

LMAP Working Group
INTERNET-DRAFT
Intended Status: Informational
Expires: July 13, 2015

L. Deng
China Mobile
L. Zheng
Huawei
M. Ackermann
BCBS Michigan
G. Mirsky
Ericsson
Jan 11, 2015

Use-cases for Passive Measurement in Wireless Networks
draft-deng-ippm-passive-wireless-usecase-01

Abstract

This document presents use-cases for passive IP performance measurements in wireless networks.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
2	Conventions Used in This Document	4
2.1	Terminology	4
2.2	Requirements Language	4
3	Use-cases	4
3.1	Performance Monitoring for Network Planning/Optimization . .	4
3.2	End-to-end Measurement for Wireless Subscribers	5
3.3	Accurate Fault Identification	6
4	Security Considerations	8
5	IANA Considerations	8
6	References	9
6.1	Normative References	9
6.2	Informative References	9
	Authors' Addresses	10

1 Introduction

It is well-accepted that mobile Internet usage is going to increase fast in the coming years and replace the traditional voice service as the dominant revenue source for mobile operators. In the meantime, fast evolving network and terminal technologies and changing service trends (e.g. social networking, video on demand, online reading, etc.) result in more stringent user service requirements. Therefore, as the basic infrastructure service providers, operators are deemed responsible for mobile Internet end-to-end performance because subscribers want to get what they want, which gives rise to a basic yet important question: how does network service provider manage end-to-end Quality of Service (QoS)? In particular, there are two goals for operator's quality management initiative:

- o to make sure and validate the QoS metrics of specific IP flows against the values pre-defined by the service Service Level Agreement(SLA) from the perspective of either the subscriber or the Internet Content Provider (ICP); and
- o to make sure and validate the sanity of network devices/links.

Passive measurements, where observation on existing traffic is the only means for measurement entities, have been extensively used in scenarios where active measurement alone may not be sufficient to characterize performance over a particular service path. For example, the active measurement traffic may not be in-band with the real traffic that it is intended to simulate as a result of dynamics in routing techniques, e.g. Equal Cost Multi-Path (ECMP) [RFC2991] or device pooling[3GPP TS23.236].

Overall there are many characteristics of injected active test traffic that can render behaviors and measured metrics may be different from the actual user traffic flows and performance. Since the ultimate goal is understanding the actual user traffic performance, measuring the actual (Passive) traffic itself, represents an important measurement method to achieve effective and accurate results.

In this draft, we present three use-cases of passive measurements for wireless networks, where active IP performance measurements are not desirable or accurate enough in achieving the above goals.

It is worth notice that some of the use-cases are not unique to Wireless networks, and can be applied to wired networks as well.

2 Conventions Used in This Document

2.1 Terminology

ECMP - Equal Cost Multi-Path

ISP - Internet Service Provider

QoE - Quality of Experience

QoS - Quality of Service

RAN - Radio Access Network

SLA - Service Level Agreement

UE - User Equipment

MP - Measurement Point, a node or juncture within an IP Network Session Path at which information regarding the performance or reliability of the session path is observed, examined or measured.

Ma - Measurement Agent, the software entity integrated into a measurement point that actually does the function of performance measurement related tasks.

2.2 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3 Use-cases

In light of the introduction of more capable passive measurement methods than pure observation in[I.D-zheng-ippm-framework-passive], it is expected that passive measurements would be the basic building block in performance monitoring in highly dynamic and resource-limited production networks like wireless access networks.

This section presents use-cases for passive measurements in wireless networks.

3.1 Performance Monitoring for Network Planning/Optimization

As mentioned earlier, it is important for Internet Service Providers

(ISPs) to understand their network performance through continuous and accurate performance monitoring in terms of the experience of network customers in addition to the status of the physical network.

However, due to the traffic dynamics in terms of its geographic and time distribution, accurate monitoring of actual traffic QoS, necessary for network planning or adaptive network optimization, cannot be achieved by active measurements alone. Because active measurement methods measure performance metrics by means of carefully designed and injected active measurement traffic, whose characteristics may be quite different from those of the real traffic in a production network, and not flexible to account for the impact from traffic dynamics. Moreover, the injected active traffic could even skew results or measurements, rendering the continuous non-interfering monitoring of traffic QoS impossible with active measurements along. This could be especially problematic when associated results are used for performing network planning and optimization. E.g. for network planning, it is important to evaluate the Quality of Experience (QoE) and network performance during the peak hours, when active measurements are least desirable. It is also helpful to understand the user experience during non-peak hours in order to better assist the application and verify its sophisticated dynamic resource provisioning schemes, such as elastic resource pooling.

Alternatively, deploying passive measurement points/agents in the wireless network, operators can draw a continuous graph of the network usage and performance metric as the basis of network/resource planning. Since interference to network performance introduced by data collection of a passive measurement may be viewed as negligent, it can be initiated almost any time of the day and applied to rush hours as well.

3.2 End-to-end Measurement for Wireless Subscribers

For wireless networks, almost all the time, the wireless "last mile" would be the bottleneck for end-to-end QoS and QoE, indicating the necessity to include the wireless segment into the measurement path.

However, due to the limited availability and/or relatively high cost of wireless resources, it is not economic for either ISP or the user to conduct resource-demanding active measurements over the wireless link.

For instance, unlike the fixed network providers where the access network resource is shared by a group of subscribers who are charged by the duration of their subscriptions independent of their actual

network usage, wireless/mobile ISPs make extensive use of resource allocation and reservation for individual terminals/IP flows and often use the traffic volume consumption as the basis for subscription billing. In other words, the subscriber may be charged for the active measurement traffic despite the fact that it degrades QoE of real application data transfer during the Active measurement.

Therefore, measurements pertaining to the performance of Subscribers or End Users, are particularly dependent upon passive measurements. As stated, Active measurements conducted in this realm can be expensive and even affect QoE. Possibly even greater concern is that the results of the Active measurements may not match the results of the actual End User traffic (for various reasons discussed in Section 3.1). The Passive measurements more likely match the results of the actual End User traffic, because they are based on the same traffic.

3.3 Accurate Fault Identification

It is quite common that there are multiple domains (belonging to different operational or administrative bodies) along the data path from a mobile user equipment (UE) to the Internet.

Case 1: intermediary ISP: consider the example of a mobile subscriber getting access from a 3GPP network. Besides a local mobile network operator, intermediary ISPs may exist in between before subscriber's traffic reaches the Internet.

Case2: path partitioning: as shown in Figure 1, within the local operator's network, radio access network (RAN), RAN backhaul and local core network could actually be constructed and managed by staff from different departments.

Case3: geographic partitioning: for large operators, employing layered network operation and management architecture based on geographic partitions, there may be a further more subpath partitioning between local IP backhaul (managed by state sub-ordinaries) and national IP backhaul (managed by headquarters).

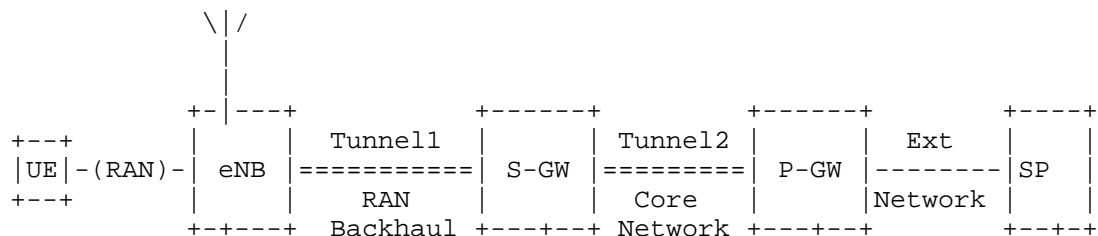


Figure 1: Example of path partition in 3GPP network

Case4: roaming partitioning: Moreover, for roaming cases under home-routed mode, all the traffic from a roaming UE would first traverse from the visited ISP and potentially another Internet operator before getting back to homing ISP network.

In these cross-domain scenarios, in order to do effective trouble shooting for degraded QoS, one needs to first identify the faulty domain or cross-domain interconnection from well performing domains, and then further drill down for the overloaded device/link within the identified domain. If Active measurements are employed, cross-boundary traffic and cross-provider coordination on the interconnections may be required to complicate the process.

On the contrary, passive measurements can help in accurate trouble-shooting and problem demarcation between various networking technologies or operational domains that together compose an end-to-end traffic path, since it does not require extra cross-boundary traffic to be injected into the path or strict synchronization to be conducted between participating measurement points/agents as Active measurements do.

Passive measurements can be used both for the end-to-end problem identification and the hop-by-hop demarcation. By deploying measurement points/agents both within the domains and at the cross-boundary interconnections, passive measurements can quickly identify the faulty domain/device/link without introducing extra cross-boundary measurement traffic.

For instance, Passive measurement points/agents can be deployed at both the ingress and the egress point of each domain and work independently along the path for the passive performance measurement. A simple aggregation at a third-party data collector can do the drilling measurement result analysis to identify the problematic flow.

More importantly, in the above cross-domain cases, timely fault isolation is critical. Alerts/alarms and other indications of potential faults may be provided more quickly by monitoring and measuring on data traffic. As alluded to in the previous paragraphs, active monitoring may require significant set up and coordination. By the time this occurs, it is conceivable that network conditions, may have changed. It is also conceivable that the difference in traffic characteristics between the actual traffic, and active traffic injected into the network, (no matter how slight the differences), may not experience the same issues or faults.

4 Security Considerations

TBA.

5 IANA Considerations

There is no IANA action in this document.

6 References

6.1 Normative References

6.2 Informative References

[I.D-zheng-ippm-framework-passive] L. Zheng et al. "Framework for IP Passive Measurements", draft-zheng-ippm-framework-passive-00(work in progress), June 2014.

[ECMP] D. Thaler et al. "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[3GPP TS23.236] 3GPP TS 23.236: "Intra Domain Connection of RAN Nodes to Multiple CN Nodes", Release 5, November 2004.

Authors' Addresses

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Michael Ackermann
Blue Cross Blue Shield of Michigan
USA

Email: mike.ackermann@bcbsmi.com

Greg Mirsky
Ericsson
USA

Email: gregory.mirsky@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 2, 2014

J. Hedin
G. Mirsky
S. Baillargeon
Ericsson
March 31, 2014

Differentiated Service Code Point and Explicit Congestion Notification
Monitoring in Two-Way Active Measurement Protocol (TWAMP)
draft-hedin-ippm-type-p-monitor-03

Abstract

This document describes an OPTIONAL feature for TWAMP [RFC5357] allowing the monitoring of the Differentiated Service Code Point and Explicit Congestion Notification fields with the TWAMP-Test protocol.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 2, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions used in this document	3
1.1.1. Terminology	3
1.1.2. Requirements Language	3
2. TWAMP Extensions	4
2.1. Setting Up Connection to Monitor DSCP and ECN	4
2.2. TWAMP-Test Extension	4
2.2.1. Session-Reflector Packet Format for DSCP and ECN Monitoring	4
2.2.2. DSCP and ECN Monitoring with RFC 6038 extensions	7
2.2.3. Consideration for TWAMP Light mode	8
3. IANA Considerations	8
4. Security Considerations	9
5. Acknowledgements	9
6. References	9
6.1. Normative References	9
6.2. Informative References	10
Authors' Addresses	10

1. Introduction

One-Way Active Measurement Protocol (OWAMP) [RFC4656] defines Type-P descriptor and negotiation of its value in OWAMP-Control protocol. Two-Way Active Measurement Protocol (TWAMP) [RFC5357] states that only Differentiated Service Code Point (DSCP) value can be defined by Type-P descriptor and the negotiated value must be used by both Session-Sender and Session-Reflector. The TWAMP specification also states that the same value of DSCP (found in the Session-Sender packet) MUST be used in the test packet reflected by the Session-Reflector. However the TWAMP-Test protocol does not specify any methods to determine or report when the DSCP value has changed or is different than expected in the forward or reverse direction. Re-marking the DSCP (changing its original value) in IP networks is possible and often accomplished by a Diffserv policy configured on a single node along the IP path. In many cases, a change of the DSCP value indicates an unintentional or erroneous behavior. At best, the Session-Sender can detect a change of the DSCP reverse direction assuming such change is actually detectable.

This document describes an OPTIONAL feature for TWAMP. It is called the DSCP and ECN monitoring feature. This feature allows the Session-Sender to know the actual DSCP value received at the Session-Reflector. Furthermore this OPTIONAL feature also tracks the Explicit Congestion Notification (ECN) value received at the Session-Reflector. This is helpful to determine if ECN is actually operating or if an ECN-capable node has detected congestion in the forward direction.

1.1. Conventions used in this document

1.1.1. Terminology

DSCP: Differentiated Service Codepoint

ECN: Explicit Congestion Notification

IPPM: IP Performance Measurement

TWAMP: Two-Way Active Measurement Protocol

OWAMP: One-Way Active Measurement Protocol

1.1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in

[RFC2119].

2. TWAMP Extensions

TWAMP connection establishment follows the procedure defined in Section 3.1 of [RFC4656] and Section 3.1 of [RFC5357] where the Modes field been used to identify and select specific communication capabilities. At the same time the Modes field been recognized and used as extension mechanism [RFC6038]. The new feature requires new bit position to identify the ability of a Session-Reflector to return value of received DSCP and ECN values back to a Session-Sender, and to support the new Session-Reflector packet format in the TWAMP-Test protocol. See the Section 3 for details on the assigned value and bit position.

2.1. Setting Up Connection to Monitor DSCP and ECN

The Server sets DSCP and ECN Monitoring flag in Modes field of the Server Greeting message to indicate its capabilities and willingness to monitor them. If the Control-Client agrees to monitor DSCP and ECN on some or all test sessions invoked with this control connection, it MUST set the DSCP and ECN Monitoring flag in Modes field in the Setup Response message.

