broadband forum

TR-421

Application-Layer Test Traffic Architecture and Requirements

ISSUE: 1

ISSUE DATE: AUGUST 2019

Notice

The Broadband Forum is a non-profit corporation organized to create guidelines for broadband network system development and deployment. This Technical Report has been approved by members of the Forum. This Technical Report is subject to change. This Technical Report is copyrighted by the Broadband Forum, and all rights are reserved. Portions of this Technical Report may be copyrighted by Broadband Forum members.

Intellectual Property

Recipients of this Technical Report are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of this Technical Report, or use of any software code normatively referenced in this Technical Report, and to provide supporting documentation.

Terms of Use

1. License

Broadband Forum hereby grants you the right, without charge, on a perpetual, non-exclusive and worldwide basis, to utilize the Technical Report for the purpose of developing, making, having made, using, marketing, importing, offering to sell or license, and selling or licensing, and to otherwise distribute, products complying with the Technical Report, in all cases subject to the conditions set forth in this notice and any relevant patent and other intellectual property rights of third parties (which may include members of Broadband Forum). This license grant does not include the right to sublicense, modify or create derivative works based upon the Technical Report except to the extent this Technical Report includes text implementable in computer code, in which case your right under this License to create and modify derivative works is limited to modifying and creating derivative works of such code. For the avoidance of doubt, except as qualified by the preceding sentence, products implementing this Technical Report are not deemed to be derivative works of the Technical Report.

2. NO WARRANTIES

THIS Technical Report IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS Technical Report SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE BROADBAND FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS Technical Report.

3. THIRD PARTY RIGHTS

Without limiting the generality of Section 2 above, BROADBAND FORUM ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE Technical Report IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE Technical Report, BROADBAND FORUM TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

The text of this notice must be included in all copies of this Technical Report.

Issue History

Issue Number	Approval Date	Publication Date	Issue Editor	Changes
1	5 August 2019	5 August 2019	Ken Ko, ADTRAN	Original

Comments or questions about this Broadband Forum Technical Report should be directed to info@broadband-forum.org.

Editors	Ken Ko	ADTRAN
	Daniel Moss	UNH IOL
Access & Transport Architecture Work Area Director	Dave Sinicrope	Ericsson

Table of Contents

List of Figures

Executive Summary

The Quality of Experience (QoE) perceived by the customers of network services is dependent on both the design of the network, and on the behavior of the traffic sent over the network by the applications used by those customers. Different types of applications generate traffic having very different characteristics. For example, video streaming applications usually send data over http/https in "chunks" carrying several seconds of video content each. In contrast, video conferencing applications send much more frequent groups of packets at regular intervals, using different protocols from video streaming at multiple layers in the network stack. The requirements of each application dictate the protocols used and the traffic sent, and the QoE experienced by the user depends on the different network performance requirements that result from each type of application. Any degradation of the user's QoE can be exacerbated by the combinations of different applications and resulting traffic patterns generated by multiple users in a subscriber household, and multiple subscriber households in a network.

The traffic generated in existing test specifications does not typically capture the kinds of application-layer behavior discussed above. In order to accurately characterize the performance of networks and algorithms under realistic conditions going forward, it is desirable to generate test traffic that exhibits the above complexity, that conforms to defined model parameters, and that can be reliably repeated. In addition, appropriate metrics are desired to characterize performance that may at times be application-specific.

This Technical Report defines an architecture and related requirements for the specification of test traffic and measurements associated with the application layer. It includes a number of use cases that benefit from testing at the application layer, and describes an approach to modeling traffic generated by different classes of applications, as well as describing some of the more common classes of applications that currently enjoy widespread use. The Technical Report also describes issues associated with test methodologies based on application-layer traffic and how they can differ from methodologies based on lower layer flows.

1 Purpose and Scope

1.1 Purpose

Many network design choices and algorithms at multiple layers in the network stack can affect the Quality of Experience (QoE) perceived by the customers of network services. QoE is affected by more than network design, however. It is also highly dependent on the traffic placed on the network by those customers, both in terms of the amount of traffic and its behavior over different protocols, within different applications, and relative to other traffic with which it shares the network. Some examples of potential interrelationships between traffic and network performance include:

- Passive Optical Network (PON) systems often oversubscribe upstream bandwidth by using Dynamic Bandwidth Allocation (DBA) to allocate bandwidth in grants to Optical Network Terminals (ONTs) that need them. The time domain behavior of these grants can have a significant effect on Transmission Control Protocol (TCP) performance.

- G.fast systems rely on Time Domain Duplexing (TDD) to avoid Near-End Crosstalk (NEXT) in the received signal. Coordinated Dynamic Time Assignment (cDTA) can optimize system performance under these conditions, but the QoE resulting from different cDTA algorithms can depend on traffic's characteristics in the time domain.

- The performance and scalability of virtualized networks may be affected by the time domain behavior of the traffic crossing them. For example, the performance of a set of Virtual Network Functions (VNFs) virtualized on a given server may be different for traffic composed of large bursts compared to a steady stream of packets.

- The relative Quality of Service (QoS) performance experienced by the customers of different virtual Service Providers sharing a common access network may be affected by the application-level characteristics of the traffic carried by each provider.

Different types of applications generate traffic having very different characteristics. For example:

- Over-The-Top (OTT) video streaming applications, which generate most of the Internet access traffic carried over residential networks, usually send data over http/https in "chunks" carrying several seconds of video content each.

- Managed video services stream multicast video groups over User Datagram Protocol (UDP), typically with QoS priority.

- OTT Voice over IP applications send small packets at regular intervals of around 10 to 20 msec.

- Managed voice services send small packets at regular intervals, typically with QoS priority.

- Video over IP applications send large packets (or groups of packets) at regular intervals.

- A web page retrieval typically generates dozens of flows including DNS requests and relatively small TCP flows that may be requested from different network addresses.

- Speedtest applications typically generate many TCP flows, and attempt to reach maximum throughput in one direction for a period of time followed by maximum throughput in the other direction for a period of time.

The complexity of application level traffic is driven not only by the different behaviors exemplified by the above list, but also because of the combinations of different applications and resulting traffic patterns generated by subscribers. The mix of applications can be different for different types of subscribers. Additionally, the mix can change over time and with different average loads as new applications enter the market and gain popularity.

The test traffic generated in existing test specifications does not capture the kinds of application-layer behavior discussed above. Most performance test plans, if they specify test traffic at all, specify it as Ethernet frames, or at best as one or a small number of UDP or TCP flows or as a file transfer over FTP. Some test specifications (e.g., TR-254 [3]) use specific control protocols such as Dynamic Host Configuration Protocol (DHCP) or Internet Group Management Protocol (IGMP), but they still do not specify user traffic generation at the application level. In order to accurately characterize the performance of networks and algorithms under realistic conditions going forward, we must be able to generate test traffic that exhibits the above complexity. The generated traffic must also conform to defined model parameters, and it must be repeatable.

This Technical Report defines an architecture and requirements for the specification of test traffic generated at the application layer. It supports specification of test traffic that exhibits the complexity resulting from multiple types of applications and subscribers aggregated in a common network and competing for resources. The Technical Report also enables test traffic to be specified using consistent, unambiguous parameters that support repeatable test results under complex scenarios across different test labs, service providers, and vendors.

The introduction of network virtualization provides an opportunity to create complex test traffic generation systems using the same common building blocks that form an NFV infrastructure. Application servers, access networks, and even subscribers can be virtualized within a testbed to create realistic loading scenarios that emulate the behavior of many subscribers interacting with the network under controlled and repeatable conditions.

The introduction of application-level test traffic generation into test cases for Open Broadband Labs and for other projects will result in more realistic, thorough test scenarios and will achieve the following benefits:

- Ability to test performance of network elements and systems affected by complex traffic characteristics
- Consistency in specification of complex traffic (eliminate ambiguity)
- Repeatability of test conditions with complex traffic (e.g., between labs and vendors)
- Faster Time to Revenue
- Scalability

- Savings in Non-Recurring Costs

1.2 Scope

This Technical Report defines an architecture and requirements for specifying the generation of test traffic at the application layer (e.g., Hypertext Transfer Protocol [HTTP], Dynamic Adaptive Streaming over HTTP [DASH], etc.), as well as associated topics such as test methodologies, metrics, and analysis of results. The resulting test traffic emulates the realistic time domain behavior exhibited by traffic generated from multiple applications and multiple subscribers, as seen in field deployments. The architecture contains models that define classes of applications based on traffic behavior, and it facilitates incorporation of new application classes as new behaviors emerge. Similarly, the architecture includes models that group application models along with usage parameters to model different types of subscribers. Initially, the models primarily address High Speed Internet Access (HSIA) for residential and home office use; in a later phase, business applications could be addressed.

The scope for this Technical Report includes:

- Overview of application-layer test traffic
  - How it differs from and complements test traffic generated as L2-L4 flows
  - How test methodologies using traffic generated at L7 differ from lower layer test methodologies
- Use cases
- Architecture description
  - Architecture elements and relationships
  - External interfaces
  - Classes of application and related parameters
  - Application sessions
  - Test methodologies and analysis of results
  - Randomization
  - Metrics
  - Network virtualization
- Requirements
  - Information models
  - Application sessions
  - Test control and methodologies

2 References and Terminology

2.1 Conventions

In this Technical Report, several words are used to signify the requirements of the specification. These words are always capitalized. More information can be found be in RFC 2119 [1].

MUST	This word, or the term "REQUIRED", means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the term "RECOMMENDED", means that there could exist valid reasons in particular circumstances to ignore this item, but the full implications need to be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED" means that there could exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications need to be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the term "OPTIONAL", means that this item is one of an allowed set of alternatives. An implementation that does not include this option MUST be prepared to inter-operate with another implementation that does include the option.

2.2 References

The following references are of relevance to this Technical Report. At the time of publication, the editions indicated were valid. All references are subject to revision; users of this Technical Report are therefore encouraged to investigate the possibility of applying the most recent edition of the references listed below.
A list of currently valid Broadband Forum Technical Reports is published at www.broadband-forum.org.

