH/$SC/$TC/$TC-$FLOW.png	Time series plot of .out file
./$TECH/$SC/$TC/$TC-$FLOW-PDV.png	PDV plot of delay-loss file
./$TECH/$SC/$TC/bw-{up,dwn}.csv	iPerf Delay/bandwidth data file
./$TECH/$SC/PCAP/$TC-modem.pcap.gz	Test case traffic file

Tables G.1 and G.2 summarize the simulation results of the managed flows for DSL and GPON respectively. Each row in the spreadsheet corresponds to a specific flow of a specific test case. Each flow results in the four $FLOW files indicated above.

Note that some of the output files, particularly the large ones, are compressed using the GZIP file format, and have an additional `.gz` suffix.

Annex G

Simulation results summary

(This annex forms an integral part of this Recommendation.)

Table G.1 and Table G.2 summarize the simulation results. Each row of the tables represent a single flow of a test case. The tables have been sorted by flow so that the behaviour can be observed across a flow type with increasing impairment severity. The coloration is an indication of the test profile (green – well managed, yellow – partially managed, red – unmanaged). Table G.3 summarizes simulation results (bandwidth, one-way delay and RTT) for iPerf flows for each test case.

Description of the columns:

Profile – W: well managed
– P: partially managed
– U: unmanaged

Case – test case name

File – output file name root (flow)

n1 – number of packets

min – min delay (ms)

max – max delay (ms)

drop – total number of packets lost

seq – number of packets lost in sequential events

drop % – total percentage of packets lost

seq % – percentage of packets lost sequentially

max-min – peak delay variation (ms)

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Dp1t	**SDTV1_Down**	bidi	113084	11.73	12.98	9	0	0.0080%	0.0000%	1.25
		dwn	113085	11.73	13.02	8	0	0.0071%	0.0000%	1.28
		up	113086	11.73	12.80	7	0	0.0062%	0.0000%	1.07
	SDTV2_Down	bidi	114887	11.73	13.31	12	0	0.0104%	0.0000%	1.58
		dwn	114891	11.73	13.35	8	0	0.0070%	0.0000%	1.62
		up	114890	11.73	13.13	9	0	0.0078%	0.0000%	1.40
	VoIP1_Fwd	bidi	20737	11.04	11.68	1	0	0.0048%	0.0000%	0.64
		dwn	20738	11.04	11.69	0	0	0.0000%	0.0000%	0.65
		up	20738	11.04	11.64	0	0	0.0000%	0.0000%	0.59
	VoIP1_Rev	bidi	20736	11.21	15.30	0	0	0.0000%	0.0000%	4.09
		dwn	20736	11.21	11.43	0	0	0.0000%	0.0000%	0.21
		up	20736	11.21	15.22	0	0	0.0000%	0.0000%	4.01
Dp2t	**SDTV1_Down**	bidi	112961	26.89	42.65	129	0	0.1142%	0.0000%	15.76
		dwn	112971	26.89	42.65	119	0	0.1053%	0.0000%	15.76
		up	112949	26.89	42.65	141	0	0.1248%	0.0000%	15.76
	SDTV2_Down	bidi	114779	26.89	43.06	120	0	0.1045%	0.0000%	16.17
		dwn	114755	26.89	43.06	144	0	0.1255%	0.0000%	16.17
		up	114779	26.89	43.06	120	0	0.1045%	0.0000%	16.17
	VoIP1_Fwd	bidi	20737	26.05	26.87	0	0	0.0000%	0.0000%	0.82
		dwn	20735	26.05	26.87	2	0	0.0096%	0.0000%	0.82
		up	20735	26.05	26.80	2	0	0.0096%	0.0000%	0.74
	VoIP1_Rev	bidi	20733	26.41	34.46	2	0	0.0096%	0.0000%	8.05
		dwn	20734	26.41	26.80	1	0	0.0048%	0.0000%	0.39
		up	20733	26.41	34.42	2	0	0.0096%	0.0000%	8.01

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Dp3t	**SDTV1_Down**	bidi	112943	52.21	117.24	143	0	0.1266%	0.0000%	65.03
		dwn	112956	52.21	117.18	130	0	0.1151%	0.0000%	64.97
		up	112945	52.21	117.24	141	0	0.1248%	0.0000%	65.03
	SDTV2_Down	bidi	114756	52.21	117.26	143	0	0.1246%	0.0000%	65.05
		dwn	114772	52.21	117.29	127	0	0.1107%	0.0000%	65.08
		up	114777	52.21	117.26	122	0	0.1063%	0.0000%	65.05
	VoIP1_Fwd	bidi	20735	51.07	52.21	2	0	0.0096%	0.0000%	1.14
		dwn	20737	51.07	52.22	0	0	0.0000%	0.0000%	1.14
		up	20735	51.07	52.11	2	0	0.0096%	0.0000%	1.04
	VoIP1_Rev	bidi	20734	51.41	59.49	0	0	0.0000%	0.0000%	8.08
		dwn	20733	51.41	51.79	1	0	0.0048%	0.0000%	0.38
		up	20732	51.41	59.42	2	0	0.0096%	0.0000%	8.01
Dp4t	**SDTV1_Down**	bidi	111826	78.14	208.98	1255	59	1.1223%	0.0528%	130.83
		dwn	111803	78.14	208.14	1278	63	1.1431%	0.0563%	129.99
		up	111809	78.14	208.99	1272	37	1.1377%	0.0331%	130.85
	SDTV2_Down	bidi	113574	78.14	208.98	1324	79	1.1658%	0.0696%	130.84
		dwn	113590	78.14	208.89	1308	58	1.1515%	0.0511%	130.75
		up	113541	78.14	208.99	1357	86	1.1952%	0.0757%	130.84
	VoIP1_Fwd	bidi	20720	76.12	78.25	16	0	0.0772%	0.0000%	2.13
		dwn	20730	76.12	78.25	6	0	0.0289%	0.0000%	2.13
		up	20723	76.12	78.05	13	0	0.0627%	0.0000%	1.93
	VoIP1_Rev	bidi	20717	76.60	88.73	16	0	0.0772%	0.0000%	12.13
		dwn	20722	76.60	77.18	11	0	0.0531%	0.0000%	0.58
		up	20721	76.60	88.72	12	0	0.0579%	0.0000%	12.12

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Dp5t	**SDTV2_Down**	bidi	113320	104.98	367.44	1578	169	1.3925%	0.1491%	262.46
		dwn	113387	104.98	367.43	1511	168	1.3326%	0.1482%	262.46
		up	113360	104.98	367.44	1538	171	1.3567%	0.1508%	262.46
	VoIP1_Fwd	bidi	20727	101.22	105.27	8	0	0.0386%	0.0000%	4.05
		dwn	20723	101.22	105.29	12	0	0.0579%	0.0000%	4.07
		up	20720	101.22	104.96	15	0	0.0724%	0.0000%	3.74
	VoIP1_Rev	bidi	20720	101.60	113.72	13	0	0.0627%	0.0000%	12.12
		dwn	20719	101.60	102.18	14	0	0.0676%	0.0000%	0.58
		up	20722	101.60	113.73	11	0	0.0531%	0.0000%	12.13
Dp6t	**SDTV1_Down**	bidi	111819	203.19	334.00	1239	39	1.1080%	0.0349%	130.81
		dwn	111778	203.19	333.18	1280	46	1.1451%	0.0412%	129.99
		up	111762	203.19	334.03	1296	39	1.1596%	0.0349%	130.85
	SDTV2_Down	bidi	113562	203.19	334.03	1335	80	1.1756%	0.0704%	130.85
		dwn	113591	203.19	333.94	1306	56	1.1497%	0.0493%	130.75
		up	113560	203.19	334.03	1337	80	1.1774%	0.0704%	130.84
	VoIP1_Fwd	bidi	20721	201.12	203.30	11	0	0.0531%	0.0000%	2.18
		dwn	20720	201.12	203.30	12	0	0.0579%	0.0000%	2.18
		up	20717	201.12	203.09	15	0	0.0724%	0.0000%	1.97
	VoIP1_Rev	bidi	20717	201.60	213.73	12	0	0.0579%	0.0000%	12.13
		dwn	20715	201.60	202.18	14	0	0.0676%	0.0000%	0.58
		up	20719	201.60	213.73	10	0	0.0483%	0.0000%	12.13
Dw1t	**HDTV1_Down**	bidi	451520	11.61	12.98	42	0	0.0093%	0.0000%	1.37
		dwn	451522	11.61	13.27	40	0	0.0089%	0.0000%	1.66
		up	451523	11.61	12.98	39	0	0.0086%	0.0000%	1.37
	VoIP1_Fwd	bidi	20738	11.04	11.62	0	0	0.0000%	0.0000%	0.58

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
		dwn	20738	11.04	11.61	0	0	0.0000%	0.0000%	0.58
		up	20737	11.04	11.62	1	0	0.0048%	0.0000%	0.58
	VoIP1_Rev	bidi	20736	11.21	15.27	0	0	0.0000%	0.0000%	4.06
		dwn	20736	11.21	11.42	0	0	0.0000%	0.0000%	0.21
		up	20736	11.21	15.22	0	0	0.0000%	0.0000%	4.01
Dw2t	**HDTV1_Down**	bidi	451520	26.62	28.52	31	0	0.0069%	0.0000%	1.90
		dwn	451497	26.62	28.53	54	0	0.0120%	0.0000%	1.91
		up	451519	26.62	28.32	32	0	0.0071%	0.0000%	1.70
	SDTV1_Down	bidi	113082	26.62	28.25	8	0	0.0071%	0.0000%	1.63
		dwn	113081	26.62	28.26	9	0	0.0080%	0.0000%	1.64
		up	113074	26.62	28.04	16	0	0.0142%	0.0000%	1.43
	VoIP1_Fwd	bidi	20737	26.04	26.63	0	0	0.0000%	0.0000%	0.59
		dwn	20737	26.04	26.63	0	0	0.0000%	0.0000%	0.59
		up	20736	26.04	26.63	1	0	0.0048%	0.0000%	0.59
	VoIP1_Rev	bidi	20735	26.21	30.39	0	0	0.0000%	0.0000%	4.17
		dwn	20735	26.21	26.42	0	0	0.0000%	0.0000%	0.21
		up	20735	26.21	30.34	0	0	0.0000%	0.0000%	4.13
Dw3t	**HDTV1_Down**	bidi	451470	51.63	75.32	63	15	0.0140%	0.0033%	23.69
		dwn	451471	51.63	75.32	62	13	0.0137%	0.0029%	23.69
		up	451458	51.63	75.32	75	15	0.0166%	0.0033%	23.69
	HDTV2_Down	bidi	391370	51.37	75.35	83	10	0.0212%	0.0026%	23.97
		dwn	391374	51.37	75.35	79	14	0.0202%	0.0036%	23.98
		up	391386	51.37	75.35	67	10	0.0171%	0.0026%	23.97
	VoIP1_Fwd	bidi	20737	51.04	51.64	0	0	0.0000%	0.0000%	0.60
		dwn	20737	51.04	51.64	0	0	0.0000%	0.0000%	0.60