2.2. TWAMP-Test Extension

Monitoring of DSCP and ECN requires support by Session-Reflector and changes format of its test packet format both in unauthenticated, authenticated and encrypted modes. Monitoring of DSCP and ECN does not alter Session-Sender test packet format but certain considerations must be taken when and if this mode is accepted in combination with Symmetrical Size mode[RFC6038].

2.2.1. Session-Reflector Packet Format for DSCP and ECN Monitoring

When Session-Reflector supports DSCP and ECN Monitoring it MUST construct Sender DSCP and ECN (S-DSCP-ECN) field for each test packet it sends to Session-Sender according to the following procedure:

- first six bits MUST be copied Differentiated Service field from received Session-Sender test packet into Sender DSCP (S-DSCP) field;
- following two bits MUST be copied ECN field from received Session-Sender test packet into Sender ECN (S-ECN) field.

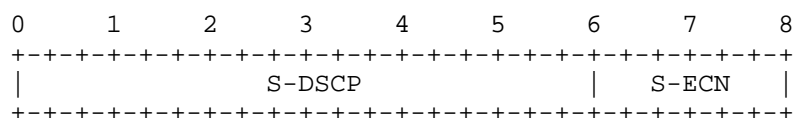


Figure 1: Sender DSCP and ECN field format

For unauthenticated mode:

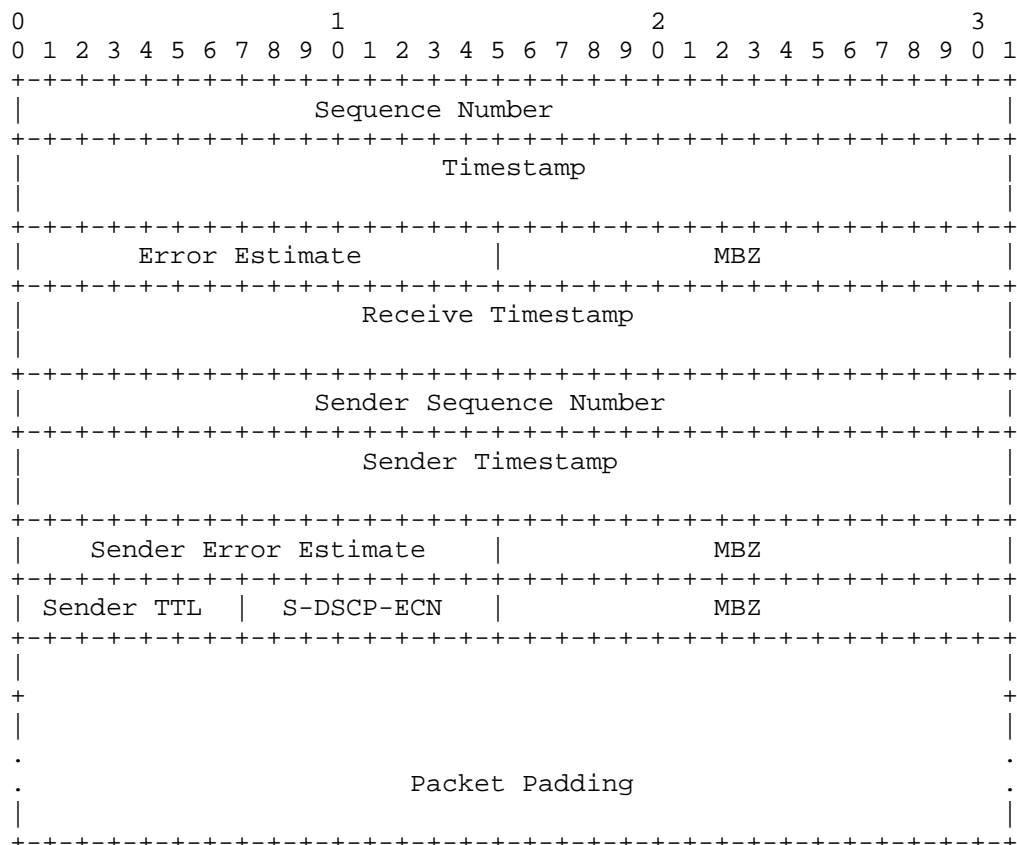
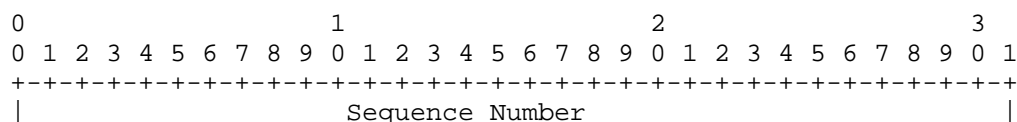


Figure 2: Session-Reflector test packet format with DSCP and ECN monitoring in unauthenticated mode

For authenticated and encrypted modes:



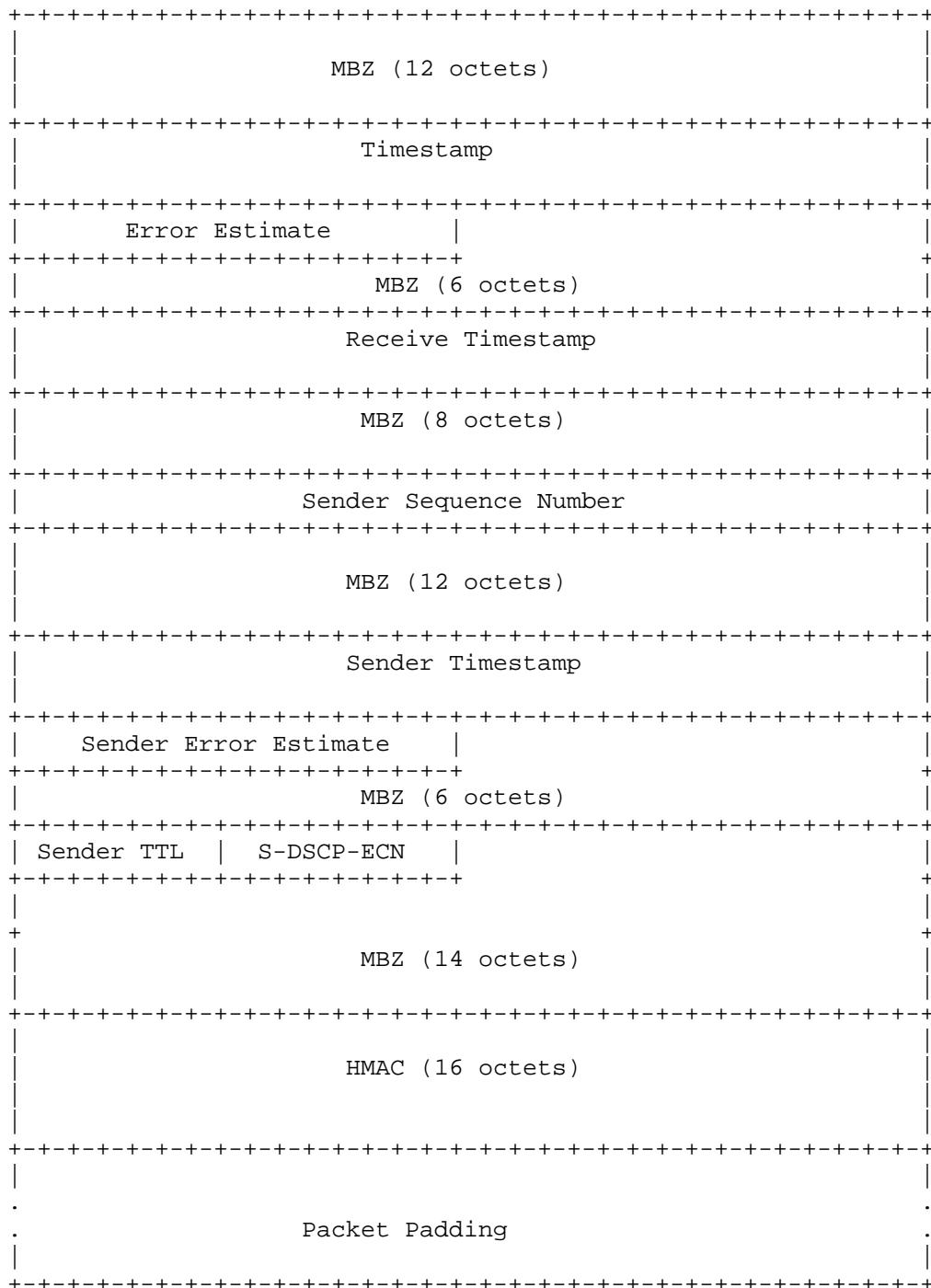


Figure 3: Session-Reflector test packet format with DSCP and ECN monitoring in authenticated or encrypted modes

The DSCP value is often copied into reflected test packets with current TWAMP implementations (with or without TWAMP-Control protocol). With DSCP and ECN Monitoring Extension Session-Reflector handles DSCP as following:

The Session-Reflector MUST extract the S-DSCP-ECN value from the DSCP and ECN values of received packets;

The Session-Reflector MUST transmit each reflected test packet with DSCP set to the negotiated/provisioned value;

If the negotiated/provisioned DSCP value is not known (e.g. TWAMP Light), the choice of the DSCP is implementation specific. For instance, Session-Reflector MAY copy the DSCP value from the received test packet and set it as DSCP in a reflected packet.

2.2.2. DSCP and ECN Monitoring with RFC 6038 extensions

[RFC6038] defined two extensions to TWAMP. First, to ensure that Session-Sender and Session-Reflector exchange TWAMP-Test packets of equal size. Second, to specify number of octets to be reflected by Session-Reflector. If DSCP and ECN monitoring and Symmetrical Size and/or Reflects Octets modes being negotiated between Server and Control-Client in Unauthenticated mode, then because Sender DSCP and Sender ECN increase size of unauthenticated Session-Reflector packet by 4 octets the Padding Length value SHOULD be ≥ 28 octets to allow for the truncation process that TWAMP recommends in Section 4.2. 1 of [RFC5357].

Value	Description	Semantics	Reference
X (proposed 128)	DSCP and ECN Monitoring Capability	bit position Y (proposed 7)	This document

Table 1: New Type-P Descriptor Monitoring Capability

4. Security Considerations

Monitoring of DSCP and ECN does not appear to introduce any additional security threat to hosts that communicate with TWAMP as defined in [RFC5357], and existing extensions [RFC6038]. The security considerations that apply to any active measurement of live networks are relevant here as well. See the Security Considerations sections in [RFC4656] and [RFC5357].

5. Acknowledgements

Authors greatly appreciate thorough review and thoughtful comments by Chritofer Flinta and Samita Chakrabarti.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.
- [RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the

Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, August 2009.

- [RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, October 2010.

6.2. Informative References

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Authors' Addresses

Jonas Hedin
Ericsson

Email: jonas.hedin@ericsson.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Steve Baillargeon
Ericsson

Email: steve.baillargeon@ericsson.com

IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: December 7, 2014

K. Pentikousis, Ed.
EICT
Y. Cui
E. Zhang
Huawei Technologies
June 5, 2014

Network Performance Measurement for IPsec
draft-ietf-ippm-ipsec-03

Abstract

The O/TWAMP security mechanism requires that both the client and server endpoints possess a shared secret. Since the currently-standardized O/TWAMP security mechanism only supports a pre-shared key mode, large scale deployment of O/TWAMP is hindered significantly. At the same time, recent trends point to wider IKEv2 deployment which, in turn, calls for mechanisms and methods that enable tunnel end-users, as well as operators, to measure one-way and two-way network performance in a standardized manner. This document discusses the use of keys derived from an IKEv2 SA as the shared key in O/TWAMP. If the shared key can be derived from the IKEv2 SA, O/TWAMP can support certificate-based key exchange, which would allow for more operational flexibility and efficiency. The key derivation presented in this document can also facilitate automatic key management.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 7, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. O/TWAMP Security	3
3.1. O/TWAMP-Control Security	4
3.2. O/TWAMP-Test Security	5
3.3. O/TWAMP Security Root	6
4. O/TWAMP for IPsec Networks	6
4.1. Shared Key Derivation	6
4.2. Server Greeting Message Update	7
4.3. Set-Up-Response Update	9
4.4. O/TWAMP over an IPsec tunnel	10
5. Security Considerations	10
6. IANA Considerations	10
7. Acknowledgments	10
8. References	11
8.1. Normative References	11
8.2. Informative References	11
Authors' Addresses	11

1. Introduction

The One-way Active Measurement Protocol (OWAMP) [RFC4656] and the Two-Way Active Measurement Protocol (TWAMP) [RFC5357] can be used to measure network performance parameters, such as latency, bandwidth, and packet loss by sending probe packets and monitoring their experience in the network. In order to guarantee the accuracy of network measurement results, security aspects must be considered. Otherwise, attacks may occur and the authenticity of the measurement results may be violated. For example, if no protection is provided, an adversary in the middle may modify packet timestamps, thus altering the measurement results.

The currently-standardized O/TWAMP security mechanism [RFC4656] [RFC5357] requires that endpoints (i.e. both the client and the server) possess a shared secret. In today's network deployments, however, the use of pre-shared keys is far from optimal. For example, in wireless infrastructure networks, certain network elements, which can be seen as the two endpoints from an O/TWAMP perspective, support certificate-based security. For instance, consider the case in which one wants to measure IP performance between an eNB and SeGW. Both eNB and SeGW are 3GPP LTE nodes and support certificate mode and IKEv2. Since the currently standardized O/TWAMP security mechanism only supports pre-shared key mode, large scale deployment of O/TWAMP is hindered significantly. Furthermore, deployment and management of "shared secrets" for massive equipment installation consumes a tremendous amount of effort and is prone to human error.