Document		Title	Source	Year
[1]	RFC 2119	*Key words for use in RFCs to Indicate Requirement Levels*	IETF	1997
[2]	TR-143 Amendment 1 Corrigendum 1	*Enabling Network Throughput Performance Tests and Statistical Monitoring*	BBF	2015

[3]	TR-254	*Functionality Tests for Ethernet Based Access Nodes*	BBF	2012
[4]	TR-304	*Broadband Access Service Attributes and Performance Metrics*	BBF	2015
[5]	TR-384	*Cloud Central Office Reference Architectural Framework*	BBF	2018
[6]	TR-390	*Performance Measurement from IP Edge to Customer Equipment using TWAMP Light*	BBF	2017
[7]	RFC 7594	*A Framework for Large-Scale Measurement of Broadband Performance (LMAP)*	IETF	2015
[8]	RFC 8193	*Information Model for Large-Scale Measurement Platforms (LMAPs)*	IETF	2017
[9]	RFC 8194	*A YANG Data Model for LMAP Measurement Agents*	IETF	2017
[10]	RFC 2681	*A Round-trip Delay Metric for IPPM*	IETF	1999
[11]	RFC 6673	*Round-Trip Packet Loss Metrics*	IETF	2012
[12]	RFC 7679	*A One-Way Delay Metric for IP Performance Metrics (IPPM)*	IETF	2016
[13]	RFC 7680	*A One-Way Loss Metric for IP Performance Metrics (IPPM)*	IETF	2016
[14]	RFC 2544	*Benchmarking Methodology for Network Interconnect Devices*	IETF	1999
[15]	MEF 10.3	*Ethernet Services Attributes Phase 3*	MEF	2013
[16]	H.323	*Packet-based multimedia communications systems*	ITU-T	2009

2.3 Definitions

The following terminology is used throughout this Technical Report.

Application class	A category of applications that can be described as having common behavior with regard to the traffic they send across the network.
Application client	An instance of a client owned by a specific virtual subscriber and having the attribute values associated with a specific application profile.
Application model	A data model describing an application class. An application model identifies the attributes describing the application class but does not assign values to them.
Application profile	A data model with values defining an application class for a specific test case. An application profile assigns values to the attributes in the application model.

Application server	An instance of a server having the attribute values associated with a specific application profile.
Application session	A set of events, actions, and information exchange associated with an application client, having parameters defined by the associated application profile.
Deterministic scheduling	The scheduling of application sessions at defined (non-random) times within a test trial.
Monte Carlo testing	A test methodology in which trials are repeated many times to create a statistically significant set of results.
Subscriber	An entity associated with a single instance of a network service. A subscriber can represent a person, a device, or a group of persons and/or devices, for example the people and devices within a household and sharing a broadband access service.
Subscriber model	A data model describing the attributes associated with a subscriber.
Subscriber profile	A data model with values defining the behavior of a subscriber for a specific test case. A subscriber profile includes one or more application profiles plus attribute values defining how each application profile is used.
Trigger file	A file defining the planned start times and session parameters for all application sessions in a specific instantiation of a test case.
Virtual Subscriber	An object modeling the behavior of a subscriber having the attribute values associated with a specific subscriber profile.

2.4 Abbreviations

This Technical Report uses the following abbreviations:

ALT	Application-Layer Testing
API	Application Programming Interface
AR	Augmented Reality
BBF	Broadband Forum
cDTA	Coordinated Dynamic Time Assignment
CPE	Customer Premises Equipment
DASH	Dynamic Adaptive Streaming over HTTP
DBA	Dynamic Bandwidth Allocation
DHCP	Dynamic Host Configuration Protocol
DPU	Distribution Point Unit
DTA	Dynamic Time Assignment
HDS	HTTP Dynamic Streaming

HLS	HTTP Live Streaming
HSIA	High Speed Internet Access
HTTP	Hypertext Transfer Protocol
IGMP	Internet Group Management Protocol
MMORPG	Massively Multiplayer Online Role-Playing Games
MS-SSTR	Microsoft Smooth Streaming Protocol
NEXT	Near-End Crosstalk
NFV	Network Function Virtualization
ONT	Optical Network Terminal
OTT	Over The Top
P2P	Peer-to-Peer
PON	Passive Optical Network
PTZ	Pan, Tilt, and Zoom
ONVIF	Open Network Video Interface Forum
QoE	Quality of Experience
QoS	Quality of Service
RTCP	RTP Control Protocol
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SRTP	Secure Real Time Protocol
SUT	System Under Test
TCP	Transmission Control Protocol
TDD	Time Domain Duplexing
TR	Technical Report
UDP	User Datagram Protocol
UNI	User-Network Interface
VNF	Virtual Network Function
VoIP	Voice over Internet Protocol
VR	Virtual Reality
XR	Extended Reality

3 Technical Report Impact

3.1 Energy Efficiency

The test traffic defined by this architecture is intended to be used in a test network environment, and not in deployed networks. As such, this Technical Report has no impact on energy efficiency.

3.2 Security

The test traffic defined by this architecture is intended to be used in a test network environment, and not in deployed networks. As such, this Technical Report has no impact on security.

3.3 Privacy

The test traffic defined by this architecture is intended to be used in a test network environment, and not in deployed networks. As such, this Technical Report has no impact on privacy.

4 Introduction to Application-Layer Testing

Broadband networks serve large numbers of subscribers with multiple services, ranging from best-effort Internet access, to managed services such as voice and video, to value-added services with dynamically configurable performance. The networks providing these services are typically oversubscribed – that is, the aggregate capacity of the network is lower than the sum of all service rates offered to the set of subscribers served by the network. Networks can be oversubscribed and still provide the necessary performance for all offered services because subscribers rarely use the full capacity of those services. In fact, when aggregated over a large set of subscribers even during peak usage hours, traffic volume is rarely more than a small percentage of the sum of the service rates.

The subscriber traffic that does traverse the network, however, is a complex combination of flows from different subscribers and applications. For example, fixed broadband Internet access services typically serve households that comprise multiple people, each of whom may in turn use multiple devices connected to the service. As a result, a single "subscriber" can be responsible for dozens of flows generated by multiple applications and devices at a given instant in time.

Even the traffic flows generated by a single application can exhibit complex behavior. Individual flows may be rate adaptive, with adaptation occurring at one or more layers in the network stack. A flow can exhibit burstiness over different time epochs, or it can send packets in a regular, periodic stream. The set of flows associated with a single application can have many different remote endpoints. Different types of applications generate traffic flows with distinctly different behaviors, each type according to the respective requirements of the application. For example, Figure 1 shows flows from two different video applications, one streaming video from a network server to a subscriber and the other sending video conferencing traffic. Although each application is generating video content and the average throughput for each is between 1.6 and 1.8 Mbps, the way in which traffic is generated by the two applications is markedly different.

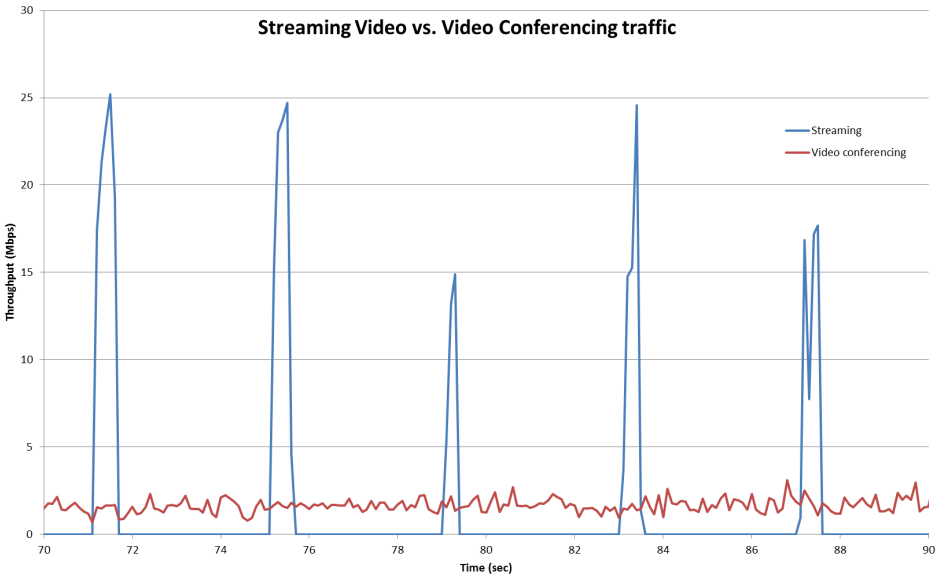

Figure 1 – Streaming video vs. video conferencing traffic

The aggregation of traffic from many subscribers within the access network provides another source of complexity. Subscribers differ greatly in the amount of traffic they generate, with a minority of subscribers accounting for the majority of the traffic. Subscribers also differ in the applications they use – while most generate traffic primarily in the downstream direction, some generate traffic with an upstream bias (e.g., security video) or in both directions (e.g., cloud-based applications and peer-to-peer traffic).

When traffic exhibiting the application-level complexity described above is aggregated in an oversubscribed access network, flows can interact with each other in indeterminate ways. This Technical Report specifies an architecture and requirements for test platforms intended to reproduce this level of complexity in a lab environment, facilitating measurement and analysis of network performance in ways that are not feasible using simpler traffic flows. It also describes a set of models for application and subscriber behavior, and requirements for test platforms, enabling unambiguous specification of traffic generation and facilitating portable and repeatable testing. Finally, it discusses issues associated with test methodologies, metrics, and measurements focusing on the application layer.

Testing using application-layer traffic is expected to complement testing with lower-layer traffic, not to replace it. While application-layer testing can measure performance in ways that may not be addressable with lower-layer testing, it typically requires more time for both testing and analysis of results than does lower-layer testing. Representative test methodologies and analysis techniques that can be applied to application-layer testing are discussed in this Technical Report.

4.1 Related Work

Application-layer testing is a new area of work in the Broadband Forum (BBF), but it builds on components created in the BBF and in other organizations. Within the BBF, TR-143 [2] specifies data model objects that support performance throughput tests, and TR-304 [4] specifies a framework for measuring performance between various points within a network. TR-390 [6] specifies requirements supporting performance measurement from the IP edge to the customer premises using TWAMP Light.

The IETF has published several RFCs related to Large-Scale Measurement of Broadband Performance, including a framework [7] and information and data models [8][9] consistent with TR-304. The IETF's IP Performance Management (IPPM) Working Group has also published a large number of RFCs defining metrics for network performance [10][11][12][13]. The MEF has a number of similar metrics defined for Ethernet frames [15]. Many of the metrics referenced above are applicable to test scenarios using application-layer test traffic.