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
		up	20737	51.04	51.64	0	0	0.0000%	0.0000%	0.60
	VoIP1_Rev	bidi	20734	51.21	55.29	0	0	0.0000%	0.0000%	4.08
		dwn	20734	51.21	51.47	0	0	0.0000%	0.0000%	0.26
		up	20734	51.21	55.22	0	0	0.0000%	0.0000%	4.01
Dw4t	**HDTV1_Down**	bidi	451474	76.80	103.61	41	0	0.0091%	0.0000%	26.81
		dwn	451469	76.80	103.45	46	0	0.0102%	0.0000%	26.65
		up	451470	76.80	103.25	45	0	0.0100%	0.0000%	26.44
	SDTV1_Down	bidi	113065	76.80	103.30	16	0	0.0142%	0.0000%	26.49
		dwn	113071	76.80	103.14	10	0	0.0088%	0.0000%	26.34
		up	113073	76.80	102.93	8	0	0.0071%	0.0000%	26.13
	SDTV2_Down	bidi	114890	76.80	103.68	8	0	0.0070%	0.0000%	26.88
		dwn	114890	76.80	103.52	8	0	0.0070%	0.0000%	26.72
		up	114891	76.80	103.32	7	0	0.0061%	0.0000%	26.51
	VoIP1_Fwd	bidi	20736	76.05	76.81	0	0	0.0000%	0.0000%	0.76
		dwn	20736	76.05	76.81	0	0	0.0000%	0.0000%	0.76
		up	20736	76.05	76.81	0	0	0.0000%	0.0000%	0.76
	VoIP1_Rev	bidi	20733	76.31	82.40	0	0	0.0000%	0.0000%	6.09
		dwn	20733	76.31	76.68	0	0	0.0000%	0.0000%	0.37
		up	20733	76.31	82.32	0	0	0.0000%	0.0000%	6.01
Dw5t	**HDTV1_Down**	bidi	451004	87.00	94.30	504	0	0.1118%	0.0000%	7.30
		dwn	450974	87.00	93.60	534	0	0.1184%	0.0000%	6.60
		up	451000	87.00	93.60	508	0	0.1126%	0.0000%	6.60
	SDTV1_Down	bidi	112944	87.00	93.33	135	0	0.1195%	0.0000%	6.33
		dwn	112973	87.00	92.62	106	0	0.0938%	0.0000%	5.62
		up	112947	87.00	92.62	132	0	0.1169%	0.0000%	5.62

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
	VoIP1_Fwd	bidi	20735	86.06	87.01	0	0	0.0000%	0.0000%	0.95
		dwn	20735	86.06	87.01	0	0	0.0000%	0.0000%	0.95
		up	20734	86.06	87.01	1	0	0.0048%	0.0000%	0.95
	VoIP1_Rev	bidi	20731	86.60	98.68	2	0	0.0096%	0.0000%	12.08
		dwn	20730	86.60	87.18	3	0	0.0145%	0.0000%	0.58
		up	20732	86.60	98.62	1	0	0.0048%	0.0000%	12.02
Dw6t	**HDTV1_Down**	bidi	450970	102.01	150.97	527	8	0.1169%	0.0018%	48.96
		dwn	450996	102.01	150.94	501	14	0.1111%	0.0031%	48.93
		up	450997	102.01	150.94	500	12	0.1109%	0.0027%	48.93
	SDTV1_Down	bidi	112945	102.01	150.63	131	5	0.1160%	0.0044%	48.62
		dwn	112930	102.01	150.91	146	6	0.1293%	0.0053%	48.89
		up	112940	102.01	150.91	136	6	0.1204%	0.0053%	48.89
	SDTV2_Down	bidi	114754	102.01	150.95	144	8	0.1255%	0.0070%	48.94
		dwn	114720	102.01	150.96	178	8	0.1552%	0.0070%	48.94
		up	114739	102.01	150.96	159	6	0.1386%	0.0052%	48.94
	VoIP1_Fwd	bidi	20735	101.06	102.02	0	0	0.0000%	0.0000%	0.96
		dwn	20735	101.06	102.02	0	0	0.0000%	0.0000%	0.96
		up	20734	101.06	102.02	1	0	0.0048%	0.0000%	0.96
	VoIP1_Rev	bidi	20729	101.60	113.62	4	0	0.0193%	0.0000%	12.02
		dwn	20731	101.60	102.25	2	0	0.0096%	0.0000%	0.65
		up	20731	101.60	113.62	2	0	0.0096%	0.0000%	12.02
Dw7t	**HDTV1_Down**	bidi	450958	127.02	175.95	521	12	0.1155%	0.0027%	48.93
		dwn	450955	127.02	175.95	524	12	0.1162%	0.0027%	48.93
		up	450962	127.02	175.95	517	12	0.1146%	0.0027%	48.93
	SDTV1_Down	bidi	112938	127.02	175.92	134	6	0.1186%	0.0053%	48.89

Table G.1 – Simulation results summary DSL managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
		dwn	112940	127.02	175.92	132	6	0.1169%	0.0053%	48.89
		up	112933	127.02	175.92	139	6	0.1231%	0.0053%	48.89
	SDTV2_Down	bidi	114740	127.02	175.97	158	8	0.1377%	0.0070%	48.94
		dwn	114732	127.02	175.97	166	6	0.1447%	0.0052%	48.94
		up	114755	127.02	175.97	143	6	0.1246%	0.0052%	48.94
	VoIP1_Fwd	bidi	20733	126.06	127.03	1	0	0.0048%	0.0000%	0.97
		dwn	20734	126.06	127.03	0	0	0.0000%	0.0000%	0.97
		up	20732	126.06	127.03	2	0	0.0096%	0.0000%	0.97
	VoIP1_Rev	bidi	20729	126.60	138.64	3	0	0.0145%	0.0000%	12.04
		dwn	20731	126.60	127.24	1	0	0.0048%	0.0000%	0.64
		up	20731	126.60	138.62	1	0	0.0048%	0.0000%	12.01
Dw8t	**HDTV1_Down**	bidi	450926	152.44	164.40	533	2	0.1182%	0.0004%	11.96
		dwn	450950	152.44	164.56	509	0	0.1129%	0.0000%	12.12
		up	450922	152.44	163.71	537	0	0.1191%	0.0000%	11.27
	VoIP1_Fwd	bidi	20732	151.08	152.45	1	0	0.0048%	0.0000%	1.37
		dwn	20733	151.08	152.46	0	0	0.0000%	0.0000%	1.38
		up	20733	151.08	152.45	0	0	0.0000%	0.0000%	1.37
	VoIP1_Rev	bidi	20731	151.60	163.62	0	0	0.0000%	0.0000%	12.01
		dwn	20729	151.60	152.47	2	0	0.0096%	0.0000%	0.86
		up	20727	151.60	163.62	4	0	0.0193%	0.0000%	12.02

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Gp1t	**SDTV1_Down**	bidi	113093	11.59	12.24	0	0	0.0000%	0.0000%	0.65
		dwn	113093	11.59	12.24	0	0	0.0000%	0.0000%	0.65
		up	113093	11.59	11.91	0	0	0.0000%	0.0000%	0.32
	SDTV2_Down	bidi	114899	11.59	12.25	0	0	0.0000%	0.0000%	0.66
		dwn	114899	11.59	12.24	0	0	0.0000%	0.0000%	0.65
		up	114899	11.59	11.91	0	0	0.0000%	0.0000%	0.32
	VoIP1_Fwd	bidi	20738	11.04	11.53	0	0	0.0000%	0.0000%	0.50
		dwn	20738	11.04	11.53	0	0	0.0000%	0.0000%	0.50
		up	20738	11.04	11.49	0	0	0.0000%	0.0000%	0.46
	VoIP1_Rev	bidi	20736	11.04	11.53	0	0	0.0000%	0.0000%	0.49
		dwn	20736	11.04	11.05	0	0	0.0000%	0.0000%	0.02
		up	20736	11.04	11.52	0	0	0.0000%	0.0000%	0.49
Gp2t	**SDTV1_Down**	bidi	113090	26.60	27.25	0	0	0.0000%	0.0000%	0.65
		dwn	113090	26.60	27.25	0	0	0.0000%	0.0000%	0.65
		up	113090	26.60	26.92	0	0	0.0000%	0.0000%	0.32
	SDTV2_Down	bidi	114899	26.60	27.24	0	0	0.0000%	0.0000%	0.64
		dwn	114899	26.60	27.25	0	0	0.0000%	0.0000%	0.65
		up	114899	26.60	26.92	0	0	0.0000%	0.0000%	0.32
	VoIP1_Fwd	bidi	20737	26.04	26.54	0	0	0.0000%	0.0000%	0.50
		dwn	20737	26.04	26.54	0	0	0.0000%	0.0000%	0.51
		up	20737	26.04	26.51	0	0	0.0000%	0.0000%	0.47
	VoIP1_Rev	bidi	20735	26.06	26.99	0	0	0.0000%	0.0000%	0.93
		dwn	20735	26.06	26.10	0	0	0.0000%	0.0000%	0.04
		up	20735	26.06	26.99	0	0	0.0000%	0.0000%	0.93