With IKEv2 widely used, employing keys derived from IKEv2 SA as shared key can be considered as a viable alternative. In mobile telecommunication networks, the deployment rate of IPsec exceeds 95% with respect to the LTE serving network. In older-technology cellular networks, such as UMTS and GSM, IPsec use penetration is lower, but still quite significant. If the shared key can be derived from the IKEv2 SA, O/TWAMP can support cert-based key exchange and make it more flexible in practice and more efficient. The use of IKEv2 also makes it easier to extend automatic key management. In general, O/TWAMP measurement packets can be transmitted inside the IPsec tunnel, as it occurs with typical user traffic, or transmitted outside the IPsec tunnel. This may depend on the operator's policy and is orthogonal to the mechanism described in this document.

The remainder of this document is organized as follows. Section 3 summarizes O/TWAMP protocol operation with respect to security. Section 4 presents a method of binding O/TWAMP and IKEv2 for network measurements between the client and the server which both support IKEv2. Finally, Section 5 discusses the security considerations arising from the proposed mechanisms.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. O/TWAMP Security

Security for O/TWAMP-Control and O/TWAMP-Test are briefly reviewed in the following subsections.

3.1. O/TWAMP-Control Security

O/TWAMP uses a simple cryptographic protocol which relies on

- o AES in Cipher Block Chaining (AES-CBC) for confidentiality
- o HMAC-SHA1 truncated to 128 bits for message authentication

Three modes of operation are supported in the OWAMP-Control protocol: unauthenticated, authenticated, and encrypted. In addition to these modes, the TWAMP-Control protocol also supports a mixed mode, i.e. the TWAMP-Control protocol operates in encrypted mode while TWAMP-Test protocol operates in unauthenticated mode. The authenticated, encrypted and mixed modes require that endpoints possess a shared secret, typically a passphrase. The secret key is derived from the passphrase using a password-based key derivation function PBKDF2 (PKCS#5) [RFC2898].

In the unauthenticated mode, the security parameters are left unused. In the authenticated, encrypted and mixed modes, the security parameters are negotiated during the control connection establishment.

Figure 1 illustrates the initiation stage of the O/TWAMP-Control protocol between a client and the server. In short, the client opens a TCP connection to the server in order to be able to send O/TWAMP-Control commands. The server responds with a Server Greeting, which contains the Modes, Challenge, Salt, Count, and MBZ fields (see Section 3.1 of [RFC4656]). If the client-preferred mode is available, the client responds with a Set-Up-Response message, wherein the selected Mode, as well as the KeyID, Token and Client IV are included. The Token is the concatenation of a 16-octet Challenge, a 16-octet AES Session-key used for encryption, and a 32-octet HMAC-SHA1 Session-key used for authentication. The Token is encrypted using AES-CBC.

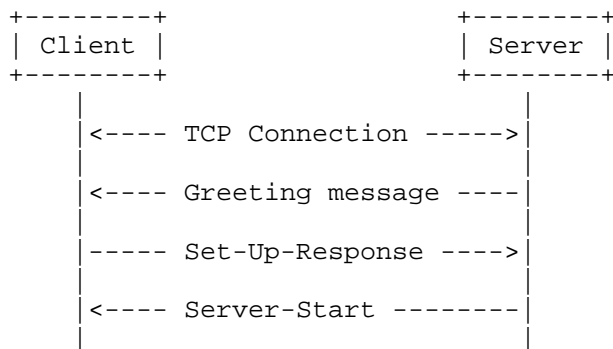


Figure 1: Initiation of O/TWAMP-Control

Encryption uses a key derived from the shared secret associated with KeyID. In the authenticated, encrypted and mixed modes, all further communication is encrypted using the AES Session-key and authenticated with the HMAC Session-key. After receiving Set-Up-Response the server responds with a Server-Start message containing Server-IV. The client encrypts everything it transmits through the just-established O/TWAMP-Control connection using stream encryption with Client-IV as the IV. Correspondingly, the server encrypts its side of the connection using Server-IV as the IV. The IVs themselves are transmitted in cleartext. Encryption starts with the block immediately following that containing the IV.

The AES Session-key and HMAC Session-key are generated randomly by the client. The HMAC Session-key is communicated along with the AES Session-key during O/TWAMP-Control connection setup. The HMAC Session-key is derived independently of the AES Session-key.

3.2. O/TWAMP-Test Security

The O/TWAMP-Test protocol runs over UDP, using the client and server IP and port numbers that were negotiated during the Request-Session exchange. O/TWAMP-Test has the same mode with O/TWAMP-Control and all O/TWAMP-Test sessions inherit the corresponding O/TWAMP-Control session mode except when operating in mixed mode.

The O/TWAMP-Test packet format is the same in authenticated and encrypted modes. The encryption and authentication operations are, however, different. Similarly with the respective O/TWAMP-Control session, each O/TWAMP-Test session has two keys: an AES Session-key and an HMAC Session-key. However, there is a difference in how the keys are obtained:

O/TWAMP-Control: the keys are generated by the client and communicated to the server during the control connection establishment with the Set-Up-Response message (as part of the Token).

O/TWAMP-Test: the keys are derived from the O/TWAMP-Control keys and the session identifier (SID), which serve as inputs of the key derivation function (KDF). The O/TWAMP-Test AES Session-key is generated using the O/TWAMP-Control AES Session-key, with the 16-octet session identifier (SID), for encrypting and decrypting the packets of the particular O/TWAMP-Test session. The O/TWAMP-Test HMAC Session-key is generated using the O/TWAMP-Control HMAC Session-key, with the 16-octet session identifier (SID), for authenticating the packets of the particular O/TWAMP-Test session.

3.3. O/TWAMP Security Root

As discussed above, the AES Session-key and HMAC Session-key used by the O/TWAMP-Test protocol are derived from the AES Session-key and HMAC Session-key which are used in O/TWAMP-Control protocol. The AES Session-key and HMAC Session-key used in the O/TWAMP-Control protocol are generated randomly by the client, and encrypted with the shared secret associated with KeyID. Therefore, the security root is the shared secret key. Thus, for large deployments, key provision and management may become overly complicated. Comparatively, a certificate-based approach using IKEv2 can automatically manage the security root and solve this problem, as we explain in Section 4.

4. O/TWAMP for IPsec Networks

This section presents a method of binding O/TWAMP and IKEv2 for network measurements between a client and a server which both support IPsec. In short, the shared key used for securing O/TWAMP traffic is derived using IKEv2 [RFC5996].

4.1. Shared Key Derivation

In the authenticated, encrypted and mixed modes, the shared secret key can be derived from the IKEv2 Security Association (SA). Note that we explicitly opt to derive the shared secret key from the IKEv2 SA, rather than the child SA, since the use case whereby an IKEv2 SA can be created without generating any child SA is possible [RFC6023].

If the shared secret key is derived from the IKEv2 SA, SKEYSEED must be generated first. SKEYSEED and its derivatives MUST be computed as per [RFC5996], where prf is a pseudorandom function:

$$\text{SKEYSEED} = \text{prf}(\text{Ni} \mid \text{Nr}, g^{\text{ir}})$$

Ni and Nr are, respectively, the initiator and responder nonces, which are negotiated during the initial exchange (see Section 1.2 of [RFC5996]). g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange and is represented as a string of octets.

The shared secret key MUST be generated as follows:

$$\text{Shared secret key} = \text{PRF}(\text{SKEYSEED}, \text{"IPPM"})$$

Wherein the string "IPPM" comprises four ASCII characters. It is recommended that the shared secret key is derived in the IPsec layer. This way, the IPsec keying material is not exposed to the O/TWAMP client. Note, however, that the interaction between the O/TWAMP and IPsec layers is host-internal and implementation-specific. Therefore, this is clearly outside the scope of this document, which focuses on the interaction between the O/TWAMP client and server. That said, one possible way could be the following: at the client side, the IPsec layer can perform a lookup in the Security Association Database (SAD) using the IP address of the server and thus match the corresponding IKEv2 SA. At the server side, the IPsec layer can look up the corresponding IKEv2 SA by using the SPIs sent by the client, and therefore extract the shared secret key. In case that both client and server do support IKEv2 but there is no current IKEv2 SA, two alternative ways could be considered. First, the O/TWAMP client initiates the establishment of the IKEv2 SA, logs this operation, and selects the mode which supports IKEv2. Alternatively, the O/TWAMP client does not initiate the establishment of the IKEv2 SA, logs an error for operational management purposes, and proceeds with the modes defined in [RFC4656][RFC5618]. Again, although both alternatives are feasible, they are in fact implementation-specific.

If rekeying for the IKEv2 SA or deletion of the IKEv2 SA occurs, the corresponding shared secret key generated from the SA can continue to be used until the lifetime of the shared secret key expires.

4.2. Server Greeting Message Update

To achieve a binding association between the key generated from IKEv2 and the O/TWAMP shared secret key, Server Greeting Message should be updated as in Figure 2.

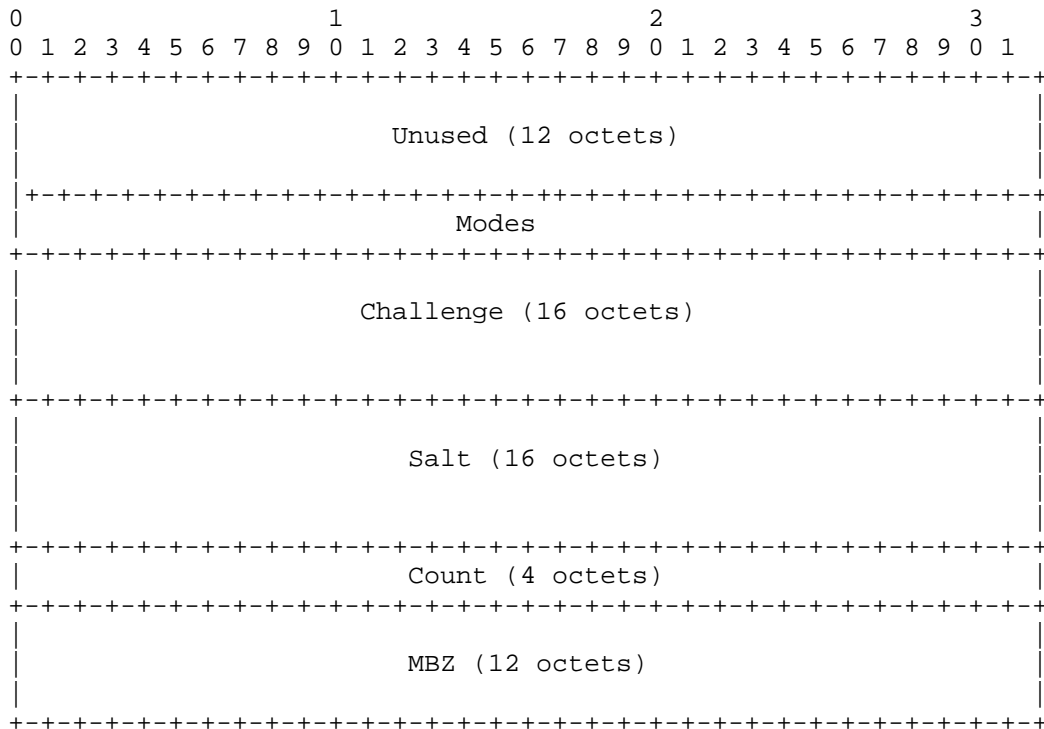


Figure 2: Server Greeting format

The Modes field in Figure 2 will need to allow for support of key derivation as discussed in Section 4.1. As such, the Modes value extension MUST be supported by implementations compatible with this document, indicating support for deriving shared key from IKEv2 SA. Three new Modes including authenticated mode over IKEv2(IANA.TBA.O/TWAMP.IKEAuth), encrypted mode over IKEv2(IANA.TBA.O/TWAMP.IKEEnc) and mixed mode over IKEv2(IANA.TBA.TWAMP.IKEMix) are proposed.

Authenticated mode over IKEv2 means that the client and server operate in authenticated mode with the shared secret key derived from IKEv2 SA. Encrypted mode over IKEv2 means that the client and server operate in encrypted mode with the shared secret key derived from IKEv2 SA. Mixed mode over IKEv2 means that the client and server operate in encrypted mode for the O/TWAMP-Control protocol while operating in unauthenticated mode for the O/TWAMP-Test protocol with shared secret key derived from IKEv2 SA.

The choice of this set of Modes values poses the least backwards compatibility problems to existing O/TWAMP clients. Robust client implementations of [RFC4656] would disregard the fact that the first

29 Modes bits in the Server Greeting is set. If the server supports other Modes, as one would assume, the client would then indicate any of the Modes defined in [RFC4656] and effectively indicate that it does not support key derivation from IKEv2. At this point, the Server would need to use the Modes defined in [RFC4656] only.

4.3. Set-Up-Response Update

The Set-Up-Response Message should be updated as in Figure 3.