5 Use Cases

5.1 Performance of Virtualized Network Functions

CloudCO [5] and other virtualization-based network architectures implement some control- and user-plane functions as virtualized functions running on general purpose compute resources within the Network Function Virtualization (NFV) infrastructure. These architectures also facilitate the offering of value-added services and functions through the instantiation of VNFs that operate on some of the traffic in the user plane. Resources for the virtualized functions are orchestrated to support the expected loads associated with the functions. Orchestration of resources must strike a balance between performance and efficiency: computing cycles are wasted if too many resources are allocated to a given function, but the function may cause excessive packet losses or exhibit other failure mechanisms during instances of peak usage if too few resources are allocated. The performance of a given VNF may also depend on the orchestration and loading of other functions on the same compute resource.

Since the resources required by a given function are dependent on the time domain behavior of the associated traffic, testing with application-layer traffic can be a valuable methodology for virtualized functions. This use case also highlights the need to differentiate traffic from different emulated users and/or devices within a single subscriber household, because some virtualized functions are user- or device-specific. Examples include virtual residential gateways and parental controls.

5.1.1 Actors

The primary actors for this use case are residential subscribers. Since the value-added features under test in some cases may be user- or device-specific, it may be valuable to nest subscriber profiles or to find another means to differentiate between devices or users within a subscriber profile that would normally represent a household. Multiple subscriber profiles may be used to reflect different types of subscriber behaviors on the network. All subscriber profiles should include a variety of application profiles. The subscriber profiles may scale with regard to overall average loading for different test sets to reflect traffic growth over time.

The System Under Test (SUT) would reflect a complete or partial CloudCO implementation.

5.1.2 Test Metrics

The standard metrics used to assess QoS are applicable to this use case. These include:

- Packet loss
- Delay
- Delay variation

In addition, application-level metrics associated with the application classes populating the subscriber profiles may apply. Performance tests or other specific application classes may be applied deterministically to measure specific metrics.

5.1.3 Test Methodology

Depending on the features being tested, the test network may be pre-provisioned with the desired service configurations. Alternatively, services may be provisioned on demand during the test.

The distribution of subscriber profiles for this use case can be designed to suit the specific test purposes. For example, virtual subscribers may be instantiated with profiles reflecting light, medium, and heavy usage in proportions as may be encountered in a typical deployment. As noted above, subscriber profiles may be nested to reflect application usage by individual users or devices.

In this use case, the traffic associated with the function under test (the "test traffic") would be a subset of the traffic generated, and would be differentiable by the test platform, perhaps via association with specific application profiles. The test itself would consist of virtual subscribers randomly instantiating application sessions per their profile parameters while the metrics listed above are measured for the test traffic. Each trial in this test would be defined by the set of subscriber profiles used, and by the associated average loading conditions of the test traffic. Each trial would run for a time sufficient to generate a statistically significant set of measurements, accounting for the variations in loading due to session instantiation. Loading conditions for multiple trials could be scaled in different ways, for example:

- By scaling the numbers of virtual subscribers served, representing growth in the take rate
- By scaling the average loading for each virtual subscriber, representing traffic growth over time
- By changing the ratio of heavy to light users, representing variation in subscriber populations

Post-test processing would include aggregation and statistical analysis of the measurements.

5.2 Multi-tenant QoS in CloudCO

One of the features of CloudCO is a QoS framework that supports wholesale and retail service providers with a set of connectivity constructs and associated User-Network Interfaces (UNIs). A UNI can be defined across the switch fabric, which raises the possibility of shared UNIs in which multiple service end points feed a single wholesaler service access point. This structure creates challenges with regard to coordinating traffic shaping instances in such a way as to minimize packet loss while also maximizing use of the service. Additional challenges may be associated with sharing capacity between retail service providers at potential congestion points such as oversubscribed PONs.

Testing of multi-tenant networks such as those described above is best performed using traffic that exhibits the complexity in the time domain associated with multiple types of applications and subscribers. One of the defining features of these networks is oversubscription of services, potentially at both the retail and wholesale levels, making "worst-case" flows that fill pipes to capacity unusable as test traffic. Less-than-worst case TCP and UDP flows generated at the transport layer may not sufficiently reflect the burstiness associated with application-layer behavior, causing misleading test results.

5.2.1 Actors

This use case involves three tiers of actors:

- The wholesale service provider who may be the network operator. This actor provides wholesale services to retail providers and usually implements functions at appropriate UNIs to police the traffic associated with the contracted wholesale services.

- The retail service provider. This actor leases wholesale services from the wholesale provider and offers retail services to the subscriber. The retail provider usually implements functions to shape traffic at appropriate wholesale UNIs to prevent packet loss, and may also implement policing at retail UNIs.

- The retail subscriber. This actor uses the retail service through a set of applications.

Multiple retail subscriber profiles may be used to reflect different types of subscriber behaviors on the network. All subscriber profiles should include a variety of application profiles. The subscriber profiles may scale with regard to overall average loading for different test sets to reflect traffic growth over time.

The SUT would reflect a complete or partial CloudCO implementation supporting multiple retail service providers.

5.2.2 Test Metrics

The standard metrics used to assess QoS are applicable to this use case. These include:
- Packet loss
- Delay
- Delay variation

Depending on the SUT and on how loads are applied, the resulting utilization or aggregate traffic statistics at specific links may also apply as metrics.

In addition, application-level metrics associated with the application classes populating the subscriber profiles may apply. Performance tests or other specific application classes may be applied deterministically to measure specific metrics.

5.2.3 Test Methodology

Depending on the features being tested, the test network may be pre-provisioned with the desired wholesale and retail service configurations, including any shaping, policing, and other QoS mechanisms in the design. Alternatively, dynamic services may be provisioned on demand.

The distribution of subscriber profiles for this use case can be designed to suit the specific test purposes. For example, virtual subscribers may be instantiated with profiles reflecting light,

medium, and heavy usage in proportions as may be encountered in a typical deployment. Virtual subscribers are instantiated for each retail service provider represented in the test.

One test type would consist of all virtual subscribers randomly instantiating application sessions per their profile parameters while the metrics listed above are measured. Each trial in this test would be defined by the set of subscriber profiles used and the associated average loading conditions. Each trial would run for a time sufficient to generate a statistically significant set of measurements, accounting for the variations in loading due to session instantiation. Overall loading conditions for multiple trials could be scaled in different ways, for example:

- By scaling the numbers of virtual subscribers served by one or more retail providers representing growth in the provider's take rate
- By scaling the average loading for each virtual subscriber, representing traffic growth over time
- By changing the ratio of heavy to light users, representing variation in subscriber populations

A different test could add application sessions scheduled at pre-defined times for the purpose of measuring specific metrics. This test would typically be run as a set of Monte Carlo trials, with a large number of test runs having randomized initial states. Overall loading for multiple Monte Carlo test sets could be scaled in the same ways listed above.

Post-test processing would include aggregation and statistical analysis of the measurements.

5.3 Adaptive Rate Streaming Video Testing

This use case demonstrates how application-layer testing can enable metrics that directly measure application-based QoE. Adaptive rate streaming video applications provide video at any of a number of pre-encoded rates to subscribers. The video is normally delivered at the highest rate supported by the network. This rate may change as the network path experiences momentary congestion, as network faults occur or paths change, or as virtualized resources are re-assigned. Since the same video encoded at different rates can be expected to have different levels of visual artifacts, the delivered streaming rate is an application-layer metric for streaming video QoE. Other application-layer metrics include the number and duration of "stalls" (video frame freeze due to buffer underrun), and the time required to fill the receive buffer to the point required to initiate playout.

Adaptive rate streaming video is well suited to application-layer testing for multiple reasons. First, it represents the largest share of all application classes as a percentage of total consumer Internet traffic volume. Second, the adaptive rate behavior for each stream is independently controlled by the application client, so video streams interact with each other and with other traffic on the network in unpredictable ways. Finally, test clients can perform measurements based on the metrics described above at the application layer, facilitating more direct evaluation of subscriber QoE than would be possible using traffic and metrics based on lower network layers. Other application classes can similarly benefit from definition of appropriate application-layer metrics specific to each class.

5.3.1 Actors

This use case focuses on residential subscribers. The subscriber profiles should have a high percentage of streaming video traffic, both because that is the application being measured and because it is reflective of actual statistics. The video may be delivered Over The Top (OTT) of Internet access services, within video services managed by a service provider, or via both mechanisms. The profiles should also include other application classes in proportion to their use in the field.

The use case can be designed to test different network subsegments at different levels of scaling. For example, in a small scale use case the SUT could be a single PON serving 64 subscribers. In a large scale use case, the SUT could be a virtualized central office serving thousands of subscribers.

5.3.2 Test Metrics

A non-exhaustive list of the application-layer metrics associated with streaming video includes the following:

- Encoded streaming rates: what percentage of time is content streamed at each predefined rate?
- Stalls: how many video stalls (buffer under-runs) occur and of what duration? What is the resulting percentage of time that video is stalled?
- Time-to-start: how long does it take to deliver enough encoded content to begin playout at the receiving client?
- Streaming video performance vs. overall traffic: how does video performance (measured per the above metrics) change with changes in subscriber or network loading?

If streaming video is delivered as part of a managed service distinct from Internet access, the following secondary metric may apply:

- Streaming video performance vs. number of streams: how does video performance (measured per the above metrics) change as the number of streams provided to a given subscriber over the managed video service changes?

5.3.3 Test Methodology

The first step is to instantiate the desired number of virtual subscribers with profiles including the streaming video application class. Each virtual subscriber has within its profile one or more client applications that emulate the video streaming class and that are capable of recording the desired measurements. One or more emulated video servers are instantiated in the network cloud.

Upon test initiation, the virtual subscribers start generating application sessions for video and for their other application profiles. After an initial stabilization period in which the network transitions from no traffic to a steady state condition, each virtual subscriber's streaming video client applications can begin to record measurements for received video rate, stalls, and time-to-start.

The test should run as long as necessary to generate statistically significant sets of measurements. The required test duration may depend on the number of virtual subscribers served by the test network and the number of video clients associated with each virtual subscriber.

The test may add varying load conditions, either by introducing disruptive events (e.g., large file downloads and/or uploads) or by running multiple tests with different subscriber profiles. If disruptive events are added, the test results should include time stamps allowing correlation of measurements with events in time. If the streaming video is delivered as part of a managed video service, an additional test may generate video streams over the service deterministically to measure video performance vs. the number of streams concurrently provided to a virtual subscriber.

Post-test processing would include aggregation of the measurements from all video clients and analysis of the resulting statistics and distributions.