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Gp3t	**SDTV1_Down**	bidi	113086	51.73	52.65	0	0	0.0000%	0.0000%	0.91
		dwn	113086	51.73	52.65	0	0	0.0000%	0.0000%	0.91
		up	113086	51.73	52.19	0	0	0.0000%	0.0000%	0.45
	SDTV2_Down	bidi	114899	51.73	52.65	0	0	0.0000%	0.0000%	0.91
		dwn	114899	51.73	52.65	0	0	0.0000%	0.0000%	0.91
		up	114899	51.73	52.19	0	0	0.0000%	0.0000%	0.46
	VoIP1_Fwd	bidi	20737	51.05	51.69	0	0	0.0000%	0.0000%	0.65
		dwn	20737	51.05	51.69	0	0	0.0000%	0.0000%	0.65
		up	20737	51.05	51.64	0	0	0.0000%	0.0000%	0.60
	VoIP1_Rev	bidi	20734	51.05	51.69	0	0	0.0000%	0.0000%	0.65
		dwn	20734	51.05	51.07	0	0	0.0000%	0.0000%	0.02
		up	20734	51.05	51.69	0	0	0.0000%	0.0000%	0.65
Gp4t	**SDTV1_Down**	bidi	113081	77.06	110.41	0	0	0.0000%	0.0000%	33.35
		dwn	113081	77.06	110.73	0	0	0.0000%	0.0000%	33.67
		up	113081	77.06	110.21	0	0	0.0000%	0.0000%	33.15
	SDTV2_Down	bidi	114895	77.06	110.54	3	0	0.0026%	0.0000%	33.48
		dwn	114895	77.06	110.86	3	0	0.0026%	0.0000%	33.80
		up	114895	77.06	110.34	3	0	0.0026%	0.0000%	33.28
	VoIP1_Fwd	bidi	20736	76.06	77.05	0	0	0.0000%	0.0000%	0.99
		dwn	20736	76.06	77.05	0	0	0.0000%	0.0000%	0.99
		up	20736	76.06	76.96	0	0	0.0000%	0.0000%	0.90
	VoIP1_Rev	bidi	20733	76.14	78.65	0	0	0.0000%	0.0000%	2.51
		dwn	20733	76.14	76.26	0	0	0.0000%	0.0000%	0.12
		up	20733	76.14	78.62	0	0	0.0000%	0.0000%	2.48

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Gp5t	**SDTV2_Down**	bidi	114833	103.53	260.64	65	42	0.0566%	0.0366%	157.11
		dwn	114835	103.53	260.53	63	42	0.0549%	0.0366%	157.01
		up	114836	103.53	260.54	62	42	0.0540%	0.0366%	157.01
	VoIP1_Fwd	bidi	20735	101.14	103.67	0	0	0.0000%	0.0000%	2.53
		dwn	20735	101.14	103.67	0	0	0.0000%	0.0000%	2.52
		up	20735	101.14	103.44	0	0	0.0000%	0.0000%	2.30
	VoIP1_Rev	bidi	20733	101.31	107.43	0	0	0.0000%	0.0000%	6.11
		dwn	20733	101.31	101.64	0	0	0.0000%	0.0000%	0.33
		up	20733	101.31	107.48	0	0	0.0000%	0.0000%	6.16
Gp6t	**SDTV1_Down**	bidi	113054	202.10	235.25	4	0	0.0035%	0.0000%	33.15
		dwn	113055	202.10	235.25	3	0	0.0027%	0.0000%	33.15
		up	113058	202.10	235.25	0	0	0.0000%	0.0000%	33.15
	SDTV2_Down	bidi	114895	202.10	235.38	2	0	0.0017%	0.0000%	33.28
		dwn	114896	202.10	235.38	1	0	0.0009%	0.0000%	33.28
		up	114896	202.10	235.38	1	0	0.0009%	0.0000%	33.28
	VoIP1_Fwd	bidi	20732	201.07	202.10	0	0	0.0000%	0.0000%	1.03
		dwn	20732	201.07	202.10	0	0	0.0000%	0.0000%	1.03
		up	20732	201.07	202.01	0	0	0.0000%	0.0000%	0.94
	VoIP1_Rev	bidi	20729	201.14	203.69	0	0	0.0000%	0.0000%	2.54
		dwn	20729	201.14	201.31	0	0	0.0000%	0.0000%	0.17
		up	20729	201.14	203.69	0	0	0.0000%	0.0000%	2.55
Gw1t	**HDTV1_Down**	bidi	451561	11.50	11.93	1	0	0.0002%	0.0000%	0.43
		dwn	451562	11.50	11.90	0	0	0.0000%	0.0000%	0.40
		up	451562	11.50	11.71	0	0	0.0000%	0.0000%	0.22

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
	VoIP1_Fwd	bidi	20738	11.03	11.50	0	0	0.0000%	0.0000%	0.47
		dwn	20738	11.03	11.50	0	0	0.0000%	0.0000%	0.47
		up	20738	11.03	11.50	0	0	0.0000%	0.0000%	0.47
	VoIP1_Rev	bidi	20736	11.04	11.67	0	0	0.0000%	0.0000%	0.62
		dwn	20736	11.04	11.07	0	0	0.0000%	0.0000%	0.02
		up	20736	11.04	11.66	0	0	0.0000%	0.0000%	0.62
Gw2t	**HDTV1_Down**	bidi	451552	26.51	26.97	0	0	0.0000%	0.0000%	0.46
		dwn	451552	26.51	26.97	0	0	0.0000%	0.0000%	0.46
		up	451552	26.51	26.73	0	0	0.0000%	0.0000%	0.23
	SDTV1_Down	bidi	113090	26.51	26.96	0	0	0.0000%	0.0000%	0.45
		dwn	113090	26.51	26.97	0	0	0.0000%	0.0000%	0.47
		up	113090	26.51	26.73	0	0	0.0000%	0.0000%	0.22
	VoIP1_Fwd	bidi	20737	26.03	26.51	0	0	0.0000%	0.0000%	0.48
		dwn	20737	26.03	26.51	0	0	0.0000%	0.0000%	0.48
		up	20737	26.03	26.51	0	0	0.0000%	0.0000%	0.48
	VoIP1_Rev	bidi	20735	26.04	26.68	0	0	0.0000%	0.0000%	0.63
		dwn	20735	26.04	26.07	0	0	0.0000%	0.0000%	0.02
		up	20735	26.04	26.68	0	0	0.0000%	0.0000%	0.63
Gw3t	**HDTV1_Down**	bidi	451529	51.61	73.98	4	2	0.0009%	0.0004%	22.37
		dwn	451526	51.61	73.96	7	7	0.0016%	0.0016%	22.35
		up	451530	51.61	73.91	3	0	0.0007%	0.0000%	22.30
	HDTV2_Down	bidi	391440	51.36	73.97	13	4	0.0033%	0.0010%	22.61
		dwn	391442	51.36	73.97	11	0	0.0028%	0.0000%	22.61
		up	391440	51.36	73.96	13	4	0.0033%	0.0010%	22.60

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
	VoIP1_Fwd	bidi	20737	51.04	51.62	0	0	0.0000%	0.0000%	0.58
		dwn	20737	51.04	51.62	0	0	0.0000%	0.0000%	0.58
		up	20737	51.04	51.62	0	0	0.0000%	0.0000%	0.58
	VoIP1_Rev	bidi	20734	51.05	51.69	0	0	0.0000%	0.0000%	0.65
		dwn	20734	51.05	51.07	0	0	0.0000%	0.0000%	0.02
		up	20734	51.05	51.69	0	0	0.0000%	0.0000%	0.65
Gw4t	**HDTV1_Down**	bidi	451514	76.62	78.23	1	0	0.0002%	0.0000%	1.61
		dwn	451515	76.62	78.26	0	0	0.0000%	0.0000%	1.63
		up	451515	76.62	78.11	0	0	0.0000%	0.0000%	1.49
	SDTV1_Down	bidi	113081	76.62	78.00	0	0	0.0000%	0.0000%	1.38
		dwn	113081	76.62	77.82	0	0	0.0000%	0.0000%	1.20
		up	113081	76.62	77.82	0	0	0.0000%	0.0000%	1.20
	SDTV2_Down	bidi	114898	76.62	78.11	0	0	0.0000%	0.0000%	1.49
		dwn	114898	76.62	77.91	0	0	0.0000%	0.0000%	1.29
		up	114898	76.62	77.81	0	0	0.0000%	0.0000%	1.19
	VoIP1_Fwd	bidi	20736	76.04	76.63	0	0	0.0000%	0.0000%	0.59
		dwn	20736	76.04	76.63	0	0	0.0000%	0.0000%	0.59
		up	20736	76.04	76.63	0	0	0.0000%	0.0000%	0.59
	VoIP1_Rev	bidi	20733	76.05	76.70	0	0	0.0000%	0.0000%	0.66
		dwn	20733	76.05	76.07	0	0	0.0000%	0.0000%	0.02
		up	20733	76.05	76.70	0	0	0.0000%	0.0000%	0.65
Gw5t	**HDTV1_Down**	bidi	451508	86.76	90.35	0	0	0.0000%	0.0000%	3.60
		dwn	451508	86.76	90.38	0	0	0.0000%	0.0000%	3.62
		up	451508	86.76	89.93	0	0	0.0000%	0.0000%	3.17