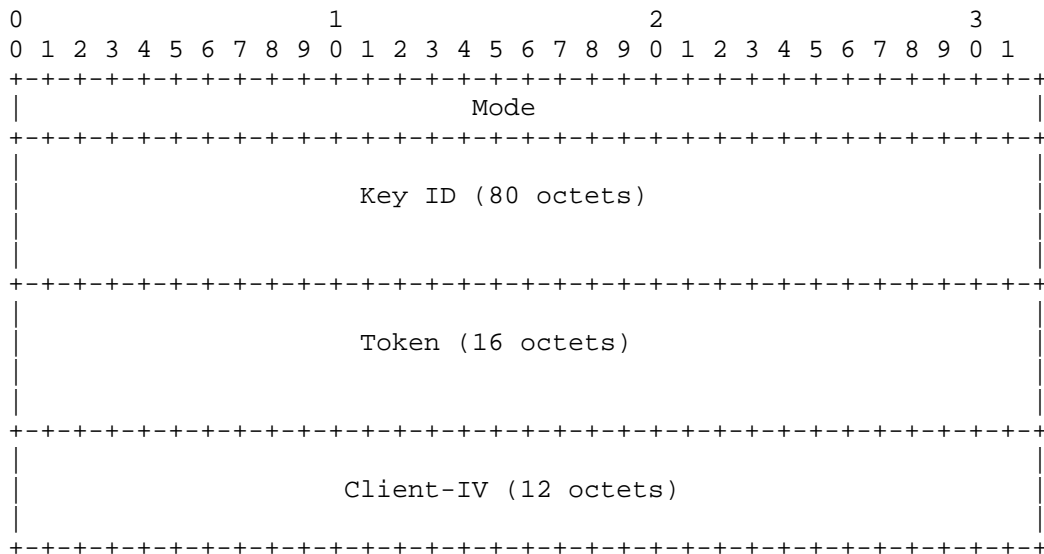


Figure 3: Set-Up-Response Message

The Security Parameter Index (SPI)(see [RFC4301] [RFC5996]) can uniquely identify the Security Association (SA). If the client supports the derivation of shared secret key from IKEv2 SA, it will choose the corresponding mode value and carry SPIi and SPIr in the Key ID field. SPIi and SPIr are included in the Key ID field of Set-Up-Response Message to indicate the IKEv2 SA from which the O/TWAMP shared secret key derived from. The length of SPI is 4 octets. Therefore, the first 4 octets of Key ID field carry SPIi and the second 4 octets carry SPIr. The remaining bits of the Key ID field are set to zero.

A O/TWAMP server which supports the specification of this document, can obtain the SPIi and SPIr from the first 8 octets and ignore the rest octets of the Key ID field. Then, the client and the server can derive the shared secret key based on the mode value and SPI. If the O/TWAMP server cannot find the IKEv2 SA corresponding to the SPIi and

SPIr received, it MUST log the event for operational management purposes. In addition, the O/TWAMP server SHOULD set the Accept field of the Server-Start message to the value 6 to indicate that server is not willing to conduct further transactions in this O/TWAMP-Control session since it can not find the corresponding IKEv2 SA.

4.4. O/TWAMP over an IPsec tunnel

IPsec AH [RFC4302] and ESP [RFC4303] provide confidentiality and data integrity to IP datagrams. Thus an IPsec tunnel can be used to provide the protection needed for O/TWAMP Control and Test packets, even if the peers choose the unauthenticated mode of operation. If the two endpoints are already connected through an IPsec tunnel it is RECOMMENDED that the O/TWAMP measurement packets are forwarded over the IPsec tunnel if the peers choose the unauthenticated mode in order to ensure authenticity and security.

5. Security Considerations

As the shared secret key is derived from the IKEv2 SA, the key derivation algorithm strength and limitations are as per [RFC5996]. The strength of a key derived from a Diffie-Hellman exchange using any of the groups defined here depends on the inherent strength of the group, the size of the exponent used, and the entropy provided by the random number generator employed. The strength of all keys and implementation vulnerabilities, particularly Denial of Service (DoS) attacks are as defined in [RFC5996].

As a more general note, the IPPM community may want to revisit the arguments listed in [RFC4656], Sec. 6.6. Other widely-used Internet security mechanisms, such as TLS and DTLS, may also be considered for future use over and above of what is already specified in [RFC4656] [RFC5357].

6. IANA Considerations

IANA will need to allocate additional values for the Modes options presented in this document.

7. Acknowledgments

Emily Bi contributed to an earlier version of this document.

We thank Eric Chen, Yakov Stein, Brian Trammell, and John Mattsson for their comments on this draft, and Al Morton for the discussion and pointers to related earlier work in IPPM WG.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.
- [RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, August 2009.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

8.2. Informative References

- [RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, October 2010.

Authors' Addresses

Kostas Pentikousis (editor)
EICT GmbH
EUREF-Campus Haus 13
Torgauer Strasse 12-15
10829 Berlin
Germany

Email: k.pentikousis@eict.de

Yang Cui
Huawei Technologies
Otemachi First Square 1-5-1 Otemachi
Chiyoda-ku, Tokyo 100-0004
Japan

Email: cuiyang@huawei.com

Emma Zhang
Huawei Technologies
Huawei Building, Q20, No.156, Rd. BeiQing
Haidian District , Beijing 100095
P. R. China

Email: emma.zhanglijia@huawei.com

IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: August 18, 2014

M. Mathis
Google, Inc
A. Morton
AT&T Labs
February 14, 2014

Model Based Bulk Performance Metrics
draft-ietf-ippm-model-based-metrics-02.txt

Abstract

We introduce a new class of model based metrics designed to determine if an end-to-end Internet path can meet predefined transport performance targets by applying a suite of IP diagnostic tests to successive subpaths. The subpath-at-a-time tests are designed to accurately detect if any subpath will prevent the full end-to-end path from meeting the specified target performance. Each IP diagnostic test consists of a precomputed traffic pattern and a statistical criteria for evaluating packet delivery.

The IP diagnostics tests are based on traffic patterns that are precomputed to mimic TCP or other transport protocol over a long path but are independent of the actual details of the subpath under test. Likewise the success criteria depends on the target performance and not the actual performance of the subpath. This makes the measurements open loop, eliminating nearly all of the difficulties encountered by traditional bulk transport metrics.

This document does not fully define diagnostic tests, but provides a framework for designing suites of diagnostics tests that are tailored the confirming the target performance.

By making the tests open loop, we eliminate standards congestion control equilibrium behavior, which otherwise causes every measured parameter to be sensitive to every component of the system. As an open loop test, various measurable properties become independent, and potentially subject to an algebra enabling several important new uses.

Interim DRAFT Formatted: Fri Feb 14 14:07:33 PST 2014

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering

Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 18, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
1.1. TODO	7
2. Terminology	7
3. New requirements relative to RFC 2330	10
4. Background	11
4.1. TCP properties	12
4.2. Diagnostic Approach	13
5. Common Models and Parameters	15
5.1. Target End-to-end parameters	15
5.2. Common Model Calculations	15
5.3. Parameter Derating	16
6. Common testing procedures	17
6.1. Traffic generating techniques	17
6.1.1. Paced transmission	17
6.1.2. Constant window pseudo CBR	18
6.1.3. Scanned window pseudo CBR	18
6.1.4. Concurrent or channelized testing	19
6.1.5. Intermittent Testing	19
6.1.6. Intermittent Scatter Testing	20
6.2. Interpreting the Results	20
6.2.1. Test outcomes	20
6.2.2. Statistical criteria for measuring run_length	22
6.2.2.1. Alternate criteria for measuring run_length	24
6.2.3. Reordering Tolerance	25
6.3. Test Qualifications	26
7. Diagnostic Tests	27
7.1. Basic Data Rate and Run Length Tests	27
7.1.1. Run Length at Paced Full Data Rate	27
7.1.2. Run Length at Full Data Windowed Rate	28
7.1.3. Background Run Length Tests	28
7.2. Standing Queue tests	28
7.2.1. Congestion Avoidance	29
7.2.2. Bufferbloat	30
7.2.3. Non excessive loss	30
7.2.4. Duplex Self Interference	30
7.3. Slowstart tests	30
7.3.1. Full Window slowstart test	31
7.3.2. Slowstart AQM test	31
7.4. Sender Rate Burst tests	31
7.5. Combined Tests	32
7.5.1. Sustained burst test	32
7.5.2. Live Streaming Media	33
8. Examples	34
8.1. Near serving HD streaming video	34
8.2. Far serving SD streaming video	34
8.3. Bulk delivery of remote scientific data	35

9. Validation	35
10. Acknowledgements	37
11. Informative References	37
Appendix A. Model Derivations	39
A.1. Queueless Reno	39
A.2. CUBIC	40
Appendix B. Complex Queueing	41
Appendix C. Version Control	42
Authors' Addresses	42

1. Introduction

Bulk performance metrics evaluate an Internet path's ability to carry bulk data. Model based bulk performance metrics rely on mathematical TCP models to design a targeted diagnostic suite (TDS) of IP performance tests which can be applied independently to each subpath of the full end-to-end path. These targeted diagnostic suites allow independent tests of subpaths to accurately detect if any subpath will prevent the full end-to-end path from delivering bulk data at the specified performance target, independent of the measurement vantage points or other details of the test procedures used for each measurement.

The end-to-end target performance is determined by the needs of the user or application, outside the scope of this document. For bulk data transport, the primary performance parameter of interest is the target data rate. However, since TCP's ability to compensate for less than ideal network conditions is fundamentally affected by the Round Trip Time (RTT) and the Maximum Transmission Unit (MTU) of the entire end-to-end path over which the data traverses, these parameters must also be specified in advance. They may reflect a specific real path through the Internet or an idealized path representing a typical user community. The target values for these three parameters, Data Rate, RTT and MTU, inform the mathematical models used to design the TDS.

Each IP diagnostic test in a TDS consists of a precomputed traffic pattern and statistical criteria for evaluating packet delivery.

Mathematical models are used to design traffic patterns that mimic TCP or other bulk transport protocol operating at the target data rate, MTU and RTT over a full range of conditions, including flows that are bursty at multiple time scales. The traffic patterns are computed in advance based on the three target parameters of the end-to-end path and independent of the properties of individual subpaths. As much as possible the measurement traffic is generated deterministically in ways that minimize the extent to which test methodology, measurement points, measurement vantage or path partitioning affect the details of the measurement traffic.

Mathematical models are also used to compute the bounds on the packet delivery statistics for acceptable IP performance. Since these statistics, such as packet loss, are typically aggregated from all subpaths of the end-to-end path, the end-to-end statistical bounds need to be apportioned as a separate bound for each subpath. Note that links that are expected to be bottlenecks are expected to contribute more packet loss and/or delay. In compensation, other links have to be constrained to contribute less packet loss and

delay. The criteria for passing each test of a TDS is an apportioned share of the total bound determined by the mathematical model from the end-to-end target performance.

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons including, the precomputed traffic pattern was not accurately generated, measurement results were not statistically significant, and others such as failing to meet some test preconditions.

This document describes a framework for deriving traffic patterns and delivery statistics for model based metrics. It does not fully specify any measurement techniques. Important details such as packet type-p selection, sampling techniques, vantage selection, etc. are not specified here. We imagine Fully Specified Targeted Diagnostic Suites (FSTDS), that define all of these details. We use TDS to refer to the subset of such a specification that is in scope for this document. A TDS includes the target parameters, documentation of the models and assumptions used to derive the diagnostic test parameters, specifications for the traffic and delivery statistics for the tests themselves, and a description of a test setup that can be used to validate the tests and models.

Section 2 defines terminology used throughout this document.

It has been difficult to develop Bulk Transport Capacity [RFC3148] metrics due to some overlooked requirements described in Section 3 and some intrinsic problems with using protocols for measurement, described in Section 4.

In Section 5 we describe the models and common parameters used to derive the targeted diagnostic suite. In Section 6 we describe common testing procedures. Each subpath is evaluated using suite of far simpler and more predictable diagnostic tests described in Section 7. In Section 8 we present three example TDS', one that might be representative of HD video, when served fairly close to the user, a second that might be representative of standard video, served from a greater distance, and a third that might be representative of high performance bulk data delivered over a transcontinental path.

There exists a small risk that model based metric itself might yield a false pass result, in the sense that every subpath of an end-to-end path passes every IP diagnostic test and yet a real application fails to attain the performance target over the end-to-end path. If this happens, then the validation procedure described in Section 9 needs to be used to prove and potentially revise the models.

Future documents will define model based metrics for other traffic

classes and application types, such as real time streaming media.

1.1. TODO

Please send comments on this draft to ippm@ietf.org. See <http://goo.gl/02tkD> for more information including: interim drafts, an up to date todo list and information on contributing.

Formatted: Fri Feb 14 14:07:33 PST 2014

2. Terminology

Terminology about paths, etc. See [RFC2330] and [I-D.morton-ippm-lmap-path].

[data] sender Host sending data and receiving ACKs.

[data] receiver Host receiving data and sending ACKs.

subpath A portion of the full path. Note that there is no requirement that subpaths be non-overlapping.

Measurement Point Measurement points as described in [I-D.morton-ippm-lmap-path].

test path A path between two measurement points that includes a subpath of the end-to-end path under test, and could include infrastructure between the measurement points and the subpath.

[Dominant] Bottleneck The Bottleneck that generally dominates traffic statistics for the entire path. It typically determines a flow's self clock timing, packet loss and ECN marking rate. See Section 4.1.

front path The subpath from the data sender to the dominant bottleneck.

back path The subpath from the dominant bottleneck to the receiver.

return path The path taken by the ACKs from the data receiver to the data sender.

cross traffic Other, potentially interfering, traffic competing for resources (network and/or queue capacity).

Properties determined by the end-to-end path and application. They are described in more detail in Section 5.1.

Application Data Rate General term for the data rate as seen by the application above the transport layer. This is the payload data rate, and excludes transport and lower level headers(TCP/IP or other protocols) and as well as retransmissions and other data that does not contribute to the total quantity of data delivered to the application.

Link Data Rate General term for the data rate as seen by the link or lower layers. The link data rate includes transport and IP headers, retransmits and other transport layer overhead. This document is agnostic as to whether the link data rate includes or excludes framing, MAC, or other lower layer overheads, except that they must be treated uniformly.

end-to-end target parameters: Application or transport performance goals for the end-to-end path. They include the target data rate, RTT and MTU described below.

Target Data Rate: The application data rate, typically the ultimate user's performance goal.

