Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 5, 2013
M. Hamilton
Ixia
S. Banks
Aerohive Networks
Feb 2013

Benchmarking Methodology for Content-Aware Network Devices
draft-ietf-bmwg-ca-bench-meth-04

Abstract

This document defines a set of test scenarios and metrics that can be used to benchmark content-aware network devices. The scenarios in the following document are intended to more accurately predict the performance of these devices when subjected to dynamic traffic patterns. This document will operate within the constraints of the Benchmarking Working Group charter, namely black box characterization in a laboratory environment.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Content-aware and deep packet inspection (DPI) device deployments have grown significantly in recent years. No longer are devices simply using Ethernet and IP headers to make forwarding decisions. This class of device now uses application-specific data to make these decisions. For example, a web-application firewall (WAF) may use search criteria upon the HTTP uniform resource indicator (URI)[1] to decide whether a HTTP GET method may traverse the network. In the case of lawful/legal intercept technology, a device could use the phone number within the Session Description Protocol[14] to determine whether a voice-over-IP phone may be allowed to connect. In addition to the development of entirely new classes of devices, devices that could historically be classified as 'stateless' or raw forwarding devices are now performing DPI functionality. Devices such as core and edge routers are now being developed with DPI functionality to make more intelligent routing and forwarding decisions.

The Benchmarking Working Group (BMWG) has historically produced Internet Drafts and Requests for Comment that are focused specifically on creating output metrics that are derived from a very specific and well-defined set of input parameters that are completely and unequivocally reproducible from test bed to test bed. The end goal of such methodologies is to, in the words of the RFC 2544 [2], reduce "specsmanship" in the industry and hold vendors accountable for performance claims.

The end goal of this methodology is to generate performance metrics in a lab environment that will closely relate to actual observed performance on production networks. By utilizing dynamic traffic patterns relevant to modern networks, this methodology should be able to closely tie laboratory and production metrics. It should be further noted than any metrics acquired from production networks SHOULD be captured according to the policies and procedures of the IPPM or PMOL working groups.

An explicit non-goal of this document is to replace existing methodology/terminology pairs such as RFC 2544 [2]/RFC 1242 [3] or RFC 3511 [4]/RFC 2647 [5]. The explicit goal of this document is to create a methodology more suited for modern devices while complementing the data acquired using existing BMWG methodologies. This document does not assume completely repeatable input stimulus. The nature of application-driven networks is such that a single dropped packet inherently changes the input stimulus from a network perspective. While application flows will be specified in great detail, it simply is not practical to require totally repeatable input stimulus.

1.1. 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 [6].

2. Scope

Content-aware devices take many forms, shapes and architectures. These devices are advanced network interconnect devices that inspect deep into the application payload of network data packets to do classification. They may be as simple as a firewall that uses application data inspection for rule set enforcement, or they may have advanced functionality such as performing protocol decoding and validation, anti-virus, anti-spam and even application exploit filtering. The document will universally call these devices middleboxes, as defined by RFC 3234 [7].

This document is strictly focused on examining performance and robustness across a focused set of metrics: throughput(min/max/avg/sample std dev), transaction rates(successful/failed), application response times, concurrent flows, and unidirectional packet latency. None of the metrics captured through this methodology are specific to a device and the results are DUT implementation independent. Functional testing of the DUT is outside the scope of this methodology.

Devices such as firewalls, intrusion detection and prevention devices, wireless LAN controllers, application delivery controllers, deep packet inspection devices, wide-area network(WAN) optimization devices, and unified threat management systems generally fall into the content-aware category. While this list may become obsolete, these are a subset of devices that fall under this scope of testing.

3. Test Setup

This document will be applicable to most test configurations and will not be confined to a discussion on specific test configurations. Since each DUT/SUT will have their own unique configuration, users SHOULD configure their device with the same parameters that would be used in the actual deployment of the device or a typical deployment, if the actual deployment is unknown. A summary of the DUT configuration MUST be published with the final benchmarking results. In order to improve repeatability, the published configuration information SHOULD include command-line scripts used to configure the DUT, if any, and SHOULD also include any configuration information for the test equipment used."

3.1. Test Considerations

3.2. Clients and Servers

Content-aware device testing SHOULD involve multiple clients and multiple servers. As with RFC 3511 [4], this methodology will use the terms virtual clients/servers because both the client and server will be represented by the tester and not actual clients/servers. Similarly defined in RFC 3511 [4], a data source may emulate multiple clients and/or servers within the context of the same test scenario. The test report SHOULD indicate the number of virtual clients/servers used during the test. IANA has reserved address ranges for laboratory characterization. These are defined for IPv4 and IPv6 by RFC 2544 Appendix C [2] and RFC 5180 Section 5.2 [8] respectively and SHOULD be consulted prior to testing.