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
	SDTV1_Down	bidi	113079	86.76	89.87	0	0	0.0000%	0.0000%	3.11
		dwn	113079	86.76	89.89	0	0	0.0000%	0.0000%	3.14
		up	113079	86.76	89.45	0	0	0.0000%	0.0000%	2.69
	VoIP1_Fwd	bidi	20735	86.05	86.77	0	0	0.0000%	0.0000%	0.72
		dwn	20735	86.05	86.77	0	0	0.0000%	0.0000%	0.72
		up	20735	86.05	86.77	0	0	0.0000%	0.0000%	0.72
	VoIP1_Rev	bidi	20733	86.05	86.71	0	0	0.0000%	0.0000%	0.67
		dwn	20733	86.05	86.07	0	0	0.0000%	0.0000%	0.02
		up	20733	86.05	86.72	0	0	0.0000%	0.0000%	0.67
Gw6t	**HDTV1_Down**	bidi	451494	101.77	115.62	3	0	0.0007%	0.0000%	13.85
		dwn	451492	101.77	115.62	5	0	0.0011%	0.0000%	13.85
		up	451494	101.77	115.31	3	0	0.0007%	0.0000%	13.54
	SDTV1_Down	bidi	113077	101.77	115.27	0	0	0.0000%	0.0000%	13.50
		dwn	113077	101.77	115.26	0	0	0.0000%	0.0000%	13.49
		up	113076	101.77	114.96	1	0	0.0009%	0.0000%	13.19
	SDTV2_Down	bidi	114897	101.77	115.56	1	0	0.0009%	0.0000%	13.80
		dwn	114897	101.77	115.56	1	0	0.0009%	0.0000%	13.79
		up	114897	101.77	115.25	1	0	0.0009%	0.0000%	13.49
	VoIP1_Fwd	bidi	20735	101.05	101.78	0	0	0.0000%	0.0000%	0.73
		dwn	20735	101.05	101.78	0	0	0.0000%	0.0000%	0.73
		up	20735	101.05	101.78	0	0	0.0000%	0.0000%	0.73
	VoIP1_Rev	bidi	20733	101.05	101.72	0	0	0.0000%	0.0000%	0.67
		dwn	20733	101.05	101.07	0	0	0.0000%	0.0000%	0.02
		up	20733	101.05	101.73	0	0	0.0000%	0.0000%	0.68

Table G.2 – Simulation results summary GPON managed flows

Case	File	Scen	n1	min	max	drop	seq	drop %	seq %	max-min
Gw7t	HDTV1_Down	bidi	451472	126.78	140.45	7	0	0.0016%	0.0000%	13.68
		dwn	451475	126.78	140.73	4	0	0.0009%	0.0000%	13.95
		up	451473	126.78	140.32	6	0	0.0013%	0.0000%	13.54
	SDTV1_Down	bidi	113072	126.78	140.10	0	0	0.0000%	0.0000%	13.32
		dwn	113071	126.78	140.37	1	0	0.0009%	0.0000%	13.59
		up	113071	126.78	139.97	1	0	0.0009%	0.0000%	13.19
	SDTV2_Down	bidi	114897	126.78	140.40	1	0	0.0009%	0.0000%	13.62
		dwn	114897	126.78	140.67	1	0	0.0009%	0.0000%	13.89
		up	114897	126.78	140.27	1	0	0.0009%	0.0000%	13.49
	VoIP1_Fwd	bidi	20734	126.05	126.79	0	0	0.0000%	0.0000%	0.74
		dwn	20734	126.05	126.79	0	0	0.0000%	0.0000%	0.74
		up	20734	126.05	126.79	0	0	0.0000%	0.0000%	0.74
	VoIP1_Rev	bidi	20732	126.05	126.74	0	0	0.0000%	0.0000%	0.69
		dwn	20732	126.05	126.07	0	0	0.0000%	0.0000%	0.02
		up	20732	126.05	126.74	0	0	0.0000%	0.0000%	0.69
Gw8t	HDTV1_Down	bidi	451455	152.08	158.84	5	0	0.0011%	0.0000%	6.76
		dwn	451454	152.08	159.21	6	0	0.0013%	0.0000%	7.13
		up	451456	152.08	158.58	4	0	0.0009%	0.0000%	6.50
	VoIP1_Fwd	bidi	20733	151.06	152.09	0	0	0.0000%	0.0000%	1.02
		dwn	20733	151.06	152.09	0	0	0.0000%	0.0000%	1.03
		up	20733	151.06	152.09	0	0	0.0000%	0.0000%	1.02
	VoIP1_Rev	bidi	20731	151.14	153.67	0	0	0.0000%	0.0000%	2.53
		dwn	20731	151.14	151.27	0	0	0.0000%	0.0000%	0.13
		up	20731	151.14	153.67	0	0	0.0000%	0.0000%	2.53

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Dp1	up	bidi	Partly Mgd	2.9	2.9	0.1	0.0	3.0	229.8	232.8	29.1	11.2	285.7	240.4	22.4	528.8
Dp1	dwn	bidi	Partly Mgd	5.4	5.5	2.1	0.0	12.4	26.1	23.2	42.9	11.0	1146.9	23.2	22.2	24.2
Dp2	up	bidi	Partly Mgd	1.4	1.5	0.1	0.0	1.5	416.3	422.6	95.1	26.3	682.2	428.7	52.7	1094.3
Dp2	dwn	bidi	Partly Mgd	2.2	1.9	1.6	0.0	9.1	46.2	31.0	130.8	26.0	2952.3	53.3	52.3	54.4
Dp3	up	bidi	Partly Mgd	1.5	1.5	0.1	0.0	1.5	447.1	459.7	98.1	51.3	785.9	487.3	102.8	1129.2
Dp3	dwn	bidi	Partly Mgd	2.0	1.6	1.5	0.0	8.0	77.8	61.6	100.7	51.1	2315.3	103.4	102.3	104.4
Dp4	up	bidi	Partly Mgd	0.7	0.7	0.2	0.0	1.0	117.9	98.6	74.1	76.5	1498.5	197.6	153.1	763.3
Dp4	dwn	bidi	Partly Mgd	0.7	0.7	0.3	0.0	2.7	98.7	89.3	40.1	78.3	1682.1	154.8	154.6	155.0
Dp5	up	bidi	Partly Mgd	0.5	0.5	0.3	0.0	1.0	162.6	137.8	104.4	101.5	3561.3	327.5	203.2	1495.3
Dp5	dwn	bidi	Partly Mgd	0.6	0.6	0.3	0.0	2.7	225.9	158.7	224.0	101.2	4679.8	203.6	202.6	204.7
Dp6	up	bidi	Partly Mgd	0.5	0.5	0.2	0.0	1.0	249.0	219.9	100.5	201.5	1638.5	449.0	403.1	1058.4
Dp6	dwn	bidi	Partly Mgd	0.5	0.4	0.3	0.0	2.1	226.1	211.2	65.1	201.1	1994.4	404.4	404.3	404.6
Du1	up	bidi	Unmgd	2.9	2.9	0.1	0.0	3.0	231.0	233.2	29.0	11.2	280.1	240.5	22.4	533.0
Du1	dwn	bidi	Unmgd	5.5	5.6	2.2	0.0	16.3	28.0	21.1	75.7	11.0	1485.2	23.2	22.2	24.2
Du2	up	bidi	Unmgd	1.5	1.5	0.1	0.0	1.5	397.8	401.7	101.8	26.3	676.5	409.0	52.7	1055.2
Du2	dwn	bidi	Unmgd	2.3	1.9	1.7	0.0	10.0	34.3	29.5	32.6	26.0	696.2	53.3	52.3	54.4
Du3	up	bidi	Unmgd	1.5	1.5	0.1	0.0	1.5	420.3	419.0	100.8	51.3	815.7	455.9	102.8	1119.7
Du3	dwn	bidi	Unmgd	2.0	1.7	1.5	0.0	9.7	66.6	56.1	70.3	51.1	1910.0	103.4	102.3	104.4
Du4	up	bidi	Unmgd	0.7	0.7	0.2	0.0	1.0	105.7	94.4	36.3	76.5	441.8	178.4	153.1	447.4
Du4	dwn	bidi	Unmgd	0.8	0.7	0.4	0.0	3.6	85.9	82.1	15.0	76.1	626.3	153.6	152.5	154.6
Du5	up	bidi	Unmgd	0.6	0.6	0.2	0.0	1.0	130.2	119.3	34.8	101.5	516.4	230.0	203.2	580.1
Du5	dwn	bidi	Unmgd	0.6	0.6	0.3	0.0	2.8	116.5	111.2	18.9	101.2	448.8	203.6	202.6	204.7
Du6	up	bidi	Unmgd	0.5	0.5	0.2	0.0	1.0	232.2	219.4	47.2	201.5	858.8	430.7	403.1	1201.3

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Du6	dwn	bidi	Unmgd	0.6	0.5	0.4	0.0	3.4	215.8	206.4	42.5	201.1	1256.2	403.5	402.5	404.6
Du7	up	bidi	Unmgd	0.5	0.4	0.2	0.0	1.0	285.6	269.4	65.6	251.5	1429.8	530.5	503.2	1344.7
Du7	dwn	bidi	Unmgd	0.5	0.4	0.4	0.1	3.0	271.7	259.3	49.5	255.3	1108.8	504.6	504.7	504.7
Dw1	up	bidi	Well Mgd	2.9	2.9	0.1	0.0	3.0	230.3	233.4	29.4	11.2	286.5	239.8	22.4	527.9
Dw1	dwn	bidi	Well Mgd	5.4	5.6	2.2	0.0	13.2	24.8	20.8	54.4	11.0	1219.3	23.2	22.2	24.2
Dw2	up	bidi	Well Mgd	2.9	2.9	0.1	0.0	3.0	242.5	246.3	32.5	26.2	302.5	266.7	52.4	537.0
Dw2	dwn	bidi	Well Mgd	5.1	5.2	2.3	0.0	14.8	38.3	34.5	39.5	26.0	1040.6	53.2	52.2	54.2
Dw3	up	bidi	Well Mgd	2.9	2.9	0.1	0.0	3.0	259.2	263.9	36.8	51.2	328.7	308.5	102.4	547.2
Dw3	dwn	bidi	Well Mgd	4.2	4.5	2.2	0.0	11.3	64.6	62.9	25.0	51.0	746.8	103.2	102.2	104.2
Dw4	up	bidi	Well Mgd	1.9	1.9	0.1	0.0	2.0	391.8	401.0	61.3	76.2	584.3	462.8	152.6	841.1
Dw4	dwn	bidi	Well Mgd	2.5	2.1	1.9	0.0	8.4	106.1	93.1	121.9	76.0	2795.9	153.3	152.3	154.3
Dw5	up	bidi	Well Mgd	1.0	1.0	0.1	0.0	1.0	724.9	740.4	139.2	86.5	1258.6	781.2	173.1	1728.2
Dw5	dwn	bidi	Well Mgd	1.4	0.9	1.4	0.0	6.1	137.6	92.5	236.5	-49.8	3944.4	174.5	174.5	174.5
Dw6	up	bidi	Well Mgd	1.0	1.0	0.1	0.0	1.0	708.8	729.0	166.7	101.5	1488.6	783.3	203.1	1861.8
Dw6	dwn	bidi	Well Mgd	1.3	1.0	1.1	0.0	4.9	166.1	114.7	229.6	101.0	4280.4	203.5	202.5	204.5
Dw7	up	bidi	Well Mgd	1.0	1.0	0.1	0.0	1.0	717.5	729.1	175.9	126.5	1351.7	810.2	253.1	1798.6
Dw7	dwn	bidi	Well Mgd	1.4	1.0	1.2	0.0	5.0	184.4	141.7	177.0	127.0	3464.5	254.0	253.4	254.5
Dw8	up	bidi	Well Mgd	1.0	1.0	0.1	0.0	1.0	704.5	727.2	215.4	151.5	1631.3	850.7	303.1	2007.3
Dw8	dwn	bidi	Well Mgd	1.2	1.3	0.7	0.0	2.2	266.9	203.7	266.6	152.0	5150.4	569.0	304.6	833.4
Dp1	dwn	dwn	Partly Mgd	20.2	20.0	1.0	0.0	24.3	36.0	37.2	8.5	11.0	95.0	40.2	22.2	58.2
Dp2	dwn	dwn	Partly Mgd	5.9	5.8	1.6	0.0	14.3	27.6	27.5	1.4	26.0	100.0	91.9	52.3	131.4
Dp3	dwn	dwn	Partly Mgd	3.7	3.5	1.4	0.0	8.9	54.3	53.2	6.5	51.5	294.3	166.6	102.8	230.4
Dp4	dwn	dwn	Partly Mgd	0.8	0.7	0.3	0.0	6.0	92.4	83.4	38.9	77.3	1794.5	250.7	153.7	347.6