Target RTT (Round Trip Time): The baseline (minimum) RTT of the longest end-to-end path over which the application expects to meet the target performance. TCP and other transport protocol's ability to compensate for path problems is generally proportional to the number of round trips per second. The Target RTT determines both key parameters of the traffic patterns (e.g. burst sizes) and the thresholds on acceptable traffic statistics. The Target RTT must be specified considering authentic packets sizes: MTU sized packets on the forward path, ACK sized packets (typically the header_overhead) on the return path.

Target MTU (Maximum Transmission Unit): The maximum MTU supported by the end-to-end path the over which the application expects to meet the target performance. Assume 1500 Byte packet unless otherwise specified. If some subpath forces a smaller MTU, then it becomes the target MTU, and all model calculations and subpath tests must use the same smaller MTU.

Effective Bottleneck Data Rate: This is the bottleneck data rate inferred from the ACK stream, by looking at how much data the ACK stream reports delivered per unit time. If the path is thinning ACKs or batching packets the effective bottleneck rate can be much higher than the average link rate. See Section 4.1 and Appendix B for more details.

[sender | interface] rate: The burst data rate, constrained by the data sender's interfaces. Today 1 or 10 Gb/s are typical.

Header_overhead: The IP and TCP header sizes, which are the portion of each MTU not available for carrying application payload. Without loss of generality this is assumed to be the size for returning acknowledgements (ACKs). For TCP, the Maximum Segment Size (MSS) is the Target MTU minus the header_overhead.

Basic parameters common to models and subpath tests. They are described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and link IP performance (includes headers).

pipe size A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. This is the window size which is normally the onset of queueing.

target_pipe_size: The number of packets in flight (the window size) needed to exactly meet the target rate, with a single stream and no cross traffic for the specified application target data rate, RTT, and MTU. It is the amount of circulating data required to meet the target data rate, and implies the scale of the bursts that the network might experience.

run length A general term for the observed, measured, or specified number of packets that are (to be) delivered between losses or ECN marks. Nominally one over the loss or ECN marking probability, if there are independently and identically distributed.

target_run_length The target_run_length is an estimate of the minimum required headway between losses or ECN marks necessary to attain the target_data_rate over a path with the specified target_RTT and target_MTU, as computed by a mathematical model of TCP congestion control. A reference calculation is show in Section 5.2 and alternatives in Appendix A

Ancillary parameters used for some tests

derating: Under some conditions the standard models are too conservative. The modeling framework permits some latitude in relaxing or derating some test parameters as described in Section 5.3 in exchange for a more stringent TDS validation procedures, described in Section 9.

subpath_data_rate The maximum IP data rate supported by a subpath. This typically includes TCP/IP overhead, including headers, retransmits, etc.

test_path_RTT The RTT between two measurement points using appropriate data and ACK packet sizes.

test_path_pipe The amount of data necessary to fill a test path. Nominally the test path RTT times the subpath_data_rate (which should be part of the end-to-end subpath).

test_window The window necessary to meet the target_rate over a subpath. Typically $\text{test_window} = \text{target_data_rate} * \text{test_RTT} / (\text{target_MTU} - \text{header_overhead})$.

Tests can be classified into groups according to their applicability.

Capacity tests determine if a network subpath has sufficient capacity to deliver the target performance. As long as the test traffic is within the proper envelope for the target end-to-end performance, the average packet losses or ECN must be below the threshold computed by the model. As such, capacity tests reflect parameters that can transition from passing to failing as a consequence of cross traffic, additional presented load or the

actions of other network users. By definition, capacity tests also consume significant network resources (data capacity and/or buffer space), and the test schedules must be balanced by their cost.

Monitoring tests are designed to capture the most important aspects of a capacity test, but without presenting excessive ongoing load themselves. As such they may miss some details of the network's performance, but can serve as a useful reduced-cost proxy for a capacity test.

Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with TCP-style self clocked protocols and adaptive congestion control based on packet loss and ECN marks. These tests are likely to have complicated interactions with other traffic and under some conditions can be inversely sensitive to load. For example a test to verify that an AQM algorithm causes ECN marks or packet drops early enough to limit queue occupancy may experience a false pass result in the presence of bursty cross traffic. It is important that engineering tests be performed under a wide range of conditions, including both in situ and bench testing, and over a wide variety of load conditions. Ongoing monitoring is less likely to be useful for engineering tests, although sparse in situ testing might be appropriate.

General Terminology:

Targeted Diagnostic Test (TDS) A set of IP Diagnostics designed to determine if a subpath can sustain flows at a specific `target_data_rate` over a path that has a `target_RTT` using `target_MTU` sized packets.

Fully Specified Targeted Diagnostic Test A TDS together with additional specification such as "type-p", etc which are out of scope for this document, but need to be drawn from other standards documents.

apportioned To divide and allocate, as in budgeting packet loss rates across multiple subpaths to accumulate below a specified end-to-end loss rate.

open loop A control theory term used to describe a class of techniques where systems that exhibit circular dependencies can be analyzed by suppressing some of the dependences, such that the resulting dependency graph is acyclic.

3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirement that were not recognized at the time RFC 2330 was written [RFC2330]. These missing requirements may have significantly

contributed to policy difficulties in the IP measurement space. Some additional requirements are:

- o IP metrics must be actionable by the ISP - they have to be interpreted in terms of behaviors or properties at the IP or lower layers, that an ISP can test, repair and verify.
- o Metrics must be vantage point invariant over a significant range of measurement point choices, including off path measurement points. The only requirements on MP selection should be that the portion of the test path that is not under test is effectively ideal (or is non ideal in ways that can be calibrated out of the measurements) and the test RTT between the MPs is below some reasonable bound.
- o Metrics must be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It must be possible for different parties to make the same measurement and observe the same results. In particular it is specifically important that both a consumer (or their delegate) and ISP be able to perform the same measurement and get the same result.

NB: All of the metric requirements in RFC 2330 should be reviewed and potentially revised. If such a document is opened soon enough, this entire section should be dropped.

4. Background

At the time the IPPM WG was chartered, sound Bulk Transport Capacity measurement was known to be beyond our capabilities. By hindsight it is now clear why it is such a hard problem:

- o TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test.
- o Congestion control is an equilibrium process, such that transport protocols change the network (raise loss probability and/or RTT) to conform to their behavior.
- o TCP's ability to compensate for network flaws is directly proportional to the number of roundtrips per second (i.e. inversely proportional to the RTT). As a consequence a flawed link may pass a short RTT local test even though it fails when the path is extended by a perfect network to some larger RTT.
- o TCP has a meta Heisenberg problem - Measurement and cross traffic interact in unknown and ill defined ways. The situation is actually worse than the traditional physics problem where you can at least estimate the relative momentum of the measurement and measured particles. For network measurement you can not in general determine the relative "elasticity" of the measurement traffic and cross traffic, so you can not even gauge the relative magnitude of their effects on each other.

These properties are a consequence of the equilibrium behavior intrinsic to how all throughput optimizing protocols interact with the network. The protocols rely on control systems based on multiple network estimators to regulate the quantity of data sent into the network. The data in turn alters network and the properties observed by the estimators, such that there are circular dependencies between every component and every property. Since some of these estimators are non-linear, the entire system is nonlinear, and any change anywhere causes difficult to predict changes in every parameter.

Model Based Metrics overcome these problems by forcing the measurement system to be open loop: the delivery statistics (akin to the network estimators) do not affect the traffic. The traffic and traffic patterns (bursts) are computed on the basis of the target performance. In order for a network to pass, the resulting delivery statistics and corresponding network estimators have to be such that they would not cause the control systems slow the traffic below the target rate.

4.1. TCP properties

TCP and SCTP are self clocked protocols. The dominant steady state behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The receiver reports arriving data by returning ACKs to the data sender, the data sender typically responds by sending exactly the same quantity of data back into the network. The total quantity of data plus the data represented by ACKs circulating in the network is referred to as the window. The mandatory congestion control algorithms incrementally adjust the window by sending slightly more or less data in response to each ACK. The fundamentally important property of this systems is that it is entirely self clocked: The data transmissions are a reflection of the ACKs that were delivered by the network, the ACKs are a reflection of the data arriving from the network.

A number of phenomena can cause bursts of data, even in idealized networks that are modeled as simple queueing systems.

During slowstart the data rate is doubled on each RTT by sending twice as much data as was delivered to the receiver on the prior RTT. For slowstart to be able to fill such a network the network must be able to tolerate slowstart bursts up to the full pipe size inflated by the anticipated window reduction on the first loss or ECN mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for exactly one RTT in duration. This burst causes a queue which is exactly equal to the pipe size (i.e. the window is exactly twice the pipe size) so when the window is halved in response to the first

loss, the new window will be exactly the pipe size.

Note that if the bottleneck data rate is significantly slower than the rest of the path, the slowstart bursts will not cause significant queues anywhere else along the path; they primarily exercise the queue at the dominant bottleneck.

Other sources of bursts include application pauses and channel allocation mechanisms. Appendix B describes the treatment of channel allocation systems. If the application pauses (stops reading or writing data) for some fraction of one RTT, state-of-the-art TCP catches up to the earlier window size by sending a burst of data at the full sender interface rate. To fill such a network with a realistic application, the network has to be able to tolerate interface rate bursts from the data sender large enough to cover application pauses.

Although the interface rate bursts are typically smaller than last burst of a slowstart, they are at a higher data rate so they potentially exercise queues at arbitrary points along the front path from the data sender up to and including the queue at the dominant bottleneck. There is no model for how frequent or what sizes of sender rate bursts should be tolerated.

To verify that a path can meet a performance target, it is necessary to independently confirm that the path can tolerate bursts in the dimensions that can be caused by these mechanisms. Three cases are likely to be sufficient:

- o Slowstart bursts sufficient to get connections started properly.
- o Frequent sender interface rate bursts that are small enough where they can be assumed not to significantly affect delivery statistics. (Implicitly derated by selecting the burst size).
- o Infrequent sender interface rate full target_pipe_size bursts that do affect the delivery statistics. (Target_run_length is derated).

4.2. Diagnostic Approach

The MBM approach is to open loop TCP by precomputing traffic patterns that are typically generated by TCP operating at the given target parameters, and evaluating delivery statistics (packet loss, ECN marks and delay). In this approach the measurement software explicitly controls the data rate, transmission pattern or cwnd (TCP's primary congestion control state variables) to create repeatable traffic patterns that mimic TCP behavior but are independent of the actual behavior of the subpath under test. These patterns are manipulated to probe the network to verify that it can

deliver all of the traffic patterns that a transport protocol is likely to generate under normal operation at the target rate and RTT.

By opening the protocol control loops, we remove most sources of temporal and spatial correlation in the traffic delivery statistics, such that each subpath's contribution to the end-to-end statistics can be assumed to be independent and stationary (The delivery statistics depend on the fine structure of the data transmissions, but not on long time scale state imbedded in the sender, receiver or other network components.) Therefore each subpath's contribution to the end-to-end delivery statistics can be assumed to be independent, and spatial composition techniques such as [RFC5835] apply.

In typical networks, the dominant bottleneck contributes the majority of the packet loss and ECN marks. Often the rest of the path makes insignificant contribution to these properties. A TDS should apportion the end-to-end budget for the specified parameters (primarily packet loss and ECN marks) to each subpath or group of subpaths. For example the dominant bottleneck may be permitted to contribute 90% of the loss budget, while the rest of the path is only permitted to contribute 10%.

A TDS or FSTDS MUST apportion all relevant packet delivery statistics between different subpaths, such that the spatial composition of the metrics yields end-to-end statistics which are within the bounds determined by the models.

A network is expected to be able to sustain a Bulk TCP flow of a given data rate, MTU and RTT when the following conditions are met:

- o The raw link rate is higher than the target data rate.
- o The observed run length is larger than required by a suitable TCP performance model
- o There is sufficient buffering at the dominant bottleneck to absorb a slowstart rate burst large enough to get the flow out of slowstart at a suitable window size.
- o There is sufficient buffering in the front path to absorb and smooth sender interface rate bursts at all scales that are likely to be generated by the application, any channel arbitration in the ACK path or other mechanisms.
- o When there is a standing queue at a bottleneck for a shared media subpath, there are suitable bounds on how the data and ACKs interact, for example due to the channel arbitration mechanism.
- o When there is a slowly rising standing queue at the bottleneck the onset of packet loss has to be at an appropriate point (time or queue depth) and progressive. This typically requires some form of Automatic Queue Management [RFC2309].

We are developing a tool that can perform many of the tests described

here[MBMSource].

5. Common Models and Parameters

5.1. Target End-to-end parameters

The target end-to-end parameters are the target data rate, target RTT and target MTU as defined in Section 2. These parameters are determined by the needs of the application or the ultimate end user and the end-to-end Internet path over which the application is expected to operate. The target parameters are in units that make sense to upper layers: payload bytes delivered to the application, above TCP. They exclude overheads associated with TCP and IP headers, retransmits and other protocols (e.g. DNS).

Other end-to-end parameters defined in Section 2 include the effective bottleneck data rate, the sender interface data rate and the TCP/IP header sizes (overhead).

The target data rate must be smaller than all link data rates by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance fluctuations in the actual data rate, needed to meet the specified average rate. Specifying a target rate with insufficient headroom are likely to result in brittle measurements having little predictive value.

Note that the target parameters can be specified for a hypothetical path, for example to construct TDS designed for bench testing in the absence of a real application, or for a real physical test, for in situ testing of production infrastructure.

The number of concurrent connections is explicitly not a parameter to this model. If a subpath requires multiple connections in order to meet the specified performance, that must be stated explicitly and the procedure described in Section 6.1.4 applies.

5.2. Common Model Calculations

The end-to-end target parameters are used to derive the `target_pipe_size` and the reference `target_run_length`.