3.3. Traffic Generation Requirements

The explicit purposes of content-aware devices vary widely, but these devices use information deeper inside the application flow to make decisions and classify traffic. This methodology will utilize traffic flows that resemble real application traffic without utilizing captures from live production networks. Application Flows, as defined in Section 1.1 RFC 2724 [9] are able to be well-defined without simply referring to a network capture. An example traffic template is defined and listed in Appendix A of this document. A user of this methodology is free to utilize the example mix as provided in the appendix. If a user of this methodology understands the traffic patterns in their production network, that user MAY use the template provided in Appendix A to describe a traffic mix appropriate for their environment. In all cases, users MUST report the traffic mix used in the test, and SHOULD report this using a template similar to that in Appendix A.

The test tool SHOULD be able to create application flows between every client and server, regardless of direction. The tester SHOULD be able to open TCP connections on multiple destination ports and SHOULD be able to direct UDP traffic to multiple destination ports.

3.4. Discussion of Network Limitations

Prior to executing the methodology as outlined in the following sections, it is imperative to understand the implications of utilizing representative application flows for the traffic content of the benchmarking effort. One interesting aspect of utilizing application flows is that each flow is inherently different from every other application flow. The content of each flow will vary from application to application, and in most cases, even varies within the same type of application flow. The following description of the methodology will individually benchmark every individual type and subset of application flow, prior to performing similar tests with a traffic mix as specified either by the example mix in Appendix A, or as defined by the user of this methodology.

The purpose of this process is to ensure that any performance implications that are discovered during the mixed testing aren't due to the inherent physical network limitations. As an example of this phenomena, it is useful to examine a network device inserted into a single path, as illustrated in the following diagram.

```
           +----------+
+---+  1gE |   DUT/   | 1gE  +---+
|C/S|------|   SUT    |------|C/S|
+---+      +----------+      +---+
```

Simple Inline DUT Configuration

Figure 1: Simple Middle-box Example

For the purpose of this discussion, let's take a hypothetical application flow that utilizes UDP for the transport layer. Assume that the sample transaction we will be using to model this particular flow requires 10 UDP datagrams to complete the transaction. For simplicity, each datagram within the flow is exactly 64 bytes, including associated Ethernet, IP, and UDP overhead. With any network device,there are always three metrics which interact with each other: number of concurrent application flows, number of application flows per second, and layer-7 throughput.

Our example test bed is a single-path device connected by 1 gigabit Ethernet links. The purpose of this benchmark effort is to quantify the number of application flows per second that may be processed through our device under test. Let's assume that the result from our scenario is that the DUT is able to process 10,000 application flows per second. The question is whether that ceiling is the actual ceiling of the device, or if it is actually being limited by one of the other metrics. If we do the appropriate math, 10000 flows per second, with each flow at 640 total bytes means that we are achieving an aggregate bitrate of roughly 49 Mbps. This is dramatically less than the 1 gigabit physical link we are using. We can conclude that 10,000 flows per second is in fact the performance limit of the device.

If we change the example slightly and increase the size of each datagram to 1312 bytes, then it becomes necessary to recompute the load. Assuming the same observed DUT limitation of 10,000 flows per second, it must be ensured that this is an artifact of the DUT, and not of physical limitations. For each flow, we'll require 104,960 bits. 10,000 flows per second implies a throughput of roughly 1 Gbps. At this point, we cannot definitively answer whether the DUT is actually limited to 10,000 flows per second. If we are able to modify the scenario, and utilize 10 Gigabit interfaces, then perhaps the flow per second ceiling will be reached at a higher number than 10,000.

This example illustrates why a user of this methodology SHOULD benchmark each application variant individually to ensure that the cause of a measured limit is fully understood

3.5. Framework for Traffic Specification

The following table SHOULD be specified for each application flow variant.

- o Data Exchanged By Flow, Bits
- o Offered Percentage of Total Flows
- o Transport Protocol(s)
- o Destination Port(s)

3.6. Multiple Client/Server Testing

In actual network deployments, connections are being established between multiple clients and multiple servers simultaneously. Device vendors have been known to optimize the operation of their devices for easily defined patterns. The connection sequence ordering scenarios a device will see on a network will likely be much less deterministic. In fact, many application flows have multiple layer 4 connections within a single flow, with client and server reversing roles. Flow initiation SHOULD be in a pseudo-random manner across ingress ports.