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Dp5	dwn	dwn	Partly Mgd	0.6	0.6	0.3	0.0	3.0	222.5	166.6	212.5	102.0	4681.6	326.6	203.5	449.7
Dp6	dwn	dwn	Partly Mgd	0.5	0.4	0.3	0.0	2.7	222.3	207.5	63.1	203.2	1826.2	634.6	404.6	864.6
Du1	dwn	dwn	Unmgd	24.2	24.2	1.0	0.0	24.3	30.6	31.6	8.1	11.0	43.7	40.2	22.2	58.2
Du2	dwn	dwn	Unmgd	5.9	5.8	1.7	0.0	14.1	27.3	27.3	0.3	26.0	36.1	91.9	52.3	131.4
Du3	dwn	dwn	Unmgd	4.0	3.6	1.7	0.0	12.1	53.0	52.7	0.9	51.1	73.2	166.4	102.3	230.4
Du4	dwn	dwn	Unmgd	0.8	0.8	0.3	0.0	6.0	82.3	79.2	9.0	76.1	166.2	250.1	152.5	347.6
Du5	dwn	dwn	Unmgd	0.7	0.6	0.3	0.0	3.0	111.7	107.3	14.8	101.2	276.7	326.1	202.6	449.7
Du6	dwn	dwn	Unmgd	0.5	0.4	0.4	0.0	4.9	213.4	204.2	47.3	201.1	1372.2	633.5	402.5	864.6
Du7	dwn	dwn	Unmgd	0.5	0.4	0.4	0.0	3.0	268.6	257.2	58.5	251.2	1506.6	795.1	502.6	1087.7
Dw1	dwn	dwn	Well Mgd	25.2	25.3	1.2	0.0	33.3	28.2	29.0	7.8	11.0	41.9	40.2	22.2	58.2
Dw2	dwn	dwn	Well Mgd	21.6	23.2	2.8	0.0	33.4	34.8	29.5	9.4	26.0	59.5	91.7	52.2	131.2
Dw3	dwn	dwn	Well Mgd	13.8	14.5	4.2	0.0	23.3	59.2	52.9	12.5	51.0	231.4	166.2	102.2	230.2
Dw4	dwn	dwn	Well Mgd	9.2	9.8	2.0	0.0	16.4	104.1	101.2	25.8	76.0	550.3	245.5	152.8	338.3
Dw5	dwn	dwn	Well Mgd	3.6	3.4	1.3	0.0	6.3	92.2	88.5	14.6	86.7	238.5	278.3	173.1	383.6
Dw6	dwn	dwn	Well Mgd	3.1	3.1	1.0	0.0	6.1	131.6	107.8	47.9	101.8	606.8	325.4	203.2	447.6
Dw7	dwn	dwn	Well Mgd	2.7	2.6	1.0	0.0	5.8	151.6	130.9	46.2	126.9	665.7	403.0	253.4	552.6
Dw8	dwn	dwn	Well Mgd	1.9	2.1	0.4	0.0	9.0	289.7	271.2	120.5	151.1	806.9	476.0	302.5	649.6
Dp1	up	up	Partly Mgd	3.0	3.0	0.1	0.0	3.0	229.8	233.7	29.7	11.2	283.1	238.1	22.4	495.8
Dp2	up	up	Partly Mgd	1.5	1.5	0.0	0.0	1.5	412.3	413.8	99.0	26.3	705.0	423.2	52.7	1038.6
Dp3	up	up	Partly Mgd	1.5	1.5	0.0	0.0	1.5	420.6	433.4	118.1	51.3	802.3	455.8	102.8	1127.5
Dp4	up	up	Partly Mgd	0.7	0.7	0.2	0.0	1.0	118.7	95.8	63.0	76.5	1370.3	195.1	153.1	774.0
Dp5	up	up	Partly Mgd	0.6	0.5	0.2	0.0	1.0	145.3	125.1	80.7	101.5	3339.8	251.2	203.2	714.2
Dp6	up	up	Partly Mgd	0.5	0.4	0.2	0.0	1.0	232.5	219.5	47.7	201.5	1395.7	432.5	403.1	1324.7

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Du1	up	up	Unmgd	3.0	3.0	0.1	0.0	3.0	230.6	235.2	29.5	11.2	285.1	239.0	22.4	496.4
Du2	up	up	Unmgd	1.5	1.5	0.0	0.0	1.5	398.5	410.6	110.2	26.3	663.4	410.7	52.7	1009.1
Du3	up	up	Unmgd	1.5	1.5	0.0	0.0	1.5	409.7	423.5	118.3	51.3	742.5	444.7	102.8	1081.2
Du4	up	up	Unmgd	0.7	0.7	0.2	0.0	1.0	106.7	94.3	50.2	76.5	854.3	179.8	153.1	1345.7
Du5	up	up	Unmgd	0.6	0.6	0.2	0.0	1.0	134.3	119.3	63.7	101.5	1028.9	233.1	203.2	1646.0
Du6	up	up	Unmgd	0.5	0.5	0.2	0.0	1.0	236.1	219.4	73.5	201.5	1394.1	430.3	403.1	1324.7
Du7	up	up	Unmgd	0.5	0.4	0.2	0.0	1.0	287.7	269.4	76.4	251.5	1597.6	530.4	503.2	1664.7
Dw1	up	up	Well Mgd	3.0	3.0	0.1	0.0	3.0	230.2	233.5	29.8	11.2	283.1	238.5	22.4	495.5
Dw2	up	up	Well Mgd	3.0	3.0	0.1	0.0	3.0	242.6	245.9	32.1	26.2	298.9	265.7	52.4	523.8
Dw3	up	up	Well Mgd	3.0	3.0	0.1	0.0	3.0	257.8	265.1	39.0	51.2	384.2	303.7	102.4	507.8
Dw4	up	up	Well Mgd	2.0	2.0	0.1	0.0	2.0	392.9	405.7	62.1	76.2	539.8	458.7	152.6	783.5
Dw5	up	up	Well Mgd	1.0	1.0	0.0	0.0	1.0	697.1	726.5	171.2	86.5	1321.6	742.5	173.1	1704.8
Dw6	up	up	Well Mgd	1.0	1.0	0.0	0.0	1.0	682.7	700.2	182.5	101.5	1331.3	739.9	203.1	1727.4
Dw7	up	up	Well Mgd	1.0	1.0	0.1	0.0	1.0	699.2	721.8	187.6	126.5	1385.0	779.0	253.1	1457.6
Dw8	up	up	Well Mgd	1.0	1.0	0.0	0.0	1.0	742.6	773.3	188.0	151.5	1404.1	842.2	303.1	1432.4
Gp1	up	bidi	Partly Mgd	32.8	33.3	2.5	0.0	35.0	18.5	15.5	6.0	11.0	32.4	38.9	22.1	76.7
Gp1	dwn	bidi	Partly Mgd	29.7	30.4	2.1	0.0	32.9	25.2	26.3	6.7	11.0	40.7	23.0	22.1	24.0
Gp2	up	bidi	Partly Mgd	14.8	14.9	0.5	0.0	15.1	60.0	59.1	7.9	26.1	76.8	80.3	52.1	136.6
Gp2	dwn	bidi	Partly Mgd	15.4	15.0	3.9	0.0	27.1	35.3	35.2	3.3	26.0	112.0	53.1	52.1	54.1
Gp3	up	bidi	Partly Mgd	12.4	12.4	3.2	0.0	22.0	60.2	60.0	4.4	51.0	124.9	116.2	102.1	237.1
Gp3	dwn	bidi	Partly Mgd	16.4	17.7	4.3	0.0	22.0	65.6	65.2	6.8	51.0	114.7	103.1	102.1	104.1
Gp4	up	bidi	Partly Mgd	4.9	4.9	0.2	0.0	5.0	181.9	181.6	26.2	76.1	263.1	229.1	152.3	442.4
Gp4	dwn	bidi	Partly Mgd	4.3	4.6	1.5	0.0	10.4	101.4	102.4	19.6	76.0	699.3	153.5	152.9	154.2