5.4 Long-term Stability of the Home Gateway in the Presence of Multiple Applications

Even with the move toward virtualization of the access network, the home gateway must support a rich feature set, resulting in complex software. This complexity increases for devices that are application aware and attempt to optimize for certain applications (gaming, video streaming, etc.). Services in the core network - routing, switching, DHCP, DNS, Voice over Internet Protocol (VoIP), and more, are typically taken care of by several different pieces of dedicated hardware or advanced cloud platforms in the datacenter. At the remote gateway, however, all of these services and features are handled by a single piece of hardware that is cost-sensitive with limited CPU and memory resources. This illustrates an issue with the scalability of individual applications as well as their throughput.

Moreover, the realistic mix of applications and services by users can expose bugs and coding artifacts that result in memory leaks or memory fragmentation. This involves more than just application throughput – it includes device behavior while repeatedly exercising routine services such as DHCP, as gateways do more than just route IP traffic. Over long periods of time, this can cause stability failures in even the most robust CPE.

Take, for example, this graph of a laboratory test of a CPE in the presence of line-rate throughput. In between each data point, additional exercises such as hosts renewing DHCP, routing user's DNS and HTTP traffic, etc., were performed. Over time, performance of the device decreased almost 80%, requiring a restart of the device:

Figure 2 – Example gateway throughput showing degradation in the presence of regular use over time

In addition, lab-level performance testing of CPE often focuses on raw line-rate testing using metrics such as those described in RFC 2544 [14]. While this may stress test the network processing engine, it does little to demonstrate the application-specific performance (DNS latency, for example) that will be important to the end-user.

5.4.1 Actors

This use case outlines two major, though different, cases for traffic pattern simulation through the home gateway:

- The performance of applications between server and end-user through a gateway
- The stability of the gateway after handling application traffic over time and in the presence of other protocols

As such, the actors in this use case includes the application servers, virtual subscribers and their clients (browsers, etc.), and other hosts.

5.4.2 Test Metrics

The test metrics could include (not an exhaustive list):

- Aggregate throughput in the presence of a traffic mix
- Aggregate throughput over time
- Per-application throughput and throughput over time

- Per-application latency and jitter

5.4.3 Test Methodology

Perform Monte Carlo testing at each of a number of per-subscriber average traffic patterns. Note that for this use case, it is important that the units under test not be reset between trials. In each Monte Carlo trial with N virtual subscribers:

1. N-1 virtual subscribers generate application sessions statistically per their predefined profiles.
2. The Nth virtual subscriber performs throughput testing in each direction.
3. Record the results of the throughput testing plus the aggregate throughput for all virtual subscribers during the trial.
4. Repeat the test, varying the subscriber profiles of the virtual subscribers to simulate different end-user activity.
5. Record any changes in throughput over time and after which profile changes.

5.5 Coordinated Dynamic Time Assignment (cDTA) in G.fast

A number of subscribers are served from a common Distribution Point Unit by G.fast. G.fast uses Time Division Duplexing to separate upstream and downstream traffic, because NEXT is too high to cancel at the frequencies used. For the same reason, all of the subscribers served by the DPU must use the same TDD ratio in any given symbol interval. Current generation G.fast specifies a configurable but fixed TDD ratio, which limits the maximum upstream and downstream rates that can be supported by the services offered from the DPU. However, by using coordinated DTA to assign the best TDD ratio for the next G.fast frame based on the momentary offered load on each subscriber loop in each direction, it would be possible to offer peak service rates that exceed those available using a fixed TDD ratio, possibly including a mix of asymmetric and symmetric services from the same DPU.

Figure 3 shows an example G.fast frame with the downstream and upstream momentary loads on three loops represented by blue and purple bars, respectively. The common TDD ratio is represented by the division between light blue and purple symbol periods. The extension of the blue bar on Line 3, and the purple bar on Line 2, represent offered load that cannot be carried in the current frame with the selected TDD ratio.

Figure 3 – Momentary offered loads and TDD ratio example

In this scenario, we seek answers to the following types of questions:

- What are the tradeoffs between varying the TDD ratio to maximize the aggregate throughput in both directions, vs. varying it to maximize the achievable peak rate in both directions?

- What are the effects of different algorithms on subscribers' achievable throughput in both directions?

- What are the effects of different algorithms on latency in both directions?

- How do the answers to the above questions change as average subscriber loads are increased?

5.5.1 Actors

This Use Case requires a mix of subscribers and applications that stress both directions of traffic. In typical residential services most traffic is downstream, and most subscribers should have profiles consistent with that fact. But there should also be a minority of subscribers using upstream or symmetric applications such as file transfer or peer-to-peer applications.

5.5.2 Test Metrics

The test metrics could include (not an exhaustive list):

- Throughput testing in both directions performed by a 'test' virtual subscriber for whom throughput test sessions are scheduled deterministically (see Section 6.2)
- Aggregate throughput for all virtual subscribers
- Latency and jitter

5.5.3 Test Methodology

Perform Monte Carlo testing at each of a number of per-subscriber average traffic loads. In each Monte Carlo trial with N virtual subscribers:

1. N-1 virtual subscribers generate application sessions statistically per their predefined profiles.
2. The Nth virtual subscriber performs throughput testing in each direction.
3. Record the results of the throughput testing plus the aggregate throughput for all virtual subscribers during the trial.

Post-test processing would include a statistical analysis of the performance testing and aggregate throughput results that may include CDFs of the results at each average load value tested.

6 Application-Layer Test (ALT) Models and Relationships

ALT provides traffic with application-specific behavior having defined loading characteristics to systems under test in a laboratory environment, enabling unambiguous, portable, and repeatable specification of test traffic generation. It does so using two types of profiles – a first type that specifies application behavior, and a second that specifies subscriber behavior. By combining profiles for multiple application classes in a subscriber profile, we specify a set of behaviors that defines the average downstream and upstream loads for a virtual subscriber as well as the statistical time domain behaviors of the traffic generated by that type of subscriber.

Multiple virtual subscribers are instantiated using one or more subscriber profiles in a network test profile. The combination of virtual subscribers in a given test profile, and the combination of application profiles and subscriber profiles used to instantiate the virtual subscribers, defines the generated traffic. The test process and the metrics and measurements applied are also defined in the test profile. The relationships between application profiles, subscriber profiles, virtual subscribers, processes, metrics, and network test profiles are shown in Figure 4.

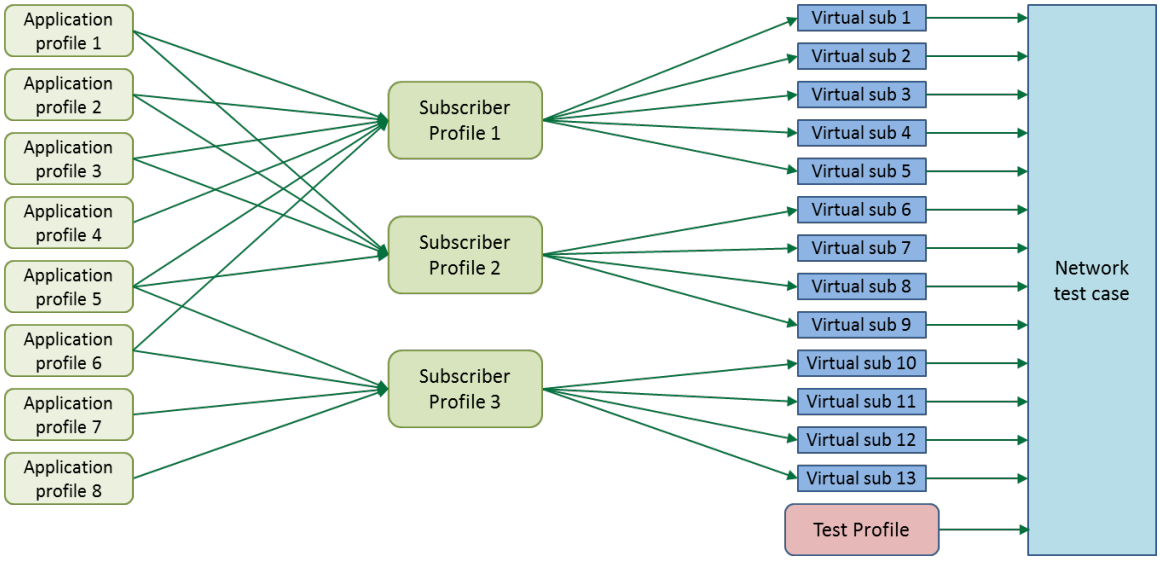

Figure 4 – Application profiles, subscriber profiles, virtual subscribers, and network test profiles

Application and subscriber profiles are built from data models that specify the sets of parameters used. The application and subscriber data models are described in the sections below.

6.1 Application Profiles

In the context of ALT, an application is a generic function that generates traffic emulating the behavior of software designed for a specific purpose. Applications are generic in that they don't model a specific implementation. Instead, they model a set of behaviors that may be common to many implementations. The term "application class" is used to describe this set of behaviors. Example application classes include streaming video, voice over IP, and gaming.

The primary purpose of an ALT application is to generate traffic that emulates the behavior of applications of its class. There is usually no requirement for an ALT application to either generate or process meaningful payloads. For example, a streaming video client does not need to decode or display video, nor does a streaming video server need to send actual encoded video. These capabilities are not precluded, and fully functional applications may be used at times, for example to provide a visual demonstration of received video quality. However, the primary purpose of an ALT streaming video server is to generate chunks of traffic with the expected size in response to requests from an ALT streaming video client, and the primary purpose of the client is to request chunks of the appropriate size and with the appropriate timing given the stream's arrival history. The chunks themselves may contain randomly generated payloads, and the client may discard them upon reception. In the same spirit, payload-related functions such as encryption are generally not supported unless they are required for a specific test purpose.

An application data model specifies a set of parameters used to create an application profile. The general characteristics used to define application classes are described below and in Appendix A .

The characteristics common to all application classes are shown below. In addition to these common characteristics, an application class may incorporate characteristics specific to that class.

Direction	The primary direction of traffic, not including requests, reverse direction protocol traffic such as Acks, etc. For example: • Streaming video is primarily downstream • File backup is primarily upstream • VoIP sends traffic in both directions concurrently • File transfer can occur in either direction
Session size definition	The type of unit (time or volume) used to define the size of a session. For example, a streaming video's session size is defined using time, because a given program has a fixed duration in time regardless of the encoding resolution (hence the number of bytes) used to deliver it. Conversely, file transfer is defined using volume, since the file size is a constant regardless of how long it takes to transfer.
Session size distribution	The statistical distribution used to instantiate the sizes of individual application sessions. Common distributions include uniform, exponential, and Gamma.
Protocols	The primary protocols used by the application. Examples include DASH, HTTP, TCP/UDP, etc.
Rate adaptation	Whether or not the flows generated by an application are rate adaptive, and at what layers. Traffic sent over TCP rate adapts at the transport layer. Some applications such as video conferencing send traffic over UDP and rate adapt at the application layer. Other applications, such as streaming video, send traffic over TCP which rate adapts within each chunk, but control the spacing and size of chunks (hence the long-term average rate) at the application layer.
Packet size	The typical size of the packets transmitted.