The `target_pipe_size`, is the average window size in packets needed to meet the target rate, for the specified target RTT and MTU. It is given by:

$$\text{target_pipe_size} = \text{target_rate} * \text{target_RTT} / (\text{target_MTU} - \text{header_overhead})$$

Target_run_length is an estimate of the minimum required headway between losses or ECN marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSM097], and by design is quite conservative. The alternate models described in Appendix A generally yield smaller run_lengths (higher loss rates), but may not apply in all situations. In any case alternate models should be compared to the reference target_run_length computed here.

Reference target_run_length is derived as follows: assume the subpath_data_rate is infinitesimally larger than the target_data_rate plus the required header_overhead. Then target_pipe_size also predicts the onset of queueing. A larger window will cause a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease congestion control [RFC5681] (but not Appropriate Byte Counting [RFC3465]) and the receiver is using standard delayed ACKs. Reno increases the window by one packet every pipe_size worth of ACKs. With delayed ACKs this takes 2 Round Trip Times per increase. To exactly fill the pipe losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_pipe_size otherwise the multiplicative window reduction triggered by the loss would cause the network to be underfilled. Following [MSM097] the number of packets between losses must be the area under the AIMD sawtooth. They must be no more frequent than every 1 in $((3/2)*target_pipe_size)*(2*target_pipe_size)$ packets, which simplifies to:

$$target_run_length = 3*(target_pipe_size^2)$$

Note that this calculation is very conservative and is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. If a less conservative model is used, a fully specified TDS or FSTDs MUST document the actual method for computing target_run_length along with the rationale for the underlying assumptions and the ratio of chosen target_run_length to the reference target_run_length calculated above.

These two parameters, target_pipe_size and target_run_length, directly imply most of the individual parameters for the tests in Section 7.

5.3. Parameter Derating

Since some aspects of the models are very conservative, this framework permits some latitude in derating test parameters. Rather than trying to formalize more complicated models we permit some test

parameters to be relaxed as long as they meet some additional procedural constraints:

- o The TDS or FSTDS MUST document and justify the actual method used compute the derated metric parameters.
- o The validation procedures described in Section 9 must be used to demonstrate the feasibility of meeting the performance targets with infrastructure that infinitesimally passes the derated tests.
- o The validation process itself must be documented in such a way that other researchers can duplicate the validation experiments.

Except as noted, all tests below assume no derating. Tests where there is not currently a well established model for the required parameters explicitly include derating as a way to indicate flexibility in the parameters.

6. Common testing procedures

6.1. Traffic generating techniques

6.1.1. Paced transmission

Paced (burst) transmissions: send bursts of data on a timer to meet a particular target rate and pattern. In all cases the specified data rate can either be the application or link rates. Header overheads must be included in the calculations as appropriate.

Paced single packets: Send individual packets at the specified rate or headway.

Burst: Send sender interface rate bursts on a timer. Specify any 3 of: average rate, packet size, burst size (number of packets) and burst headway (burst start to start). These bursts are typically sent as back-to-back packets at the testers interface rate.

Slowstart bursts: Send 4 packet sender interface rate bursts at an average data rate equal to twice effective bottleneck link rate (but not more than the sender interface rate). This corresponds to the average rate during a TCP slowstart when Appropriate Byte Counting [RFC3465] is present or delayed ack is disabled. Note that if the effective bottleneck link rate is more than half of the sender interface rate, slowstart bursts become sender interface rate bursts.

Repeated Slowstart bursts: Slowstart bursts are typically part of larger scale pattern of repeated bursts, such as sending `target_pipe_size` packets as slowstart bursts on a `target_RTT` headway (burst start to burst start). Such a stream has three different average rates, depending on the averaging interval. At the finest time scale the average rate is the same as the sender interface rate, at a medium scale the average rate is twice the effective bottleneck link rate and at the longest time scales the

average rate is equal to the target data rate.

Note that in conventional measurement theory exponential distributions are often used to eliminate many sorts of correlations. For the procedures above, the correlations are created by the network elements and accurately reflect their behavior. At some point in the future, it may be desirable to introduce noise sources into the above pacing models, but they are not warranted at this time.

6.1.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard protocol such as TCP with a fixed bound on the window size. The rate is only maintained in average over each RTT, and is subject to limitations of the transport protocol.

The bound on the window size is computed from the `target_data_rate` and the actual RTT of the test path.

If the transport protocol fails to maintain the test rate within prescribed limits the test would typically be considered inconclusive or failing, depending on what mechanism caused the reduced rate. See the discussion of test outcomes in Section 6.2.1.

6.1.3. Scanned window pseudo CBR

Same as the above, except the window is scanned across a range of sizes designed to include two key events, the onset of queueing and the onset of packet loss or ECN marks. The window is scanned by incrementing it by one packet for every $2 \times \text{target_pipe_size}$ delivered packets. This mimics the additive increase phase of standard congestion avoidance and normally separates the the window increases by approximately twice the `target_RTT`.

There are two versions of this test: one built by applying a window clamp to standard congestion control and one one built by stiffening a non-standard transport protocol. When standard congestion control is in effect, any losses or ECN marks cause the transport to revert to a window smaller than the clamp such that the scanning clamp loses control the window size. The NPAD pathdiag tool is an example of this class of algorithms [Pathdiag].

Alternatively a non-standard congestion control algorithm can respond to losses by transmitting extra data, such that it maintains the specified window size independent of losses or ECN marks. Such a stiffened transport explicitly violates mandatory Internet congestion control and is not suitable for in situ testing. It is only appropriate for engineering testing under laboratory conditions. The

Windowed Ping tools implemented such a test [WPING]. This tool has been updated and is under test.[mpingSource]

The test procedures in Section 7.2 describe how to the partition the scans into regions and how to interpret the results.

6.1.4. Concurrent or channelized testing

The procedures described in his document are only directly applicable to single stream performance measurement, e.g. one TCP connection. In an ideal world, we would disallow all performance claims based multiple concurrent streams but this is not practical due to at least two different issues. First, many very high rate link technologies are channelized and pin individual flows to specific channels to minimize reordering or other problems and second, TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the later problem is more deeply rooted.

All standard [RFC5681] and de facto standard congestion control algorithms [CUBIC] have scaling limits, in the sense that as a long fast network (LFN) with a fixed RTT and MTU gets faster, all congestion control algorithms get less accurate and as a consequence have difficulty filling the network [SLOWScaling]. These properties are a consequence of the original Reno AIMD congestion control design and the requirement in RFC 5681 that all transport protocols have uniform response to congestion.

There are a number of reasons to want to specify performance in term of multiple concurrent flows, however this approach is not recommended for data rates below several Mb/s, which can be attained with run lengths under 10000 packets. Since run length goes as the square of the data rate, at higher rates the run lengths can be unfeasibly large, and multiple connection might be the only feasible approach. For an example of this problem see Section 8.3.

If multiple connections are deemed necessary to meet aggregate performance targets then this MUST be stated both the design of the TDS and in any claims about network performance. The tests MUST be performed concurrently with the specified number of connections. For the the tests that using bursty traffic, the bursts should be synchronized across flows.

6.1.5. Intermittent Testing

Any test which does not depend on queueing (e.g. the CBR tests) or experiences periodic zero outstanding data during normal operation (e.g. between bursts for the various burst tests), can be formulated

as an intermittent test, to reduce the perceived impact on other traffic. The approach is to insert periodic pauses in the test at any point when there is no expected queue occupancy.

Intermittent testing can be used for ongoing monitoring for changes in subpath quality with minimal disruption users. However it is not suitable in environments where there are reactive links[REACTIVE].

6.1.6. Intermittent Scatter Testing

Intermittent scatter testing is a technique for non-disruptively evaluating the front path from a sender to a subscriber aggregation point within an ISP at full load by intermittently testing across a pool of subscriber access links, such that each subscriber sees tolerable test traffic loads. The load on the front path should be limited to be no more than that which would be caused by a single test to an known to otherwise be idle subscriber. This test in aggregate mimics a full load test from a content provider to the aggregation point.

Intermittent scatter testing can be used to reduce the measurement noise introduced by unknown traffic on customer access links.

6.2. Interpreting the Results

6.2.1. Test outcomes

To perform an exhaustive test of an end-to-end network path, each test of the TDS is applied to each subpath of an end-to-end path. If any subpath fails any test then an application running over the end-to-end path can also be expected to fail to attain the target performance under some conditions.

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons. Proper instrumentation and treatment of inclusive outcomes is critical to the accuracy and robustness of Model Based Metrics. Tests can be inconclusive if the precomputed traffic pattern was not accurately generated; the measurement results were not statistically significant; and others causes such as failing to meet some required preconditions for the test.

For example consider a test that implements Constant Window Pseudo CBR (Section 6.1.2) by adding rate controls and detailed traffic instrumentation to TCP (e.g. [RFC4898]). TCP includes built in control systems which might interfere with the sending data rate. If such a test meets the the run length specification while failing to attain the specified data rate it must be treated as an inconclusive

result, because we can not a priori determine if the reduced data rate was caused by a TCP problem or a network problem, or if the reduced data rate had a material effect on the run length measurement itself.

Note that for load tests such as this example, an observed run length that is too small can be considered to have failed the test because it doesn't really matter that the test didn't attain the required data rate.

The really important new properties of MBM, such as vantage independence, are a direct consequence of opening the control loops in the protocols, such that the test traffic does not depend on network conditions or traffic received. Any mechanism that introduces feedback between the traffic measurements and the traffic generation is at risk of introducing nonlinearities that spoil these properties. Any exceptional event that indicates that such feedback has happened should cause the test to be considered inconclusive.

One way to view inconclusive tests is that they reflect situations where a test outcome is ambiguous between limitations of the network and some unknown limitation of the diagnostic test itself, which was presumably caused by some uncontrolled feedback from the network.

Note that procedures that attempt to sweep the target parameter space to find the bounds on some parameter (for example to find the highest data rate for a subpath) are likely to break the location independent properties of Model Based Metrics, because the boundary between passing and inconclusive is sensitive to the RTT because TCP's ability to compensate for problems scales with the number of round trips per second. Repeating the same procedure from another vantage point with a different RTT is likely get a different result, because TCP will get lower performance on the path with the longer RTT.

One of the goals for evolving TDS designs will be to keep sharpening distinction between inconclusive, passing and failing tests. The criteria for for passing, failing and inclusive tests MUST be explicitly stated for every test in the TDS or FSTDs.

One of the goals of evolving the testing process, procedures tools and measurement point selection should be to minimize the number of inconclusive tests.

It may be useful to keep raw data delivery statistics for deeper study of the behavior of the network path and to measure the tools. This can help to drive tool evolution. Under some conditions it might be possible to reevaluate the raw data for satisfying alternate performance targets. However such procedures are likely to introduce

sampling bias and other implicit feedback which can cause false results and exhibit MP vantage sensitivity.

6.2.2. Statistical criteria for measuring run_length

When evaluating the observed run_length, we need to determine appropriate packet stream sizes and acceptable error levels for efficient measurement. In practice, can we compare the empirically estimated packet loss and ECN marking probabilities with the targets as the sample size grows? How large a sample is needed to say that the measurements of packet transfer indicate a particular run length is present?

The generalized measurement can be described as recursive testing: send packets (individually or in patterns) and observe the packet delivery performance (loss ratio or other metric, any marking we define).

As each packet is sent and measured, we have an ongoing estimate of the performance in terms of the ratio of packet loss or ECN mark to total packets (i.e. an empirical probability). We continue to send until conditions support a conclusion or a maximum sending limit has been reached.

We have a target_mark_probability, 1 mark per target_run_length, where a "mark" is defined as a lost packet, a packet with ECN mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length =
 $3 * (\text{target_pipe_size})^2$ packets

and we can stop sending packets if on-going measurements support accepting H0 with the specified Type I error = alpha (= 0.05 for example).

We also have an alternative Hypothesis to evaluate: if performance is significantly lower than the target_mark_probability. Based on analysis of typical values and practical limits on measurement duration, we choose four times the H0 probability:

H1: one or more marks in (target_run_length/4) packets

and we can stop sending packets if measurements support rejecting H0 with the specified Type II error = beta (= 0.05 for example), thus preferring the alternate hypothesis H1.

H0 and H1 constitute the Success and Failure outcomes described elsewhere in the memo, and while the ongoing measurements do not

support either hypothesis the current status of measurements is inconclusive.

The problem above is formulated to match the Sequential Probability Ratio Test (SPRT) [StatQC]. Note that as originally framed the events under consideration were all manufacturing defects. In networking, ECN marks and lost packets are not defects but signals, indicating that the transport protocol should slow down.

The Sequential Probability Ratio Test also starts with a pair of hypothesis specified as above:

H0: p_0 = one defect in target_run_length

H1: p_1 = one defect in target_run_length/4

As packets are sent and measurements collected, the tester evaluates the cumulative defect count against two boundaries representing H0 Acceptance or Rejection (and acceptance of H1):

Acceptance line: $X_a = -h_1 + sn$

Rejection line: $X_r = h_2 + sn$

where n increases linearly for each packet sent and

$h_1 = \{ \log((1-\alpha)/\beta) \} / k$

$h_2 = \{ \log((1-\beta)/\alpha) \} / k$

$k = \log\{ (p_1(1-p_0)) / (p_0(1-p_1)) \}$

$s = [\log\{ (1-p_0)/(1-p_1) \}] / k$

for p_0 and p_1 as defined in the null and alternative Hypotheses statements above, and α and β as the Type I and Type II error.

The SPRT specifies simple stopping rules:

- o $X_a < \text{defect_count}(n) < X_b$: continue testing
- o $\text{defect_count}(n) \leq X_a$: Accept H0
- o $\text{defect_count}(n) \geq X_b$: Accept H1

The calculations above are implemented in the R-tool for Statistical Analysis [Rtool] , in the add-on package for Cross-Validation via Sequential Testing (CVST) [CVST] .