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Gp5	up	bidi	Partly Mgd	1.9	1.9	0.2	0.0	2.0	353.9	364.4	90.9	101.2	1408.0	515.6	202.6	1672.1
Gp5	dwn	bidi	Partly Mgd	2.0	2.2	0.9	0.0	4.9	214.0	193.6	123.0	101.1	2211.1	203.3	202.3	204.4
Gp6	up	bidi	Partly Mgd	3.9	4.9	1.7	0.0	5.1	272.4	262.7	76.4	201.1	800.0	421.3	402.3	954.2
Gp6	dwn	bidi	Partly Mgd	2.5	2.2	1.9	0.0	10.7	248.8	212.4	176.6	201.1	2331.4	403.6	403.0	404.2
Gu1	up	bidi	Unmgd	33.8	34.7	2.4	0.0	35.0	24.0	25.3	5.4	11.0	32.6	37.8	22.1	64.1
Gu1	dwn	bidi	Unmgd	33.5	34.6	2.6	0.0	35.0	18.3	16.2	5.2	11.0	33.0	23.0	22.0	24.0
Gu2	up	bidi	Unmgd	14.8	14.9	0.6	0.0	15.1	61.0	60.1	7.6	26.1	78.1	80.7	52.1	133.8
Gu2	dwn	bidi	Unmgd	15.3	14.9	4.0	0.0	30.2	33.8	33.6	2.7	26.0	78.2	53.1	52.1	54.1
Gu3	up	bidi	Unmgd	15.6	16.3	3.9	0.0	23.0	63.0	63.2	5.0	51.0	116.9	108.7	102.1	233.6
Gu3	dwn	bidi	Unmgd	16.0	16.6	3.6	0.0	23.3	64.3	64.4	5.9	51.0	184.5	103.1	102.1	104.1
Gu4	up	bidi	Unmgd	4.9	5.0	0.3	0.0	5.1	186.0	186.2	26.4	76.1	236.9	225.9	152.3	406.3
Gu4	dwn	bidi	Unmgd	4.5	4.9	1.8	0.0	10.5	96.0	95.5	36.4	76.0	1013.9	153.1	152.1	154.2
Gu5	up	bidi	Unmgd	2.0	2.0	0.1	0.0	2.1	395.7	402.8	67.0	101.2	689.6	507.4	202.6	956.3
Gu5	dwn	bidi	Unmgd	2.2	2.2	1.4	0.0	5.0	152.8	147.2	78.3	101.1	2393.0	203.3	202.3	204.4
Gu6	up	bidi	Unmgd	3.9	5.0	1.8	0.0	5.1	271.2	257.8	81.7	201.1	815.3	417.3	402.3	909.3
Gu6	dwn	bidi	Unmgd	2.4	1.9	2.0	0.0	9.7	256.9	208.8	218.3	201.1	2764.6	403.1	402.1	404.2
Gu7	up	bidi	Unmgd	1.9	2.0	0.3	0.0	2.1	512.1	530.0	115.0	251.2	1265.4	571.1	502.6	1459.5
Gu7	dwn	bidi	Unmgd	1.6	1.0	1.5	0.0	5.1	338.9	269.4	241.5	251.1	3528.8	503.3	502.3	504.4
Gw1	up	bidi	Well Mgd	24.8	25.0	0.9	0.0	25.1	32.8	33.7	4.8	11.0	40.6	44.7	22.1	66.3
Gw1	dwn	bidi	Well Mgd	17.7	17.3	3.2	0.0	36.7	19.8	19.9	1.1	11.0	24.0	23.0	22.0	24.0
Gw2	up	bidi	Well Mgd	24.2	24.9	2.2	0.0	25.3	43.9	42.2	6.0	26.0	57.5	61.1	52.1	131.1
Gw2	dwn	bidi	Well Mgd	16.2	15.7	4.8	0.0	36.1	33.1	33.1	2.4	26.0	90.8	53.0	52.0	54.0
Gw3	up	bidi	Well Mgd	12.2	12.2	4.0	0.0	23.0	58.5	57.6	10.6	51.0	473.4	115.7	102.1	279.0

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Gw3	dwn	bidi	Well Mgd	15.5	16.9	4.6	0.0	23.2	64.9	63.2	14.7	51.3	455.2	103.2	102.3	104.1
Gw4	up	bidi	Well Mgd	11.6	12.9	4.7	0.0	17.4	84.4	82.9	7.5	76.0	255.2	158.3	152.1	336.5
Gw4	dwn	bidi	Well Mgd	12.1	13.3	4.3	0.0	18.8	84.9	83.3	8.2	76.0	331.7	153.0	152.0	154.1
Gw5	up	bidi	Well Mgd	9.0	9.4	2.7	0.0	14.8	97.2	96.5	20.1	86.0	886.5	181.8	172.1	376.7
Gw5	dwn	bidi	Well Mgd	10.7	11.9	3.4	0.0	14.1	101.5	100.3	10.4	86.0	359.8	173.3	172.5	174.1
Gw6	up	bidi	Well Mgd	7.0	7.3	2.7	0.0	13.5	111.2	108.3	31.8	101.0	1188.9	212.5	202.1	622.4
Gw6	dwn	bidi	Well Mgd	9.2	10.4	3.0	0.0	12.4	115.6	112.8	13.2	101.3	459.9	203.2	202.4	204.1
Gw7	up	bidi	Well Mgd	6.5	6.7	3.6	0.0	13.6	134.8	131.5	18.9	126.0	588.2	261.3	252.1	551.1
Gw7	dwn	bidi	Well Mgd	7.6	8.1	2.9	0.0	13.2	138.6	135.1	14.0	126.4	588.3	253.4	252.7	254.1
Gw8	up	bidi	Well Mgd	4.2	4.9	1.4	0.0	5.1	232.4	230.5	54.7	151.1	850.2	328.5	302.3	792.5
Gw8	dwn	bidi	Well Mgd	3.2	3.4	1.9	0.0	7.2	194.0	180.7	106.5	151.4	2576.8	303.3	302.5	304.2
Gp1	dwn	dwn	Partly Mgd	31.3	31.1	1.4	0.0	35.4	30.5	31.2	4.3	11.0	45.6	40.0	22.0	58.1
Gp2	dwn	dwn	Partly Mgd	30.4	31.1	4.2	0.0	35.4	39.6	39.0	5.9	26.0	73.5	91.6	52.1	131.1
Gp3	dwn	dwn	Partly Mgd	19.3	20.9	4.7	0.0	22.9	62.3	61.5	7.4	51.0	165.3	166.1	102.1	230.1
Gp4	dwn	dwn	Partly Mgd	10.2	10.8	2.4	0.0	13.3	109.0	108.6	21.9	76.1	490.7	249.6	152.1	347.2
Gp5	dwn	dwn	Partly Mgd	3.0	2.8	0.7	0.0	5.0	293.0	296.1	153.2	101.1	3366.0	325.8	202.3	449.4
Gp6	dwn	dwn	Partly Mgd	4.5	4.6	2.9	0.0	11.1	215.1	210.4	20.0	201.1	935.0	633.1	402.1	864.2
Gu1	dwn	dwn	Unmgd	35.3	35.4	1.2	0.0	35.4	29.2	30.2	3.9	11.0	33.7	40.0	22.0	58.1
Gu2	dwn	dwn	Unmgd	34.2	35.4	4.8	0.0	35.4	38.5	37.5	5.1	26.0	48.2	91.6	52.1	131.1
Gu3	dwn	dwn	Unmgd	20.2	22.0	5.0	0.0	23.3	57.7	55.6	4.8	51.0	71.0	166.1	102.1	230.1
Gu4	dwn	dwn	Unmgd	12.8	14.7	3.9	0.0	15.1	86.4	84.2	8.2	76.0	107.3	249.6	152.1	347.2
Gu5	dwn	dwn	Unmgd	5.0	5.0	0.4	0.0	5.1	205.2	209.9	35.6	101.1	282.7	325.8	202.3	449.4
Gu6	dwn	dwn	Unmgd	4.9	4.3	3.6	0.0	12.2	211.1	207.5	18.5	201.1	935.1	633.1	402.1	864.2

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Gu7	dwn	dwn	Unmgd	3.4	3.9	1.7	0.0	5.1	284.8	280.4	27.6	251.1	585.6	794.8	502.3	1087.4
Gw1	dwn	dwn	Well Mgd	42.5	42.5	1.6	0.0	50.5	25.4	26.3	3.5	11.1	29.6	40.2	22.4	58.1
Gw2	dwn	dwn	Well Mgd	38.5	40.5	6.5	0.0	41.2	32.9	32.3	3.7	26.0	39.8	91.5	52.0	131.1
Gw3	dwn	dwn	Well Mgd	17.3	19.1	4.9	0.0	23.3	63.0	59.6	13.9	51.0	408.4	166.0	102.0	230.1
Gw4	dwn	dwn	Well Mgd	12.8	14.3	4.6	0.0	19.1	86.1	87.0	7.0	76.0	257.3	245.0	152.0	338.1
Gw5	dwn	dwn	Well Mgd	11.4	13.2	3.7	0.0	14.1	97.7	94.7	9.7	86.0	355.9	277.6	172.1	383.1
Gw6	dwn	dwn	Well Mgd	9.6	11.1	3.2	0.0	13.1	113.5	110.8	12.7	101.3	410.0	324.7	202.3	447.1
Gw7	dwn	dwn	Well Mgd	7.5	8.2	2.9	0.0	12.1	138.2	135.1	17.8	126.0	816.2	402.0	252.1	552.1
Gw8	dwn	dwn	Well Mgd	5.8	7.1	2.4	0.0	8.6	186.2	184.5	24.9	151.5	508.4	475.9	302.6	649.2
Gp1	up	up	Partly Mgd	35.3	35.3	1.2	0.0	35.4	29.3	30.3	3.9	11.0	33.7	32.7	22.1	105.1
Gp2	up	up	Partly Mgd	15.1	15.1	0.5	0.0	15.2	64.5	63.3	7.5	26.1	79.0	76.1	52.1	206.1
Gp3	up	up	Partly Mgd	20.0	22.0	5.0	0.0	23.3	59.6	60.6	6.1	51.0	78.8	105.8	102.1	365.1
Gp4	up	up	Partly Mgd	5.0	5.0	0.2	0.0	5.1	191.1	190.3	25.8	76.1	241.9	221.3	152.3	540.2
Gp5	up	up	Partly Mgd	2.0	2.0	0.1	0.0	2.0	420.7	423.5	60.2	101.2	1334.3	505.5	202.6	883.2
Gp6	up	up	Partly Mgd	3.9	5.0	1.8	0.0	5.1	265.8	245.6	83.9	201.1	767.8	417.8	402.3	1324.2
Gu1	up	up	Unmgd	35.3	35.4	1.2	0.0	35.4	29.3	30.3	3.9	11.0	33.8	32.6	22.1	105.1
Gu2	up	up	Unmgd	15.1	15.2	0.5	0.0	15.2	64.5	63.2	7.5	26.1	78.8	76.0	52.1	206.1
Gu3	up	up	Unmgd	20.3	22.0	5.1	0.0	23.4	60.5	62.9	5.9	51.0	80.8	106.0	102.1	365.1
Gu4	up	up	Unmgd	5.0	5.0	0.2	0.0	5.1	191.0	189.4	24.8	76.1	237.1	221.8	152.3	540.2
Gu5	up	up	Unmgd	2.0	2.0	0.1	0.0	2.1	416.7	418.5	56.6	101.2	708.0	500.1	202.6	701.4
Gu6	up	up	Unmgd	3.9	5.0	1.8	0.0	5.1	268.7	253.3	84.0	201.1	767.3	418.1	402.3	1324.2
Gu7	up	up	Unmgd	2.0	2.0	0.2	0.0	2.1	518.7	539.0	110.7	251.2	1306.5	575.9	502.6	1664.4
Gw1	up	up	Well Mgd	25.2	25.3	1.0	0.0	25.3	37.2	38.4	4.8	11.0	42.6	43.3	22.1	105.1