An application profile is built from an application data model by adding values that specify the behavior of the application. For example, a data model for the streaming video application class may have fields for session size distribution, chunk size, offered data rates, application-layer protocol used, and other parameters. By populating these fields with specific values, one can build an application that emulates video centered around 30- or 60-minute episodes at a range of HD streaming rates over DASH, with 2-second chunks. A second application profile built from the same data model could emulate 4K video and movie-length sessions.

Descriptions for a number of common application classes are found in Appendix A .

6.2 Subscriber Profiles

A subscriber profile is built from multiple application profiles plus additional parameters that determine, for example, how frequently application sessions are instantiated for virtual subscribers using the profile. The frequency of session instantiation, combined with the average session size, determines the average load contributed by an application class to the subscriber profile. Each application profile defines its average session size for each direction of transmission via its session size distribution parameters. The subscriber profile associates parameters with each application profile that specify how often sessions for that application are initiated, in effect specifying how heavily a subscriber defined by this profile uses the application. By combining application profiles of different types plus the associated loading parameters, a test designer can build a subscriber profile with specified average loads in both the downstream and upstream directions.

The sessions for each application profile in a subscriber profile can be scheduled either statistically or deterministically. With statistical scheduling, application sessions are randomly initiated using the parameters associated with the application profile. Deterministic scheduling initiates sessions at defined times. For example, a deterministically scheduled throughput test may be set to begin 10 seconds after the start time of each trial in a Monte Carlo test. Statistically and deterministically scheduled applications can be mixed within a subscriber profile.

A subscriber profile is constructed as shown in Figure 5.

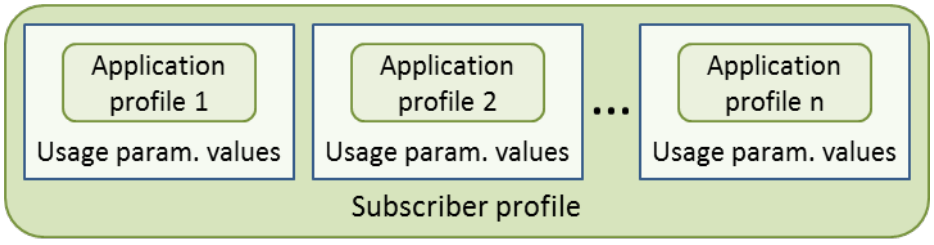

Figure 5 – Subscriber profiles

Subscriber profiles can be nested, with a subscriber profile built from other subscriber profiles. Multiple levels of nesting are possible. One example application of this is a subscriber household containing multiple people, each of which use multiple devices. The first level would comprise a separate subscriber profile representing each device, such as a tablet or smartphone, used by a single

person. These device-based subscriber profiles are nested, optionally with other application profiles, within a profile for each person. Finally, the person-based subscriber profiles are nested, optionally along with application profiles representing shared devices such as TV screens, into a top level household subscriber profile. This example is shown in Figure 6. Note that this nesting of profiles may only be needed for specific test cases, such as testing a parental control service that distinguishes between devices within a household service.

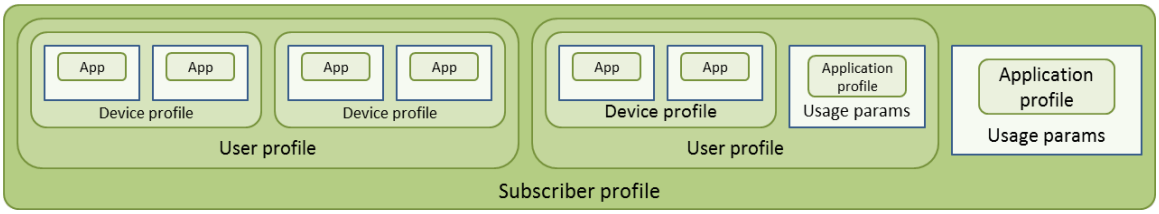

Figure 6 – Nested subscriber profiles

6.3 Test Profiles

Test profiles combine subscriber profiles with data defining how subscribers and applications are instantiated, test sequencing, measurements, information about the SUT, and other data, which together define how a test case is implemented. Test profiles are built from data models that contain the elements listed below (note that the list is not exhaustive).

- Subscriber profiles.
  - The application profiles used for the test case are contained within subscriber profiles (Section 6.2).

- Virtual subscribers to be instantiated using each subscriber profile.
  - Application clients (Section 7) are associated with virtual subscribers and can be inferred.

- Application servers to be instantiated (Section 7). Although the types of application servers necessary may be inferred, their numbers and how they are shared between virtual subscribers may require specification.

- External application servers (if applicable, see Section 7).

- Test methodology (e.g., Monte Carlo) and parameters (e.g., number of trials and randomization parameters).

- Trigger files, optional (Section 8.1).

- Measurement agents and peers (Section 7).

- Measurement sequencing and parameters.

- SUT data.

7 ALT Test Network Architecture

A high level network architecture for Application-Layer Testing is shown in Figure 7. As is typical for lab-based network testing, a test platform provides a set of test control and data plane functions that exercise interfaces connecting to a SUT. The SUT may be any portion of a network of interest, ranging from a single device or component to, for example, an access network spanning the functionality between the customer and network edges. The SUT has two or more interfaces over which test traffic is forwarded. For SUTs with a large number of interfaces, network routers and/or switches typically provide the physical and logical connections between the SUT interfaces, the ALT platform, and any other devices used in testing.

The ALT platform is a set of functional components that work together to implement ALT tests. While it is shown in Figure 7 as a block, physical implementations for the ALT platform are not constrained to any one form factor. The platform can be implemented, for example, as a single device, as a set of special purpose physical devices, or as a set of software functions implemented in a virtualized environment. It is also possible to mix implementation options in different ways for different components, for example using specialized hardware to provide high aggregate rates in the data plane while implementing control plane functions on general purpose compute resources.

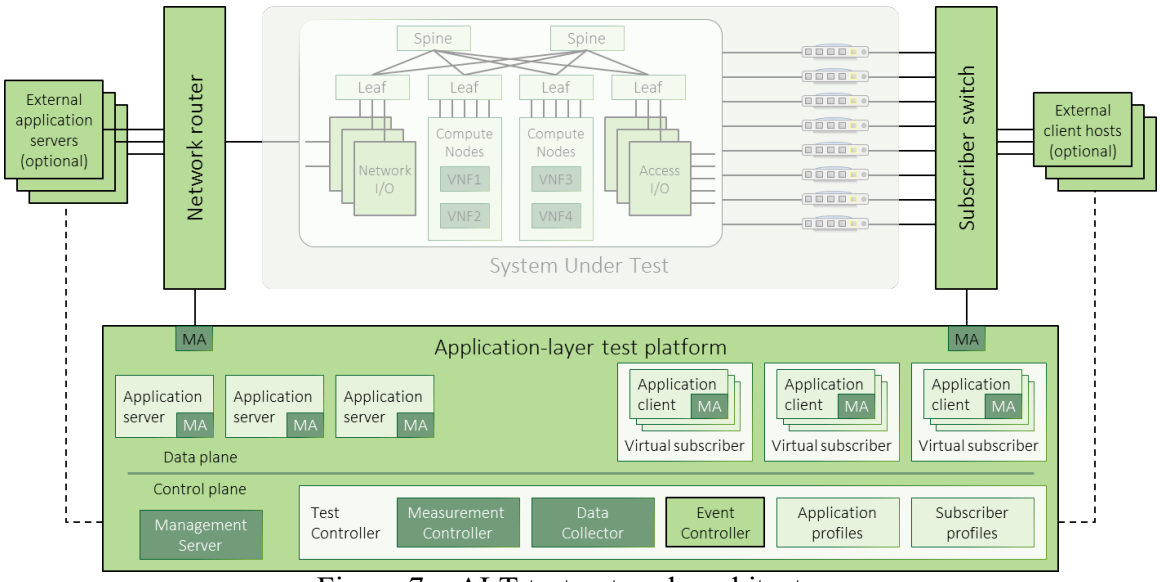

Figure 7 – ALT test network architecture

The functional components making up the ALT platform in Figure 7 are shown as aligned with the data plane or the control plane. Components can also be grouped into two broad categories: those that control the generation of test traffic; and those that control the taking and collection of measurements. In this Technical Report, components associated with measurements are described in the context of the Performance Measurement Framework of TR-304 [4] and associated RFCs [7][8][9]. ALT platforms are not required to use that specific framework, but the equivalent components are needed for a complete measurement system.

The components comprising the ALT platform are described below:

- The Test Controller takes direction from an input mechanism such as an Application Programming Interface (API), a script, or a user interface specifying the test methodology and parameters for a test case. It instantiates application clients, virtual subscribers, and application servers from the input data, as needed for each trial. It also coordinates the activities of the Event Controller, the Measurement Controller, and the Data Collector to implement each trial.

- The Event Controller creates and implements the sequences of events performed in each trial. This includes: pre-generation of trigger timing based on application and subscriber profiles or previously generated trigger files; initialization of each trial; initialization of application sessions during the trial; and notification to the Test Controller of events that trigger measurements.

- Application profiles are created from the data models for each application class, with model elements populated by parameter values specified for a particular test case. Application profiles are described in section 6.1.

- Subscriber profiles combine one or more application profiles with loading parameters to create a set of virtual subscriber behaviors specified for a particular test case. Subscriber profiles are described in section 6.2.

- Virtual subscribers are instantiated for a test case based on the subscriber profiles defined for that test case. Virtual subscribers are described in section 6.2.

- Application clients are instantiated as necessary to implement the client-side application functions specified by the application profiles for the test case. Each application client is associated with a virtual subscriber. Multiple application clients can be instantiated from each application profile for a given test case.

- Application servers are instantiated as necessary to implement the server-side application functions specified by the application profiles for the test case. Multiple application servers can be instantiated from each application profile for a given test case. Depending on the specific application and profile, an application server can serve one or more application clients.

- External applications servers and client hosts may be used for specific test cases that require the full functionality associated with a specific application, as opposed to the generic, abstracted functionality defined for an application class. As an example, video servers set-top boxes, and video monitors may be used to incorporate streams with actual encoded video content into a test case for display.