Using the equations above, we can calculate the minimum number of packets (n) needed to accept H0 when x defects are observed. For example, when $x = 0$:

$X_a = 0 = -h_1 + sn$

and $n = h1 / s$

6.2.2.1. Alternate criteria for measuring run_length

An alternate calculation, contributed by Alex Gilgur (Google).

The probability of failure within an interval whose length is target_run_length is given by an exponential distribution with rate = $1 / \text{target_run_length}$ (a memoryless process). The implication of this is that it will be different, depending on the total count of packets that have been through the pipe, the formula being:

$$P(t1 < T < t2) = R(t1) - R(t2),$$

where

T = number of packets at which a failure will occur with probability P;

t = number of packets:

t1 = number of packets (e.g., when failure last occurred)

t2 = t1 + target_run_length

R = failure rate:

$R(t1) = \exp(-t1/\text{target_run_length})$

$R(t2) = \exp(-t2/\text{target_run_length})$

The algorithm:

```
initialize the packet.counter = 0
initialize the failed.packet.counter = 0
start the loop
if paket_response = ACK:
increment the packet.counter
else:
### The packet failed
increment the packet.counter
increment the failed.packet.counter

P_fail_observed = failed.packet.counter/packet.counter

upper_bound = packet.counter + target.run.length / 2
lower_bound = packet.counter - target.run.length / 2

R1 = exp( -upper_bound / target.run.length)
R0 = R(max(0, lower_bound)/ target.run.length)

P_fail_predicted = R1-R0
Compare P_fail_observed vs. P_fail_predicted
end-if
continue the loop
```

This algorithm allows accurate comparison of the observed failure probability with the corresponding values predicted based on a fixed target_failure_rate, which is equal to $1.0 / \text{target_run_length}$.

6.2.3. Reordering Tolerance

All tests must be instrumented for packet level reordering [RFC4737]. However, there is no consensus for how much reordering should be acceptable. Over the last two decades the general trend has been to make protocols and applications more tolerant to reordering, in response to the gradual increase in reordering in the network. This increase has been due to the gradual deployment of parallelism in the network, as a consequence of such technologies as multithreaded route lookups and Equal Cost Multipath (ECMP) routing. These techniques to increase network parallelism are critical to enabling overall Internet growth to exceed Moore's Law.

Section 5 of [RFC4737] proposed a metric that may be sufficient to designate isolated reordered packets as effectively lost, because TCP's retransmission response would be the same.

TCP should be able to adapt to reordering as long as the reordering extent is no more than the maximum of one half window or 1 mS, whichever is larger. Note that there is a fundamental tradeoff between tolerance to reordering and how quickly algorithms such as fast retransmit can repair losses. Within this limit on reorder extent, there should be no bound on reordering density.

NB: Traditional TCP implementations were not compatible with this metric, however newer implementations still need to be evaluated

Parameters:

Reordering displacement: the maximum of one half of target_pipe_size or 1 mS.

6.3. Test Qualifications

This entire section need to be completely overhauled. @@@@ It might be summarized as "needs to be specified in a FSTDs".

Send pre-load traffic as needed to activate radios with a sleep mode, or other "reactive network" elements (term defined in [draft-morton-ippm-2330-update-01]).

In general failing to accurately generate the test traffic has to be treated as an inconclusive test, since it must be presumed that the error in traffic generation might have affected the test outcome. To the extent that the network itself had an effect on the the traffic generation (e.g. in the standing queue tests) the possibility exists that allowing too large of error margin in the traffic generation might introduce feedback loops that comprise the vantage independents properties of these tests.

The proper treatment of cross traffic is different for different subpaths. In general when testing infrastructure which is associated with only one subscriber, the test should be treated as inconclusive it that subscriber is active on the network. However, for shared infrastructure managed by an ISP, the question at hand is likely to be testing if ISP has sufficient total capacity. In such cases the presence of cross traffic due to other subscribers is explicitly part of the network conditions and its effects are explicitly part of the test.

These two cases do not cover all subpaths. For example, WiFi which itself shares unmanaged channel space with other devices is unlikely to be unsuitable for any prescriptive measurement.

Note that canceling tests due to load on subscriber lines may introduce sampling bias for testing other parts of the

infrastructure. For this reason tests that are scheduled but not run due to load should be treated as a special case of "inconclusive".

7. Diagnostic Tests

The diagnostic tests below are organized by traffic pattern: basic data rate and run length, standing queues, slowstart bursts, and sender rate bursts. We also introduce some combined tests which are more efficient the expense of conflating the signatures of different failures.

7.1. Basic Data Rate and Run Length Tests

We propose several versions of the basic data rate and run length test. All measure the number of packets delivered between losses or ECN marks, using a data stream that is rate controlled at or below the `target_data_rate`.

The tests below differ in how the data rate is controlled. The data can be paced on a timer, or window controlled at full target data rate. The first two tests implicitly confirm that `sub_path` has sufficient raw capacity to carry the `target_data_rate`. They are recommend for relatively infrequent testing, such as an installation or auditing process. The third, background run length, is a low rate test designed for ongoing monitoring for changes in subpath quality.

All rely on the receiver accumulating packet delivery statistics as described in Section 6.2.2 to score the outcome:

Pass: it is statistically significant that the observed run length is larger than the `target_run_length`.

Fail: it is statistically significant that the observed run length is smaller than the `target_run_length`.

A test is considered to be inconclusive if it failed to meet the data rate as specified below, meet the qualifications defined in Section 6.3 or neither run length statistical hypothesis was confirmed in the allotted test duration.

7.1.1. Run Length at Paced Full Data Rate

Confirm that the observed run length is at least the `target_run_length` while relying on timer to send data at the `target_rate` using the procedure described in in Section 6.1.1 with a burst size of 1 (single packets).

The test is considered to be inconclusive if the packet transmission can not be accurately controlled for any reason.

7.1.2. Run Length at Full Data Windowed Rate

Confirm that the observed run length is at least the `target_run_length` while sending at an average rate equal to the `target_data_rate`, by controlling (or clamping) the window size of a conventional transport protocol to a fixed value computed from the properties of the test path, typically $\text{test_window} = \text{target_data_rate} * \text{test_RTT} / \text{target_MTU}$.

Since losses and ECN marks generally cause transport protocols to at least temporarily reduce their data rates, this test is expected to be less precise about controlling its data rate. It should not be considered inconclusive as long as at least some of the round trips reached the full `target_data_rate`, without incurring losses. To pass this test the network MUST deliver `target_pipe_size` packets in `target_RTT` time without any losses or ECN marks at least once per two `target_pipe_size` round trips, in addition to meeting the run length statistical test.

7.1.3. Background Run Length Tests

The background run length is a low rate version of the target rate test above, designed for ongoing lightweight monitoring for changes in the observed subpath run length without disrupting users. It should be used in conjunction with one of the above full rate tests because it does not confirm that the subpath can support raw data rate.

Existing loss metrics such as [RFC6673] might be appropriate for measuring background run length.

7.2. Standing Queue tests

These test confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of queueing. Well behaved generally means lossless for transient queues, but once the queue has been sustained for a sufficient period of time (or reaches a sufficient queue depth) there should be a small number of losses to signal to the transport protocol that it should reduce its window. Losses that are too early can prevent the transport from averaging at the `target_data_rate`. Losses that are too late indicate that the queue might be subject to bufferbloat [Bufferbloat] and inflict excess queuing delays on all flows sharing the bottleneck queue. Excess losses make loss recovery problematic for the transport protocol. Non-linear or erratic RTT fluctuations

suggest poor interactions between the channel acquisition systems and the transport self clock. All of the tests in this section use the same basic scanning algorithm but score the link on the basis of how well it avoids each of these problems.

For some technologies the data might not be subject to increasing delays, in which case the data rate will vary with the window size all the way up to the onset of losses or ECN marks. For these technologies, the discussion of queueing does not apply, but it is still required that the onset of losses (or ECN marks) be at an appropriate point and progressive.

Use the procedure in Section 6.1.3 to sweep the window across the onset of queueing and the onset of loss. The tests below all assume that the scan emulates standard additive increase and delayed ACK by incrementing the window by one packet for every $2 \times \text{target_pipe_size}$ packets delivered. A scan can be divided into three regions: below the onset of queueing, a standing queue, and at or beyond the onset of loss.

Below the onset of queueing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the link rate, the data rate becomes fairly constant, and the RTT increases in proportion to the window size. The precise transition from one region to the other can be identified by the maximum network power, defined to be the ratio data rate over the RTT[POWER].

For technologies that do not have conventional queues, start the scan at a window equal to the test_window, i.e. starting at the target rate, instead of the power point.

If there is random background loss (e.g. bit errors, etc), precise determination of the onset of packet loss may require multiple scans. Above the onset of loss, all transport protocols are expected to experience periodic losses. For the stiffened transport case they will be determined by the AQM algorithm in the network or the details of how the window increase function responds to loss. For the standard transport case the details of periodic losses are typically dominated by the behavior of the transport protocol itself.

7.2.1. Congestion Avoidance

A link passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the power point (or test_window) and the first loss or ECN mark. If this test is implemented using a standards congestion control algorithm with a clamp, it can be used in situ in the production internet as a

capacity test. For an example of such a test see [NPAD].

7.2.2. Bufferbloat

This test confirms that there is some mechanism to limit buffer occupancy (e.g. that prevents bufferbloat). Note that this is not strictly a requirement for single stream bulk performance, however if there is no mechanism to limit buffer occupancy then a single stream with sufficient data to deliver is likely to cause the problems described in [RFC2309] and [Bufferbloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the link unusable for other flows and applications.

Pass if the onset of loss is before a standing queue has introduced more delay than twice `target_RTT`, or other well defined limit. Note that there is not yet a model for how much standing queue is acceptable. The factor of two chosen here reflects a rule of thumb. Note that in conjunction with the previous test, this test implies that the first loss should occur at a queueing delay which is between one and two times the `target_RTT`.

7.2.3. Non excessive loss

This test confirm that the onset of loss is not excessive. Pass if losses are bound by the the fluctuations in the cross traffic, such that transient load (bursts) do not cause dips in aggregate raw throughput. e.g. pass as long as the losses are no more bursty than are expected from a simple drop tail queue. Although this test could be made more precise it is really included here for pedantic completeness.

7.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path. Fail if the RTT rises by more than some fixed bound above the expected queueing time computed from from the excess window divided by the link data rate.

7.3. Slowstart tests

These tests mimic slowstart: data is sent at twice the effective bottleneck rate to exercise the queue at the dominant bottleneck.

They are deemed inconclusive if the elapsed time to send the data burst is not less than half of the time to receive the ACKs. (i.e. sending data too fast is ok, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). Space the bursts such that the average data rate is

equal to the target_data_rate.

7.3.1. Full Window slowstart test

This is a capacity test to confirm that slowstart is not likely to exit prematurely. Send slowstart bursts that are target_pipe_size total packets.

Accumulate packet delivery statistics as described in Section 6.2.2 to score the outcome. Pass if it is statistically significant that the observed run length is larger than the target_run_length. Fail if it is statistically significant that the observed run length is smaller than the target_run_length.

Note that these are the same parameters as the Sender Full Window burst test, except the burst rate is at slowstart rate, rather than sender interface rate.

7.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at slowstart_rate), until the first loss, stop, allow the network to drain and repeat, gathering statistics on the last packet delivered before the loss, the loss pattern, maximum observed RTT and window size. Justify the results. There is not currently sufficient theory justifying requiring any particular result, however design decisions that affect the outcome of this tests also affect how the network balances between long and short flows (the "mice and elephants" problem).

This is an engineering test: It would be best performed on a quiescent network or testbed, since cross traffic has the potential to change the results.

7.4. Sender Rate Burst tests

These tests determine how well the network can deliver bursts sent at sender's interface rate. Note that this test most heavily exercises the front path, and is likely to include infrastructure may be out of scope for a subscriber ISP.

Also, there are a several details that are not precisely defined. For starters there is not a standard server interface rate. 1 Gb/s and 10 Gb/s are very common today, but higher rates will become cost effective and can be expected to be dominant some time in the future.

Current standards permit TCP to send a full window bursts following an application pause. Congestion Window Validation [RFC2861], is not required, but even if was it does not take effect until an

application pause is longer than an RTT. Since this is standard behavior, it is desirable that the network be able to deliver such bursts, otherwise application pauses will cause unwarranted losses.

It is also understood in the application and serving community that interface rate bursts have a cost to the network that has to be balanced against other costs in the servers themselves. For example TCP Segmentation Offload [TSO] reduces server CPU in exchange for larger network bursts, which increase the stress on network buffer memory.

There is not yet theory to unify these costs or to provide a framework for trying to optimize global efficiency. We do not yet have a model for how much the network should tolerate server rate bursts. Some bursts must be tolerated by the network, but it is probably unreasonable to expect the network to be able to efficiently deliver all data as a series of bursts.

For this reason, this is the only test for which we explicitly encourage derating. A TDS should include a table of pairs of derating parameters: what burst size to use as a fraction of the target_pipe_size, and how much each burst size is permitted to reduce the run length, relative to the target_run_length.

7.5. Combined Tests

These tests are more efficient from a deployment/operational perspective, but may not be possible to diagnose if they fail.

7.5.1. Sustained burst test

Send target_pipe_size*derate sender interface rate bursts every target_RTT*derate, for derate between 0 and 1. Verify that the observed run length meets target_run_length. Key observations:

- o This test is subpath RTT invariant, as long as the tester can generate the required pattern.
- o The subpath under test is expected to go idle for some fraction of the time: (subpath_data_rate-target_rate)/subpath_data_rate. Failing to do so suggests a problem with the procedure and an inconclusive test result.
- o This test is more strenuous than the slowstart tests: they are not needed if the link passes this test with derate=1.
- o A link that passes this test is likely to be able to sustain higher rates (close to subpath_data_rate) for paths with RTTs smaller than the target_RTT. Offsetting this performance underestimation is part of the rationale behind permitting derating in general.