Table G.3 – Simulation results summary iPerf flow statistics

				iPerf bandwidth (Mbit/s)					iPerf one-way delay (ms)					RTT (ms)		
Case	**dir**	**Scen.**	**Mgmt.**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Min**	**Max**	**Avg**	**Min**	**Max**
Gw2	up	up	Well Mgd	24.4	25.3	3.7	0.0	25.3	45.9	45.3	6.7	26.0	57.9	55.3	52.1	206.1
Gw3	up	up	Well Mgd	19.8	21.9	5.0	0.0	23.3	59.1	58.2	6.2	51.0	87.2	108.8	102.1	426.7
Gw4	up	up	Well Mgd	13.1	14.1	4.1	0.0	17.4	83.4	80.4	8.1	76.0	253.5	156.8	152.1	523.1
Gw5	up	up	Well Mgd	11.5	12.4	3.7	0.0	15.1	94.2	91.9	9.5	86.0	355.8	177.2	172.1	597.1
Gw6	up	up	Well Mgd	9.5	11.0	3.3	0.0	13.1	108.4	104.5	11.1	101.0	420.9	207.0	202.1	690.1
Gw7	up	up	Well Mgd	7.6	8.1	3.2	0.0	12.9	134.1	129.5	17.4	126.0	664.7	258.4	252.1	849.1
Gw8	up	up	Well Mgd	4.4	5.0	1.3	0.0	5.1	239.1	243.1	55.5	151.1	617.5	322.8	302.3	1001.2

Annex H

Electronic attachment

(This annex forms an integral part of this Recommendation.)

The electronic attachment is downloadable from the ITU website and is a part of this Recommendation. This file may be converted into a physical DVD data disk using any of the available software tools or packages. The data disk contents are described on the disk itself and has the following subdirectories:

ISO 9660 format file that is downloadable from the ITU website:

– ./tc – Impairment combination standard test cases (clause 7.2);
– ./simulator.tar.gz – Ready-to-build distribution (Annexes B and H);
– ./pcap – Input packet capture files of interfering traffic (Annex C);
– ./D – DSL technology simulator output (Annex D);
– ./G – GPON technology simulator output (Annex D).

Appendix I

TCP considerations

(This appendix does not form an integral part of this Recommendation.)

This appendix is intended to explain some of the complexity added to the model due to the introduction of TCP. With TCP as a Layer 4 Protocol, wide variations in performance can result. This variation in performance is implementation dependent.

With UDP transport of video using the MPEG transport mechanism (IPTV), encoder/decoder performance and network quality are the primary influences on the quality that users will experience. However, the particular implementation of UDP has little impact. This is because UDP is intended to be a simple protocol only for providing a sending and receiving port number, length value and header checksum. All other influences of the user's experience result from factors such as network bandwidth available, network loss and delay characteristics, as well as processor efficiency in the encoder and decoder.

To the contrary, the performance of video transport over TCP can be very implementation dependent. Specifically, a particular Windows TCP stack may behave differently than a particular distribution of Linux. In addition, TCP implementations involve algorithms that make the source and destination dynamic, reacting to network conditions, round-trip-times (RTT) and available bandwidth. TCP flows will be negatively impacted by other traffic flows. If those flows are also TCP based, predicting the resulting conditions can be very complex. Even when the TCP session is created, the client and server modules will negotiate down to a common set of operating parameters. As a result, multiple clients may receive video flows differently even with the same set of network conditions and the same server.

TCP stacks have evolved through several generations: Tahoe, Reno, New Reno, Compound and Cubic. In each generation, capabilities and options were added which could be tools for the developer of the TCP stack. We will review some of these options to help clarify the sources of some of these variations.

Selective acknowledgement (SACK): If a packet is dropped, the receiving TCP entity would indicate it was still waiting for the dropped packet. That is, if packet n was dropped, and packet n+1 arrived, the receiver would indicate it was still expecting packet n+1 (The receiver always acknowledges a packet by announcing what packet it is now expecting). Even if packet n+2 and n+3 arrived, their acknowledgements would indicate that the receiver was expecting to receive n+1. The standard rule for TCP operation dictates that after three duplicate acknowledgements (four identical acknowledgements), the missing packet is declared to be lost and is retransmitted. In early implementations of TCP, our scenario would have seen the transmitting station resend packets n+1, n+2, and n+3. This duplication of transmitted packets wasted time and bandwidth. With a TCP implementation that supports SACK, only packet n+1 needs to be retransmitted. Many of the currently available TCP implementations support this.

Fast transmit/Fast recovery: Normally, TCP uses a timer to determine when to retransmit a lost packet (for which three duplicate packets have not been received). In today's Internet/intranets with long, high speed links, many packets can be sent but not acknowledged before a retransmitted packet is correctly acknowledged. Fast retransmit allows for faster transmission of lost packets and faster recovery to the normal send rate. Normal policy for a sending station that determines that a packet has been lost is to drastically reduce the number of packets sent as a block and to lower the rate of increase of transmission. The rapidity with which the transmission level recovers varies considerably among the TCP variants. In older implementations the recovery is linear. Newer implementations use scaling that is likely to be more aggressive, reducing the time to reach the maximum estimated transmission rate.

Window scaling: The window value, more properly window receive value, represents the number of bytes a receiver is announcing that they can properly receive. That is, it represents the available size of the receive buffer. In early implementations of TCP, the value was restricted to 64k bytes because the field in the TCP header that held the value was 16 bits. With modern high speed networks, this value has posed a significant limitation in filling links with packets. For example, a one gigabit/s link might be filled to 60-80 Mb/s before the window value was reached. Since the value of the window is negotiated at session start as part of the three-way handshake, either the sender or receiver can limit the value to 64k or below. To overcome this limitation, window scaling was introduced in newer implementations of TCP. In this procedure, the stations can negotiate the use of a window which is a large as 32 times 64k bytes. This significantly increases the ability of a station to saturate a link.

Appendix II

Hybrid box plots and violin plots

(This appendix does not form an integral part of this Recommendation.)

Many of the result plots in the electronic attachments are a hybrid of a box plot overlaying a violin plot [b-Violin]. These plots are an efficient way to visually display a lot of statistical information.

A violin plot can be described as a rotated probability density function that has also been reflected across a vertical axis. In many cases the resulting shape looks like a violin.

A box plot shows similar statistical information and is drawn as a box with whiskers and dots. The box is centred on the median value, and the upper and lower limits of the box represent the 25% and 75% quartiles of the data around the median. The whisker lines extend from the ends of the box and represent the 5%-95% ranges. The dots are drawn beyond the ends of the whiskers and represent the 1% outlier points of the data.

Figure II.1 compares a probability density plot, violin plot and a box plot for 1000 data points generated from a normally distributed pseudo-random number generator.

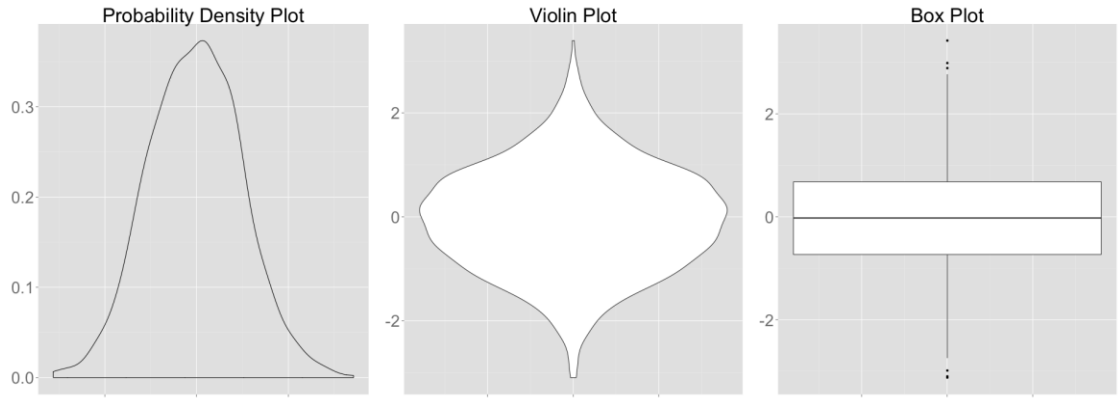


Figure II.1 – Density, violin, and box plots for a normal (0, 1) distribution

Some of the plots provided with the simulation results for this Recommendation are an overlay of a box plot with a violin plot. An example is given in Figure II.2. In addition, a 1-D density plot of the actual points is also added to the margin.