The following component descriptions are based on TR-304 [4]. More information can be found in that document.

- Measurement Agents perform measurement tasks under the direction of a Measurement Controller. The Measurement Agent registers with, and receives instructions from a Measurement Controller, performs Measurement Tasks, and reports measurement results to one or more Data Collectors. In Figure 7, Measurement Agents (labeled 'MA') are shown at various locations including network interfaces, application clients, and application servers.

- Measurement Peers (not shown) perform measurement tasks in concert with one or more Measurement Agents (and perhaps other Measurement Peers). A Measurement Peer does not communicate with a Measurement Controller.

- The Measurement Controller controls the scheduling and configuration of measurement tasks that are to be done by a Measurement Agent. In an ALT platform, the Measurement Controller coordinates its activities with the Test Controller.

- The Data Collector receives measurement results reported by Measurement Agents.

- A Management Server manages and configures a physical device or network element. Examples include a TR-069 ACS (Auto-Configuration Server), or an EMS (Element Management System). In the ALT platform, the Management Server configures the Management Agents and may configure any routers, switches, and any external application server and/or client host devices.

7.1 Test Interfaces

The test traffic generated by the application servers and clients in an ALT platform needs to be forwarded to and from the correct test interfaces on the SUT. In many test cases, there is a 1:1 correspondence between these test interfaces on the subscriber side of the SUT and virtual subscribers, with each interface corresponding to a single instantiated virtual subscriber. For larger scale systems, traffic is typically forwarded between the SUT and the ALT platform across one or more switches, where there may be a large number of physical SUT test interfaces and a much smaller number of virtual subscriber side interfaces on the ALT platform.

In all cases, the test platform needs to set up appropriate isolation and forwarding mechanisms between each SUT test interface and its corresponding components in the ALT platform. Any suitable mechanism can be used (e.g., VLANs or MPLS), as long as it is compatible with the traffic to be forwarded. As an example, if a test case emulates business subscribers the traffic across the SUT interface could be single- or double-tagged, which could limit the additional levels of VLAN tagging available for isolation and forwarding across the test switches. The paths between SUT interfaces and ALT platform components also need to be non-blocking relative to the performance requirements of the SUT itself.

R-1 The ALT platform and supporting devices MUST support traffic isolation and forwarding mechanisms between each SUT test interface and its corresponding components in the ALT platform.

R-2 The test platform MUST support N:M connectivity between the interfaces connected to the SUT and the virtual test entities (e.g., virtual subscribers) used to implement the test.

R-3 The mechanism used to provide the connectivity MUST support the protocols used for the underlying traffic, e.g., Ethernet services must support IEEE 802.3 MAC addresses.

R-4 Each logical connection between an SUT interface and its corresponding components in the ALT platform MUST be non-blocking.

7.2 Use of Virtualization

As noted above, the ALT platform can be implemented any number of ways, including as a set of functions installed in a virtualized infrastructure. This is equally true of the SUT. Virtualized networks such as CloudCO are implemented in their own infrastructure, but other scenarios might test algorithms for device-level or even component-level functions that are not implemented in a physical device. Other scenarios, such as the analysis of synthesized aggregated traffic discussed in Section 5.6, may focus entirely on test traffic and emulate highly abstracted network systems over which to perform the analysis. When both the ALT platform and the SUT are virtualized, it is possible to implement them in the same infrastructure, using a virtual network to implement the interfaces between VMs or containers.

When components in the ALT platform are virtualized, or when the SUT is virtualized for the purpose of testing, it is important to make sure that the virtualized implementation does not contribute to any meaningful degradation of performance. For example, task switching in a virtualized environment could cause jitter that may not be present in an actual deployed system.

R-5 Each component in the ALT platform MUST NOT degrade the measured performance relative to the requirements of the test case.

R-6 If components of the ALT platform and the SUT are virtualized in the same infrastructure, the SUT MUST be implemented in different VMs or containers than any ALT components.

R-7 If components of the ALT platform and the SUT are virtualized in the same infrastructure, the SUT test interfaces to the ALT platform MUST be distinct.

8 Test Methodologies

Section 4 discusses the complex behavior and variability associated with the combinations of flows generated by the different applications and subscribers that are aggregated on the network. The ability to emulate this level of complexity is an advantage of application-layer testing. At the same time, the variability associated with application-layer traffic can cause a corresponding variability in the measurement results generated by such testing. It is important to recognize the potential for variability and to use test methodologies, metrics and analysis methods that are appropriate for it.

The primary approach to mitigate variability is to generate a large set of measurements that can then be analyzed using statistical methods. The body of literature for the design and analysis of statistical experiments is substantial, and we do not address it here other than to note some issues that are particularly applicable to application-layer testing of networks. Multiple methods, each with advantages or disadvantages in different test scenarios, can be used singly or in combination to generate large measurement sets. Some of these methods include:

- Repeating measurements over a continuous test period. This technique can be especially useful to identify trends that may result from extended operation or many repeated events. An example regarding the long term stability of home gateways is discussed in Section 5.4. One caveat regarding the use of repeated measurements over time is that the period between consecutive measurements needs to be considered relative to the times over which applications sessions operate, since repeated measurements that occur during the same application session can generate correlated results.

- Applying measurements to multiple virtual subscribers or application clients in parallel. This can work well in larger SUT environments where hundreds or thousands of virtual subscribers or applications may be active simultaneously. As with repeated measurements, the level of parallelism needs to be designed considering potential sources of correlation. For example, applying measurements to multiple virtual subscribers on the same PON may generate correlated results if the PON becomes congested. The same type of issue can occur if multiple application clients are served by a common application server that is performance limited.

- Generating multiple test trials, also known as Monte Carlo testing. This technique can be used to prevent correlated results by initiating independently randomized trials between each set of results. It can usually generate independent results using less test time than that required for continuous test periods, since the time required to initialize a test trial is shorter than the period associated with many common application sessions. It does, however, add complexity to the test implementation, and each trial needs to be fully initialized before performing measurements.

Additional techniques can be used to optimize the test time required for a given set of results. One powerful technique for some scenarios is deterministic scheduling. Consider a test case where the metrics of interest are all associated with performance testing, which is an infrequent, brief event compared to applications such as video streaming, and is responsible for a small percentage of the traffic generated by subscribers. It would make little sense to initiate a continuous test and to wait

hours on average between consecutive performance tests to capture measurements. Instead, specific application sessions can be scheduled to begin at pre-determined, non-random times. Using this method, a Monte Carlo trial is initialized and the SUT is given time to come to a post-initialization state, after which a "test" virtual subscriber initializes a performance test at a scheduled time. This sequence can be repeated thousands of times, using randomized initialization parameters for each trial, in the time it would take to capture a handful of randomly scheduled performance tests.

R-8 An ALT platform MUST support Monte Carlo testing.

R-9 An ALT platform MUST support both random and deterministic scheduling of events.

8.1 Trial Randomization and Replication

Trial randomization and duplication are complementary aspects of statistical testing, and each provides benefits under the right circumstances. In most test scenarios, we want application sessions to be initiated at random times with random session sizes, with the understanding that the statistics of each variate will be consistent with the parameter values specified for that variate. As a practical matter, test systems need to support randomization of the seed used at startup to prevent unintentional duplication of test event sequencing.

R-10 An ALT platform MUST support generation of random sample values consistent with the statistics specified for each variate.

R-11 An ALT platform MUST NOT generate an identical sequence of random sample values in response to an identical set of test parameters unless configured to do so.

Under specific conditions, it is highly desirable to be able to replicate a test sequence as closely as possible, for example to attempt to recreate an error condition. One way to do this, while also supporting portability between test systems, is to specify the pseudorandom generator used to generate samples. A much easier method is to pre-generate a trigger file containing the start times and session parameters for each application session instantiated during the trial. The trigger file is then used to control the trial. The same trigger file can be used as many times as necessary to duplicate the trial conditions.

The ALT platform can also generate a log file during the trial, recording the timing and parameter values associated with each event as it occurs. Log files and trigger files, however, have different uses. While log files have value for debugging and other purposes, network congestion and rate adaptation can cause the timing and sequencing of events as logged to be different from those in the original trigger file, making log files unsuitable for trial duplication.

R-12 An ALT platform MUST support generation of trigger files that document the specific start times, session durations, and other parameters used to generate application sessions in a test trial.

R-13 An ALT platform MUST support the use of trigger files as inputs to control the event timing, sequencing and other session parameters for a trial.

R-14 An ALT platform MUST support generation of log files to document events as they occurred within a trial.

8.2 Initialization of trials

When a trial is initialized, we want to bring the SUT to a "steady state" as quickly as possible in order to start performing meaningful measurements. In this context, "steady state" means a state in which the network traffic and performance conditions are no longer affected by startup transients. Under most circumstances it is not possible to achieve steady state immediately after initialization, since network and application buffers need time to stabilize and adaptive protocols such as TCP need time to complete initial stages such as slow start. We can, however, come as close as possible to starting with random application session states by "priming" the trigger file. The priming process comprises the following steps:

1. Specify a time period T_p long enough to ensure that traffic conditions at the beginning and end of the period are effectively uncorrelated.

2. Given the desired trial duration T_t, generate a sequence of events from $-T_p \leq t \leq T_t$.

3. Calculate the expected state of application sessions active at $t = 0$. For applications generating fixed rate traffic, this is typically a matter of calculating the time required to send the traffic for each session, determining which sessions are active at $t = 0$, and calculating the traffic remaining to be sent by each session at that time. Applications that send adaptive rate traffic can be approximated by an appropriate fixed rate such as a service rate, or a degraded service rate for scenarios dominated by congestion. Since the ALT platform has little or no knowledge of the network or service parameters inside the SUT, the appropriate fixed rate approximations for adaptive rate applications need to be configurable inputs in the test profile.

4. Create session initiation events at $t = 0$ for the active application sessions, with session sizes adjusted to match the traffic volume or time remaining for each session as estimated in the above step.

The above priming process initializes applications at random phases, minimizing the time required for the SUT to achieve steady state.

R-15 An ALT platform MUST support the priming process described above for trial initialization.

8.3 Sources of Variation

While variation in the test traffic generated is expected and desirable, sources of unintended variation should be minimized or eliminated wherever possible. In testing at the application layer, some of these sources can be traced to the depth of the network stack and the multiple protocols involved:

- Application profiles need to be specific with regard to the protocols used to generate the test traffic. Especially at higher network layers, different protocols may be used by different applications within an application class. Whenever possible, a single representative option should be specified.