- o This test can be implemented with standard instrumented TCP[RFC4898], using a specialized measurement application at one end and a minimal service at the other end [RFC 863, RFC 864]. It may require tweaks to the TCP implementation. [MBMSource]
- o This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern NIC hardware.
- o This test is not totally sufficient: the standing window engineering tests are also needed to be sure that the link is well behaved at and beyond the onset of congestion.
- o This one test can be proven to be the one capacity test to supplant them all.

7.5.2. Live Streaming Media

Model Based Metrics can be implemented as a side effect of serving any non-throughput maximizing traffic*, such as streaming media, with some additional controls and instrumentation in the servers. The essential requirement is that the traffic be constrained such that even with arbitrary application pauses, bursts and data rate fluctuations, the traffic stays within the envelope defined by the individual tests described above, for a specific TDS.

If the `serving_data_rate` is less than or equal to the `target_data_rate` and the `serving_RTT` (the RTT between the sender and client) is less than the `target_RTT`, this constraint is most easily implemented by clamping the transport window size to:

```
serving_window_clamp=target_data_rate*serving_RTT/  
(target_MTU-header_overhead)
```

The `serving_window_clamp` will limit the both the serving data rate and burst sizes to be no larger than the procedures in Section 7.1.2 and Section 7.4 or Section 7.5.1. Since the `serving_RTT` is smaller than the `target_RTT`, the worst case bursts that might be generated under these conditions will be smaller than called for by Section 7.4 and the sender rate burst sizes are implicitly derated by the `serving_window_clamp` divided by the `target_pipe_size` at the very least. (The traffic might be smoother than specified by the sender interface rate bursts test.)

Note that if the application tolerates fluctuations in its actual data rate (say by use of a playout buffer) it is important that the `target_data_rate` be above the actual average rate needed by the application so it can recover after transient pauses caused by congestion or the application itself.

Alternatively the sender data rate and bursts might be explicitly controlled by a host shaper or pacing at the sender. This would

provide better control and work for serving_RTTs that are larger than the target_RTT, but it is substantially more complicated to implement. With this technique, any traffic might be used for measurement.

* Note that this technique might be applied to any content, if users are willing to tolerate reduced data rate to inhibit TCP equilibrium behavior.

8. Examples

In this section we present TDS for a couple of performance specifications.

Tentatively: 5 Mb/s*50 ms, 1 Mb/s*50ms, 250kbp*100mS

8.1. Near serving HD streaming video

Today the best quality HD video requires slightly less than 5 Mb/s [HDvideo]. Since it is desirable to serve such content locally, we assume that the content will be within 50 mS, which is enough to cover continental Europe or either US coast from a single site.

5 Mb/s over a 50 ms path

End to End Parameter	Value	units
target_rate	5	Mb/s
target_RTT	50	ms
target_MTU	1500	bytes
target_pipe_size	22	packets
target_run_length	1452	packets

Table 1

This example uses the most conservative TCP model and no derating.

8.2. Far serving SD streaming video

Standard Quality video typically fits in 1 Mb/s [SDvideo]. This can be reasonably delivered via longer paths with larger. We assume 100mS.

1 Mb/s over a 100 ms path

End to End Parameter	Value	units
target_rate	1	Mb/s
target_RTT	100	ms
target_MTU	1500	bytes
target_pipe_size	9	packets
target_run_length	243	packets

Table 2

This example uses the most conservative TCP model and no derating.

8.3. Bulk delivery of remote scientific data

This example corresponds to 100 Mb/s bulk scientific data over a moderately long RTT. Note that the target_run_length is infeasible for most networks.

100 Mb/s over a 200 ms path

End to End Parameter	Value	units
target_rate	100	Mb/s
target_RTT	200	ms
target_MTU	1500	bytes
target_pipe_size	1741	packets
target_run_length	9093243	packets

Table 3

9. Validation

Since some aspects of the models are likely to be too conservative, Section 5.2 and Section 5.3 permit alternate protocol models and test parameter derating. In exchange for this latitude in the modelling process, we require demonstrations that such a TDS can robustly detect links that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified performance targets. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TDS against all possible links and subpaths.

We suggest two strategies, both of which should be applied: first, publish a fully open description of the TDS, including what assumptions were used and how it was derived, such that the research community can evaluate these decisions, test them and comment on their applicability; and second, demonstrate that applications running over an infinitesimally passing testbed do meet the performance targets.

An infinitesimally passing testbed resembles a epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TDS only pass by small (infinitesimal) margins, and demonstrate that a variety of authentic applications running over real TCP implementations (or other protocol as appropriate) meets the end-to-end target parameters over such a network. The workloads should include multiple types of streaming media and transaction oriented short flows (e.g. synthetic web traffic).

For example using our example in our HD streaming video TDS described in Section 8.1, the bottleneck data rate should be 5 Mb/s, the per packet random background loss probability should be $1/1453$, for a run length of 1452 packets, the bottleneck queue should be 22 packets and the front path should have just enough buffering to withstand 22 packet line rate bursts. We want every one of the TDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 23 packet slowstart bursts should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. On this infinitesimally passing network it should be possible for a real application using a stock TCP implementation in the vendor's default configuration to attain 5 Mb/s over an 50 ms path.

The most difficult part of setting up such a testbed is arranging to infinitesimally pass the individual tests. We suggest two approaches: constraining the network devices not to use all available resources (limiting available buffer space or data rate); and preloading subpaths with cross traffic. Note that it is important that a single environment be constructed which infinitesimally passes all tests at the same time, otherwise there is a chance that TCP can exploit extra latitude in some parameters (such as data rate) to partially compensate for constraints in other parameters (queue space, or viceversa).

To the extent that a TDS is used to inform public dialog it should be fully publicly documented, including the details of the tests, what assumptions were used and how it was derived. All of the details of the validation experiment should also be public with sufficient detail for the experiments to be replicated by other researchers. All components should either be open source or fully described

proprietary implementations that are available to the research community.

This work here is inspired by open tools running on an open platform, using open techniques to collect open data. See Measurement Lab [<http://www.measurementlab.net/>]

10. Acknowledgements

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur for helping with the statistics and contributing and alternate model.

Meredith Whittaker for improving the clarity of the communications.

11. Informative References

- [RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.
- [RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.
- [RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.
- [RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.
- [RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, February 2003.
- [RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.
- [RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, November 2006.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion

- Control", RFC 5681, September 2009.
- [RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.
- [RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, January 2011.
- [RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.
- [I-D.morton-ippm-lmap-path]
Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for LMAP", draft-morton-ippm-lmap-path-00 (work in progress), January 2013.
- [MSMO97] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review volume 27, number3, July 1997.
- [WPING] Mathis, M., "Windowed Ping: An IP Level Performance Diagnostic", INET 94, June 1994.
- [mpingSource]
Fan, X., Mathis, M., and D. Hamon, "Git Repository for mping: An IP Level Performance Diagnostic", Sept 2013, <<https://github.com/m-lab/mping>>.
- [MBMSource]
Hamon, D., "Git Repository for Model Based Metrics", Sept 2013, <<https://github.com/m-lab/MBM>>.
- [Pathdiag]
Mathis, M., Heffner, J., O'Neil, P., and P. Siemsen, "Pathdiag: Automated TCP Diagnosis", Passive and Active Measurement , June 2008.
- [StatQC] Montgomery, D., "Introduction to Statistical Quality Control - 2nd ed.", ISBN 0-471-51988-X, 1990.
- [Rtool] R Development Core Team, "R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org/>", , 2011.
- [CVST] Krueger, T. and M. Braun, "R package: Fast Cross-

Validation via Sequential Testing", version 0.1, 11 2012.

[LMCUBIC] Ledesma Goyzueta, R. and Y. Chen, "A Deterministic Loss Model Based Analysis of CUBIC, IEEE International Conference on Computing, Networking and Communications (ICNC), E-ISBN : 978-1-4673-5286-4", January 2013.

Appendix A. Model Derivations

The reference `target_run_length` described in Section 5.2 is based on very conservative assumptions: that all window above `target_pipe_size` contributes to a standing queue that raises the RTT, and that classic Reno congestion control with delayed ACKs are in effect. In this section we provide two alternative calculations using different assumptions.

It may seem out of place to allow such latitude in a measurement standard, but the section provides offsetting requirements.

The estimates provided by these models make the most sense if network performance is viewed logarithmically. In the operational Internet, data rates span more than 8 orders of magnitude, RTT spans more than 3 orders of magnitude, and loss probability spans at least 8 orders of magnitude. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 dB scale, a 3 dB error is less than 4% of the scale, even though it might represent a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating `target_run_length`, however people designing a TDS should consider the effect of their choices on the ongoing tussle about the relevance of "TCP friendliness" as an appropriate model for Internet capacity allocation. Choosing a `target_run_length` that is substantially smaller than the reference `target_run_length` specified in Section 5.2 strengthens the argument that it may be appropriate to abandon "TCP friendliness" as the Internet fairness model. This gives developers incentive and permission to develop even more aggressive applications and protocols, for example by increasing the number of connections that they open concurrently.

A.1. Queueless Reno

In Section 5.2 it is assumed that the target rate is the same as the link rate, and any excess window causes a standing queue at the bottleneck. This might be representative of a non-shared access link. An alternative situation would be a heavily aggregated subpath where individual flows do not significantly contribute to the

queueing delay, and losses are determined monitoring the average data rate, for example by the use of a virtual queue as in [AFD]. In such a scheme the RTT is constant and TCP's AIMD congestion control causes the data rate to fluctuate in a sawtooth. If the traffic is being controlled in a manner that is consistent with the metrics here, goal would be to make the actual average rate equal to the `target_data_rate`.

We can derive a model for Reno TCP and delayed ACK under the above set of assumptions: for some value of `Wmin`, the window will sweep from `Wmin` to `2*Wmin` in `2*Wmin` RTT. Unlike the queueing case where `Wmin = Target_pipe_size`, we want the average of `Wmin` and `2*Wmin` to be the `target_pipe_size`, so the average rate is the target rate. Thus we want `Wmin = (2/3)*target_pipe_size`.

Between losses each sawtooth delivers $(1/2)(Wmin+2*Wmin)(2Wmin)$ packets in `2*Wmin` round trip times.

Substituting these together we get:

`target_run_length = (4/3)(target_pipe_size^2)`

Note that this is 44% of the reference run length. This makes sense because under the assumptions in Section 5.2 the AMID sawtooth caused a queue at the bottleneck, which raised the effective RTT by 50%.

A.2. CUBIC

CUBIC has three operating regions. The model for the expected value of window size derived in [LMCUBIC] assumes operation in the "concave" region only, which is a non-TCP friendly region for long-lived flows. The authors make the following assumptions: packet loss probability, `p`, is independent and periodic, losses occur one at a time, and they are true losses due to tail drop or corruption. This definition of `p` aligns very well with our definition of `target_run_length` and the requirement for progressive loss (AQM).

Although CUBIC window increase depends on continuous time, the authors transform the time to reach the maximum Window size in terms of RTT and a parameter for the multiplicative rate decrease on observing loss, `beta` (whose default value is 0.2 in CUBIC). The expected value of Window size, `E[W]`, is also dependent on `C`, a parameter of CUBIC that determines its window-growth aggressiveness (values from 0.01 to 4).

$$E[W] = (C * (RTT/p)^3 * ((4-beta)/beta))^{1/4}$$

and, further assuming Poisson arrival, the mean throughput, `x`, is

$$x = E[W]/RTT$$

We note that under these conditions (deterministic single losses), the value of $E[W]$ is always greater than 0.8 of the maximum window size \sim reference_run_length. (as far as I can tell)

Appendix B. Complex Queueing

For many network technologies simple queueing models do not apply: the network schedules, thins or otherwise alters the timing of ACKs and data, generally to raise the efficiency of the channel allocation process when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream generally has two consequences: it raises the effective bottleneck data rate, making slowstart burst at higher rates (possibly as high as the sender's interface rate) and it effectively raises the RTT by the average time that the ACKs were delayed. The first effect can be partially mitigated by reclocking ACKs once they are beyond the bottleneck on the return path to the sender, however this further raises the effective RTT.

The most extreme example of this sort of behavior would be a half duplex channel that is not released as long as end point currently holding the channel has pending traffic. Such environments cause self clocked protocols under full load to revert to extremely inefficient stop and wait behavior, where they send an entire window of data as a single burst, followed by the entire window of ACKs on the return path.

If a particular end-to-end path contains a link or device that alters the ACK stream, then the entire path from the sender up to the bottleneck must be tested at the burst parameters implied by the ACK scheduling algorithm. The most important parameter is the Effective Bottleneck Data Rate, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACKs (relying on the cumulative nature of seg.ack to permit discarding some ACKs) is implies an effectively infinite bottleneck data rate. It is important to note that due to the self clock, ill conceived channel allocation mechanisms can increase the stress on upstream links in a long path.

Holding data or ACKs for channel allocation or other reasons (such as error correction) always raises the effective RTT relative to the minimum delay for the path. Therefore it may be necessary to replace target_RTT in the calculation in Section 5.2 by an effective_RTT,

which includes the target_RTT reflecting the fixed part of the path plus a term to account for the extra delays introduced by these mechanisms.

Appendix C. Version Control

Formatted: Fri Feb 14 14:07:33 PST 2014

Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheater Parkway
Mountain View, California 94043
USA

Email: mattmathis@google.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ 07748
USA

Phone: +1 732 420 1571
Email: acmorton@att.com
URI: <http://home.comcast.net/~acmacm/>