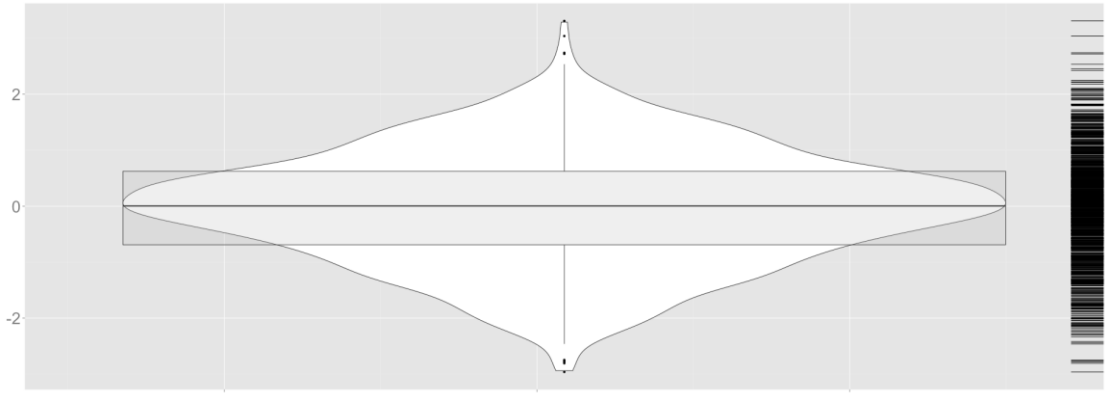


Figure II.2 – Combined violin-BFIox plot for a normal (0,1)

Appendix III

Simulation of high packet loss rates

(This appendix does not form an integral part of this Recommendation.)

Applications that use TCP are insulated from packet loss because the TCP protocol provides reliable delivery of data. TCP automatically detects when packets are lost and resends them. Therefore, at the application layer, the primary observable symptom of packet loss is increased latency. Applications such as video that depend upon continuous data transmission typically maintain a buffer in the receiver. The application reads from the buffer while the TCP layer writes received data into it. During steady state, the buffer remains at a constant level. When packet loss occurs, the application continues to read from the buffer while the transport layer (TCP) tries to recover from the loss and catch up with the application. If TCP recovery takes too long, for example because packet loss is severe, then the application buffer becomes empty and the video display freezes. It is safe to say that most Internet users have seen this phenomenon at one time or another. So when testing applications carried over TCP, packet loss is not as important as other network characteristics, such as latency.

Of course, packet loss is still important to TCP algorithms because it is a normal and vital part of TCP's congestion and rate control mechanisms. Packets are typically lost when a network queue becomes full. This happens during times of high network utilization and congestion. It is during these periods of time that maximum delays also occur. Over time, link capacity can therefore be thought of in terms of both bandwidth and delay, and losses will occur during times of maximum delay on a link. For applications carried over RTP/UDP, a packet can be considered a lost packet if it exceeds a specified delay threshold.

Therefore, high packet loss can be predicted using the empirical cumulative distribution function (CDF) of delay and the method described below.

In this update of ITU-T G.1050, the simulation produces time series tables of bandwidth and delay for the unmanaged "best effort" link of the test cases defined by the Recommendation. An example plot of this for the case Dp4 in the downward direction is shown in Figure III.1.

The cdf for each test case is provided in tabular form in six files called ./{D|G}/out.{down|up|bidi}/cdf.csv. These files are part of the electronic attachment in Annex X. Each of these files has one test case per line, and each line has the cdf value for each percentile level.

On the theory that excessively delayed packets are effectively lost packets, traffic periods that exceed a chosen delay bound can be considered dropped. As an example, consider the cdf values in Table III.1 and its plot in Figure III.2. For this test case, a 2% loss could be generated if packets exceeding 160.779 ms of delay were to be discarded.

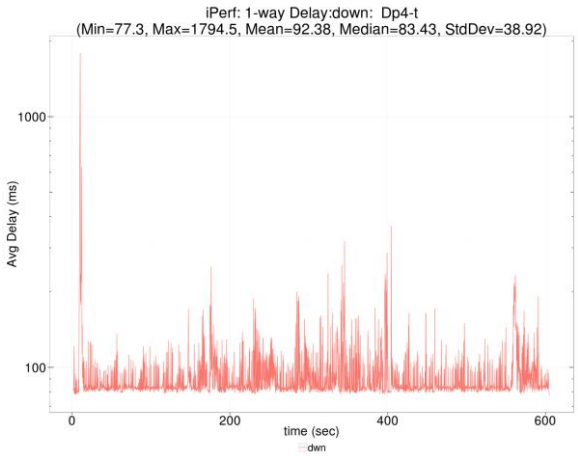


Figure III.1 – Downlink delay

Table III.1 – Example packet delay CDF data

Pct.	Dly (ms)	Pct.	Dly (ms)	Pct.	Dly (ms)	Pct.	Dly (ms)	Pct.	Dly (ms)
0%	77.300	20%	81.244	40%	82.660	60%	84.434	80%	98.222
1%	79.030	21%	81.320	41%	82.738	61%	84.622	81%	99.305
2%	79.310	22%	81.434	42%	82.810	62%	84.810	82%	100.179
3%	79.520	23%	81.530	43%	82.870	63%	84.970	83%	100.953
4%	79.690	24%	81.590	44%	82.940	64%	85.240	84%	102.135
5%	79.831	25%	81.675	45%	82.990	65%	85.579	85%	103.482
6%	79.961	26%	81.740	46%	83.070	66%	86.133	86%	104.699
7%	80.060	27%	81.800	47%	83.139	67%	86.627	87%	105.954
8%	80.150	28%	81.866	48%	83.230	68%	87.080	88%	107.338
9%	80.252	29%	81.930	49%	83.330	69%	87.821	89%	109.744
10%	80.340	30%	82.000	50%	83.430	70%	88.636	90%	111.494
11%	80.430	31%	82.080	51%	83.480	71%	89.670	91%	114.031
12%	80.530	32%	82.130	52%	83.560	72%	90.630	92%	116.294
13%	80.610	33%	82.210	53%	83.631	73%	91.669	93%	120.589
14%	80.710	34%	82.260	54%	83.700	74%	92.559	94%	124.939
15%	80.833	35%	82.330	55%	83.780	75%	94.160	95%	128.780
16%	80.900	36%	82.380	56%	83.900	76%	95.035	96%	135.242
17%	80.993	37%	82.450	57%	84.000	77%	95.905	97%	146.009
18%	81.094	38%	82.530	58%	84.100	78%	96.672	**98%**	**160.779**
19%	81.184	39%	82.600	59%	84.270	79%	97.440	99%	188.326
								100%	1794.500

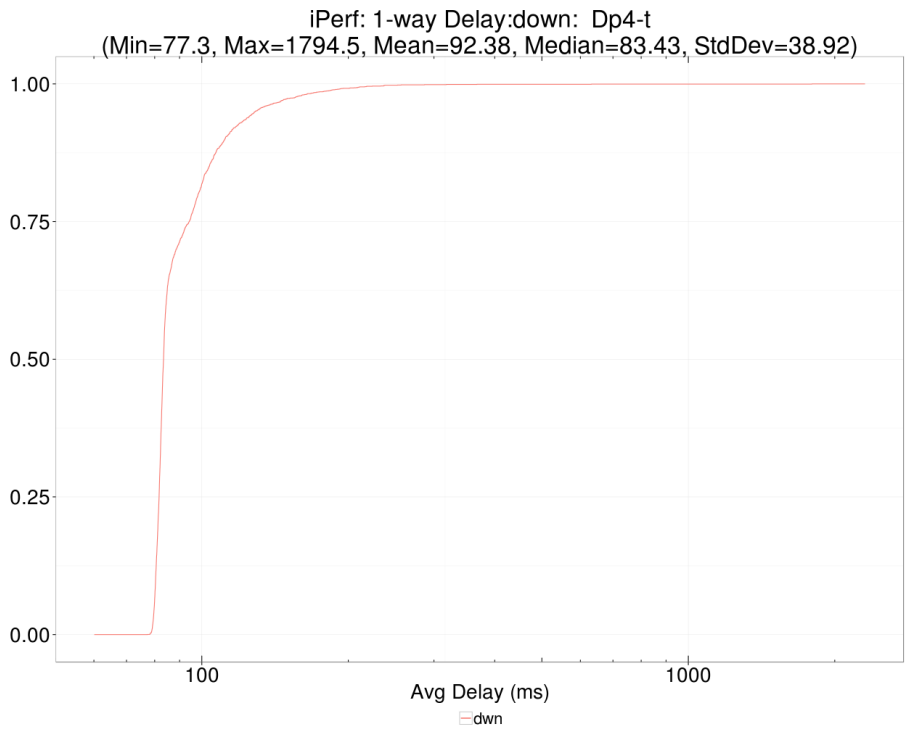


Figure III.2 – Delay CDF plot

Bibliography

[b-ANSI/TIA-912C] TIA-912-C (2016), *Telecommunications – IP Telephony Equipment – Voice Gateway Transmission Requirements*.

[b-ETSI TS 101 329-2] ETSI TS 101 329-2 (2002), *Telecommunications and Internet Protocol Harmonization Over Networks* (*TIPHON*) *Release 3, End-to-end Quality of Service in TIPHON systems; Part 2: Definition of speech Quality of Service (QoS) classes*.

[b-IEEE 802.11A] IEEE 802.11A-1999, *IEEE standard for Telecommunications and Information Exchange Between Systems – LAN/MAN specific Requirements*.

[b-IEEE 802.11B] IEEE 802.11B/COR 1-2001, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher Speed Physical Layer Extension in the 2.4 GHz Band*.

[b-IEEE 802.11G] IEEE 802.11G-2003, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further High Data Rate Extension in the 2.4 GHz Band*.

[b-TIA-810-A] TIA-810-A (2004), *Telecommunications – Telephone Terminal Equipment – Transmission Requirements for Narrowband Voice over IP and Voice over PCM Digital Wireline Telephones*.

[b-TIA-1001] TIA-1001 (2004), *Transport of TIA-825-A Signals over IP Networks*.

[b-TIA TSB116A] TIA TSB116-A (2006), *Telecommunications – IP Telephony Equipment – Voice Quality Recommendations for IP Telephony*.

[b-Cormen] Cormen, T., *et al*. (2001), *Introduction to Algorithms*, second edition, MIT Press and McGraw-Hill, ISBN 0-262-03293-7.

[b-iPerf] Tirumala, Ajay, *et al.*, *Iperf: The TCP/UDP bandwidth measurement tool*. http://iperf.sourceforge.net/ .

[b-Violin] Hintze, Jerry L., and Ray D. Nelson (1998), *Violin plots: a box plot-density trace synergism*. The American Statistician 52.2: 181-184.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	**Transmission systems and media, digital systems and networks**
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2016