- The implementations for some protocols such as TCP have multiple versions and options, not all of which will be available on any one test platform. Ideally, the desired TCP versions and options would be specified in the test plan and used by the test platform. Failing that, the versions and options used can be specified in detail.

Appendix A – Representative Application Classes

Several representative application classes are described below. Note that the sections below describe the typical behavior of application classes as of the time of publication. The protocols used and the associated application behaviors are subject to change as new applications are introduced in the field.

A.1 Streaming Video Application Class

Streaming video contributes the largest share of traffic for any application class used by residential broadband subscribers. The application is frequently delivered on an OTT basis within Internet access services, but it can also be used for Video on Demand offerings within managed video services. A variety of protocols have been implemented for HTTP video streaming, in particular DASH, HTTP Live Streaming (HLS), HTTP Dynamic Streaming (HDS), and Microsoft Smooth Streaming Protocol (MS-SSTR). These protocols share the following common behavior:

- All are client-server protocols, in which a server (typically located in the network) sends encoded video content to a client (typically located at a customer premises), which buffers, decodes, and plays out the video.

- The video server has local storage access to a selection of videos, each of which is stored as a series of "chunks." Each chunk contains a few seconds of video content, encoded at one of several bit rates. Chunks are encoded at multiple bit rates, with each chunk for a specific portion of the video spanning the same playout time.

- When a client requests a video, the server provides information on the available bit rates. The client then requests the first few chunks (using HTTP GET) at the desired bit rate to fill its receive buffer. The chunks are delivered over one or more TCP connections. Once enough chunks have been delivered, the client starts to decode and play the video.

- As the client receives chunks, it records the relative time each chunk is received along with its size. This allows the client to estimate the available throughput from the video server.

- As the client decodes and plays the received video, it requests additional chunks as space becomes available in the receive buffer. At any point in the video, the client can request chunks at a new bit rate. If the network cannot deliver content as fast as it is being played out, the client requests subsequent chunks at a lower bit rate. Conversely, if chunks are being delivered faster than the playout rate and higher quality encoding is available, the client can request the higher rate.

Video can be captured and encoded at several major tiers of quality, including SD (Standard definition), HD (High definition), 4K (also known as UHD or Ultra high definition) and 8K. These tiers can be parametrized separately from the adaptive rates at which a specific video is encoded. For example, an HD video could be encoded at 8, 6, and 4 Mbps, while an SD video might be encoded at 2, 1.5, and 1 Mbps. Sessions are time-based, with distributions that may be exponential

for short-form video or uniform for long-form video. An ALT streaming video application can incorporate a parameter to set the relative frequency of each major quality tier.

A.2 VoIP Application Class

VoIP sends packets containing encoded voice content concurrently in both directions between two peers. The content is typically encoded at rates from 2.4 kbps to 64 kbps (or higher for wideband audio) and sent over UDP using small packets at regular intervals of 10 to 20 msec. Sessions are time-based with an exponential size distribution. A basic ALT VoIP application could send voice packets between two predefined hosts without implementing the call setup signaling required in the field. Other implementations could include signaling, perhaps making them useful over a broader range of test scenarios.

A.3 Video Conferencing Application Class

Video Conferencing sends packets containing encoded video and audio content concurrently in both directions. Video Conferencing applications can be on a peer-to-peer basis between two endpoints, or they can be multi-user, with multiple endpoints logged into a conferencing server. The content is encoded at rates that may range from hundreds of kbps for lower quality video over bandwidth constrained network paths, to tens of Mbps for high quality video. Content is transmitted in bursts of large packets corresponding to video frames, at video frame rates ranging from 15 to 60 or more frames per second. Most video conferencing applications can adapt to a rate that can be reliably sent across the network path.

Most video conferencing applications use ITU-T H.323 [16], which in turn specifies a number of other ITU-T and IETF protocols for signaling, call admission and registration, control, and encoding/decoding of content. As with VoIP, a low fidelity emulation of video conferencing traffic could send bursts of packets emulating video frames between two predefined hosts without implementing signaling or other protocols, while higher fidelity implementations could include a full suite of protocols.

The characteristics applicable to the video conferencing application class include:

- Direction is concurrent downstream and upstream with symmetric rates.
- Session size defined as time, with an exponential distribution.
- Transport protocol is UDP. Applicable protocols are specified in ITU-T H.323.
- Both the transmission rate and the video frame rate can be adapted at the application layer.

The metrics relevant to the class include:

- QoS network metrics: latency; jitter
- Application-level metrics: receive buffer overrun or underrun; adaptive rate

A.4 Web Browsing Application Class

A web browsing session can be characterized as a series of events, where each event is the download of a single page of content. Events are volume-based and can be modeled with an exponential or gamma-shaped distribution. The time between events within a session is also

modeled, using an exponential distribution. The web browsing session comprising these events is itself time-based with an exponential distribution. Web browsing uses HTTPS or HTTP and transfers content over TCP.

There are multiple sub-classes of web browsing, including: finite pages vs. "infinite scrolling" pages; search engines vs. destination sites; passive browsing vs. social media, etc. While it is possible to create statistics for each subclass, a base "web browsing" class that incorporates the subclasses in a single set of statistics may serve most testing purposes. Some more detailed characteristics of web browsing, as well as distinctions between subclasses, are summarized below.

- Web sites other than search engines typically organize content as multiple objects that are referenced by one or more index files. These objects may be downloaded over multiple TCP connections, with websites sometimes generating dozens or even hundreds of connections to load graphics, images, and embedded videos from multiple IP addresses in the network. Embedded links to new IP addresses may also spawn DNS lookups from the client host.

- Search engine sites are frequently optimized for low response time, minimizing the number of different objects and TCP connections required as well as the downloaded page size.

- Many web sites incorporate streaming video and/or games that the user can access from within their browser. These embedded applications generate streaming video or gaming traffic with behavior different from the "page loading" behavior described above. There are two approaches to modeling this behavior:

  - A compound application class based on web browsing could spawn video and/or gaming sessions with their own statistics. This would generate time-domain behavior that links web browsing events with embedded video or gaming events.

  - The proportion of video and gaming traffic in a subscriber profile could be set to account for traffic spawned by web browsing. The resulting traffic would be independent of web browsing traffic, which may be acceptable for many test scenarios.

- Social media sites incorporate the ability to upload pictures and videos, as well as text-based uploads. This traffic can be modeled either with a hybrid application class or as independent file uploads, similar to the approaches discussed above for streaming video and gaming.

- Some websites implement "infinite scrolling", causing additional data loads as the user scrolls through the page. These loads can generally be modeled the same as other web-based download events.

A.5 Upstream Security Video Application Class

Video streamed from residential and business surveillance cameras is one of the largest sources of upstream traffic, accounting for nearly 10% of upstream traffic in North America and over 4% globally. Surveillance cameras can be paired with cloud storage services, where the camera is configured to stream video to cloud storage on a fixed schedule or in response to triggers such as

motion detection. Some camera and cloud service combinations can stream video continuously. Finally, cameras can stream to remote devices on demand.

The video generated by surveillance cameras is typically encoded as an MPEG, H.264, or H.265 stream at rates ranging from several hundred kbps to ten or more Mbps. The streaming rate is typically adapted at the application layer in response to network performance. Most products use Real Time Protocol (RTP) or Secure Real Time Protocol (SRTP) over UDP to reduce latency for support of real time Pan, Tilt, and Zoom (PTZ) features, but streaming over TCP has been observed in some products. Business class surveillance products may conform to specifications from the Open Network Video Interface Forum (ONVIF), an industry forum that specifies interfaces for interoperability of IP-based security products.

The characteristics applicable to the upstream video application class include:

- Direction is primarily upstream, with lower bandwidth control plane and feedback traffic in the downstream direction.
- Upstream data rates range from hundreds of kbps to ten or more Mbps. The data rate is typically adaptive at the application layer.
- Session size is defined in terms of time. Randomly scheduled sessions, such as those initiated on demand or triggered by motion detection, have exponential distributions, while deterministically scheduled sessions can have pre-configured sizes, or can be continuous.
- The protocols for the video stream are typically RTP or SRTP over UDP, although TCP has also been observed for transport. Control plane protocols include RTCP, RTSP, and HTTP.

Application-layer metrics for upstream video include rate adaptation and receive buffer underruns/overruns caused by jitter or network slowdowns or interruptions. Network QoS metrics include delivered throughput, jitter, and latency.

A.6 File Transfer Application Class

Large file transfers are increasingly common, in part due to high resolution video captured in raw format by phones and cameras. Video and other files may be uploaded or downloaded with any number of cloud-based file storage and backup services, leading to transfers of hundreds of megabytes or more. Gaming also generates large file transfers with files sometimes exceeding 10GB. Other prominent sources include, but are not limited to, OS updates, cloud-based file backups, and social media platform uploads.

File transfers typically use one or more TCP connections for transport. Multiple files may be transferred serially, or they may be packaged together in a compressed format such as a .zip or .tar for transfer. Above the transport layer, file transfers may be initiated by events such as HTTP GET, PUT, or POST methods or by proprietary protocols. The TCP payloads are frequently encrypted, hiding higher layer information.

The primary metric for large file upload and download is throughput. This application class has potential to be very disruptive to other activities, generating traffic as fast as possible, and may affect other users on a shared medium.

This application class has multiple subclasses, as many types of distribution services rely on frequent file downloads and uploads. These distribution services may have widely varying session parameter values, but they all treat files in a similar fashion.

- Cloud storage
  - Frequent uploads of varying size as a user generates files to be backed up. Infrequent large downloads as previously synced files are restored from the backup by a user. Infrequent large uploads as a user manually places files into cloud storage. Application servers may limit the transfer rate used for remote backup or other applications.

- Online media distribution
  - Large downloads as games and locally stored video files are downloaded from a distribution service upon user request.

- Operating System Updates
  - Stochastic or user requested large downloads based on operating system configuration and update availability.

These subclasses can be emulated using HTTP and TCP with the appropriate session size and loading parameters applied to application and subscriber profiles. The application class data model should include parameters for:

- Session size statistics and distribution model
- Direction
- Number of TCP connections (max and min)
- Server-imposed rate restriction

A.7 Peer-to-peer File Sharing Application Class

Peer-to-Peer (P2P) file sharing applications allow users to share files directly with each other, without relying on a network server as an intermediary destination. The applications are also used by businesses to distribute files such as software updates and games to large numbers of customers without requiring the server capacity that would normally be associated with client-server based distribution. This application class is one of the primary contributors to upstream traffic volume from residential subscribers.

Multiple P2P protocols can be used, including BitTorrent, eDonkey, and Gnutella. The main functions associated with a P2P protocol/network are: enabling peers to discover and communicate with other peers on the network; enabling peers to discover the files available on the network; and providing files to a peer on request. Some protocols use servers to accomplish the first two functions, while others distribute those functions among the peer clients themselves. Protocols also take different approaches to file transfer: several, including BitTorrent and eDonkey, break files into

multiple pieces and distribute the file transfer mechanism among multiple peers; while others such as Gnutella transfer a file from one peer to another, sometimes using a mechanism as simple as an HTTP GET request.

P2P applications differ from other applications in several ways that can affect the design of application-layer test cases:

- File transfers take place from one client to another, rather than from server to client or vice versa. This does not mean that virtual subscribers should transfer files to each other in an ALT test, however, since in most cases an SUT with only a few virtual subscribers will be much smaller than the scale of an actual P2P network, creating an unrealistically large level of correlation in the associated traffic patterns if they were to transfer files to each other. Instead, each virtual subscriber using the application independently transfers files to and from peers emulated at the remote end of the network.

- The total volume of P2P traffic in the upstream and downstream direction can be approximately symmetric. However, if the P2P protocol divides files into pieces and distributes the transfers among multiple peers, the time domain behavior in the upstream and downstream directions is different, with multiple pieces of a requested file sent downstream in close proximity, but single pieces of different files sent upstream at unrelated times.

- The percentage of subscribers using P2P applications is smaller than those using applications such as streaming video and web browsing. Those subscribers that do use P2P regularly, however, can generate large volumes of traffic, especially in the upstream direction.

- If the P2P class is used strictly to generate traffic with the desired time domain behavior, it can be emulated with HTTP GET and PUT requests to a remote server. Actual P2P protocols and simulated networks can be used when needed to achieve higher levels of fidelity.

The characteristics applicable to the P2P application class include:

- Direction is both downstream and upstream with independent sessions in each direction. Traffic volumes can be approximately symmetric, or they can be weighted in either direction.

- Session size is defined as volume. An upstream session is defined as a single file. A downstream session can be defined as a single file or a group of related files.

- Session size distribution is exponential or Pareto. Upstream and downstream session size distributions can have different characteristics.

- Transport protocol is TCP. A P2P application may open multiple TCP connections, resulting in a high transfer rate that can disrupt other traffic. Multiple application protocols exist,

including BitTorrent, eDonkey, and Gnutella. Application protocols may use HTTP or other protocols.

- Rate adaptive transmission using TCP.

Upstream peer-to-peer traffic is initiated remotely and, while the traffic generated can affect the performance of other applications, its performance does not affect QoE associated with the P2P application itself. Downstream P2P traffic can be initiated by the subscriber, so file transfer time can be measured as an application-layer metric.

A.8 Gaming Application Class

Gaming application traffic is heavily dependent on the particular game's networking implementation. A session, defined as the duration a player spends connected to a networking server during a game, may range anywhere from a few minutes to several hours.

A game's sensitivity to network performance network utilization is based on its client/server model and where the "authority" over the game world lies. The traffic generated by a game depends on how much and how often game state is traded between a server and client.

There are many ways of dividing the authority, most of which are based on some combination of the following methods:

- **Authoritative Server:** The server holds the only real state of the game and passes it out to each client. In its simplest form, this method is very latency-sensitive, as each player must wait for the server to provide state updates.

- **Peer-to-Peer Lock-Step:** The game has one defined state, which is shared between each client with only one client allowed to play at a time. This method is most suitable for turn-based games, such as card games. Game latency is sensitive to all players' network latencies, as the game will run at the slowest player's pace.

- **Non-Authoritative Server or Client-side Authority:** The client decides how the world is updated, and the server simply syncs this information with all players. This method is seldom used without modification, due to drawbacks related to cheating and lack of synchronization between the different authorities involved. It is, however, much less sensitive to latency.

Due to the competing needs for authority and latency for player QoS, methods have been devised to push some of the authority and much of the computation to the client side, lowering the requirement on network latency. These methods require the client to predict the future state of the world as accurately as possible, since any fault in prediction will cause a noticeable lag and backwards jump when the state is re-synced. By keeping the player's ability to affect change relatively simple, the predications can be made very accurate, and the resulting faults very rare. Authoritative servers with many of these optimizations are common in modern interactive games.

Most games are designed using a game engine such as Unity, Source, or Unreal. Often these engines will define the client server model being used, determining their sensitivity to network latency. The traffic generated by a game depends on how much data is required for synchronization to each client and how often it is done.

Large game developers often fine-tune the networking side of their game implementations to improve and optimize the network performance of their game beyond that provided by a third party game engine. This type of optimization is resource-intensive and may not be done by smaller developers.

Several generalizations can be made to create a consistent model of Gaming traffic.

- A typical interactive video-based game consumes a low sustained level of traffic (e.g., <1 Mbps). Overall message complexity and communication to the server or other players is kept as low as possible for efficiency reasons, and as much computation as possible is done client side.

- Traffic is of a relatively constant background level, with larger bursts occurring as assets (graphics, game worlds, sounds) are downloaded, or larger amounts of player interaction are being synchronized with the game server.

- Different types of games utilize different transport protocols due to the type of gameplay and player interaction they support.

    - Real-time games such as First Person Shooter games typically rely on UDP to transmit data, with any mechanisms for guaranteed or ordered delivery of packets handled within the game's networking system. Traffic for these games may be split into multiple streams based on the delivery requirements for each stream, such as:
        - Guaranteed delivery with lowest possible latency.
        - Guaranteed and ordered delivery. Even this traffic may be sent over UDP, with the game's internal networking system ensuring ordered delivery.
        - Most recent state data, which is retransmitted only if it is still up to date.
        - Non-guaranteed data, which is never retransmitted.

    - Slower paced games, which can handle more latency, for example Massively Multiplayer Online Role-Playing Games (MMORPG) or Card games, rely on TCP. These are often very susceptible to increases in peak latency, which can momentarily decrease the perceived QoE. To reduce latency under low load conditions, Nagle's algorithm is typically disabled.

- As almost all game data is time sensitive, individual communications with a server are relatively small and frequent, meaning a high number of individual message-passing events, each of small size.

References for this material including further descriptions of gaming network implementations can be found in the following locations:

- Mark Frohnmayer, Tim Gift, "The TRIBES Engine Networking Model," https://www.gamedevs.org/uploads/tribes-networking-model.pdf - Retrieved 2/13/2019
- Marc Bastiaansen, "Networking in Unity," https://www.3dgep.com/networking-in-unity-3-5/ - Retrieved 2/13/2019
- Glen Fiedler, "Gaffer On Games," https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/ - Retrieved 2/13/2019

A.9 Virtual and Augmented Reality Application Class

Virtual Reality (VR) uses wearable stereoscopic displays, earpieces, a variety of sensors and input devices, and additional techniques such as haptic and other sensory feedback to immerse the user in a virtual environment that is as realistic as possible. Augmented Reality (AR)[1] uses similar technology with translucent or transparent wearable displays to integrate virtual elements into the user's real-world environment. Both technologies, which are sometimes discussed together under the term Extended Reality (XR), can be used with games, interactive videos and stories, telepresence, and a variety of other business and recreational applications.

The traffic loads placed on networks by XR applications can vary widely, depending on the nature of the application. If an application depends on real-time video from a remote source, the downstream load can be extremely heavy, with the equivalent of 4K or even 8K video streamed to each eye. In addition, the video must be delivered with extremely low latency and jitter to avoid noticeable lag between head movements and the resulting change in perspective, which can cause disorientation and even nausea in severe cases. AR applications dependent on remote processing place similar requirements on downstream throughput and delay, and can also place stringent upstream throughput and delay requirements on the network if they need to communicate real time information about the user's environment to the remote processor. At the time of publication, nearly all mass market VR applications are developed to work over best effort Internet access services as they are currently widely offered, so the applications avoid such stringent requirements by processing content locally and limiting network traffic to either non-real time downloads or the types of environmental state updates associated with non-VR games. When and where network services support the necessary QoS however, designers may offload as much processing as possible to minimize the size and weight of the XR headset.

At the current state of the art, VR application sessions tend to be 10 to 20 minutes on average because of user fatigue caused by disparities introduced by the VR headset, for example between "accommodative demand" (the distance at which the eyes focus) and "vergence demand" (the degree to which the eyes must rotate towards each other to converge on a nearby object). This fatigue can be exacerbated by issues such as video at a lower than optimum resolution or frame rate, or lag between user movements and corresponding changes in the field of view.

Due to the requirements discussed above, TCP cannot be used for real-time video streamed for XR applications, so real time video (when it is used) may be provided over UDP and a higher layer real time protocol such as RTP. Non-video information has similar requirements and can be delivered in

[1] Some AR applications augment the video displays on non-wearable devices such as smartphones and tablets. This discussion focuses on immersive AR using wearable displays, which can impose stricter requirements on the network.

the same way as gaming traffic (Section A.8) – in fact, VR applications use many of the same engines used by the gaming community.

The characteristics applicable to the XR application class include:

- Direction is concurrent downstream and upstream. Multiuser and other networked applications may have real time requirements associated with some or all of the network traffic. Even for games and other applications that are locally rendered, downstream updates to the environment typically generate more traffic than upstream notifications. If most or all processing is offloaded to the network, the required real-time downstream load can be many tens of Mbps for both VR and AR, and the real-time upstream load can be significant for AR.

- Session size is defined in terms of time, with sessions that are typically shorter than those for non-XR games.

- Session size distribution is exponential.

- Transport protocol for real time traffic is UDP. Additional, non-real time traffic may be transported over TCP. Application-level protocols may be defined by the engine applied to a specific application.

- If real time networked video is used, it is rate adaptive at the application level.

The most critical metrics associated with XR are latency jitter and packet loss, with stringent requirements on latency and jitter for real-time transmission. If real-time networked video is incorporated in the application, application-level metrics related to the delivery of that video become critical as well.

A.10 Performance Testing Application Class

Performance tests are widely available as web sites or apps from service providers, regulators, and third parties. The tests typically follow a fixed sequence of events:

1. Some applications perform a ping test to measure round trip latency between the client and the performance test server. This is neither universal nor required for the remaining steps.
2. The application then opens a number of TCP connections, followed by an HTTP GET request on each connection to generate a large file download test.
3. Once the test has run for a time sufficient to measure throughput on all connections, the application terminates the download connections.
4. The same procedure is used to measure upload throughput on each connection, using a request such as HTTP POST.

End of Broadband Forum Technical Report TR-421