3.7. Device Configuration Considerations

The configuration of the DUT may have an effect on the observed results of the following methodology. A comprehensive, but certainly not exhaustive, list of potential considerations is listed below.

3.7.1. Network Addressing

The IANA has issued a range of IP addresses to the BMWG for purposes of benchmarking. Please refer to RFC 2544 [2] and RFC 5180 [8] for more details. If more IPv4 addresses are required than the RFC 2544 allotment provides, then allocations from the private address space as defined in RFC 1918 [10] may be used.

3.7.2. Network Address Translation

Many content-aware devices are capable of performing Network Address Translation (NAT)[5]. If the final deployment of the DUT will have this functionality enabled, then the DUT SHOULD also have it enabled during the execution of this methodology. It MAY be beneficial to perform the test series in both modes in order to determine the performance differential when using NAT. The test report SHOULD indicate whether NAT was enabled during the testing process.

3.7.3. TCP Stack Considerations

The IETF has historically provided guidance and information on TCP stack considerations. This methodology is strictly focused on performance metrics at layers above 4, thus does not specifically define any TCP stack configuration parameters of either the tester or the DUTs. The TCP configuration of the tester MUST remain constant across all DUTs in order to ensure comparable results. While the following list of references is not exhaustive, each document contains a relevant discussion on TCP stack considerations.

The general IETF TCP roadmap is defined in RFC 4614 [11] and congestion control algorithms are discussed in Section 2 of RFC 3148 [12] with even more detailed references. TCP receive and congestion window sizes are discussed in detail in RFC 6349 [13].

3.7.4. Other Considerations

Various content-aware devices will have widely varying feature sets. In the interest of representative test results, the DUT features that will likely be enabled in the final deployment SHOULD be used. This methodology is not intended to advise on which features should be enabled, but to suggest using actual deployment configurations.

4. Benchmarking Tests

Each of the following benchmark scenarios SHOULD be run with each of the single application flow templates. Upon completion of all iterations, the mixed test SHOULD be completed, subject to the traffic mix as defined by the user.

4.1. Maximum Application Session Establishment Rate

4.1.1. Objective

To determine the maximum rate through which a device is able to establish and complete application flows as defined by draft-ietf-bmwg-ca-bench-term-00.

4.1.2. Setup Parameters

The following parameters SHOULD be used and reported for all tests:

For each application protocol in use during the test run, the table provided in Section 3.5 SHOULD be published.

4.1.3. Procedure

The test SHOULD generate application network traffic that meets the conditions of Section 3.3. The traffic pattern SHOULD begin with an application flow rate of 10% of expected maximum. The test SHOULD be configured to increase the attempt rate in units of 10% up through 110% of expected maximum. In the case where expected maximum is limited by physical link rate as discovered through Appendix A, the maximum rate will attempted will be 100% of expected maximum, or "wire-speed performance". The duration of each loading phase SHOULD be at least 30 seconds. This test MAY be repeated, each subsequent iteration beginning at 5% of expected maximum and increasing session establishment rate to 110% of the maximum observed from the previous test run.

This procedure MAY be repeated any reasonable number of times with the results being averaged together.

4.1.4. Measurement

The following metrics MAY be determined from this test, and SHOULD be observed for each application protocol within the traffic mix:

4.1.4.1. Maximum Application Flow Rate

The test tool SHOULD report the maximum rate at which application flows were completed, as defined by RFC 2647 [5], Section 3.7. This rate SHOULD be reported individually for each application protocol present within the traffic mix.

4.1.4.2. Application Flow Duration

The test tool SHOULD report the minimum, maximum and average application duration, as defined by RFC 2647 [5], Section 3.9. This duration SHOULD be reported individually for each application protocol present within the traffic mix.

4.1.4.3. Application Efficiency

The test tool SHOULD report the application efficiency, similarly defined for TCP by RFC 6349 [13].

```
                          Transmitted Bytes - Retransmitted Bytes
      App Efficiency % =  ---------------------------------------  X 100
                                     Transmitted Bytes
```

Figure 2: Application Efficiency Percent Calculation

Note than calculation less than 100% does not necessarily imply noticeably degraded performance since certain applications utilize algorithms to maintain a quality user experience in the face of data loss.

4.1.4.4. Application Flow Latency

The test tool SHOULD report the minimum, maximum and average amount of time an application flow member takes to traverse the DUT, as defined by RFC 1242 [3], Section 3.8. This value SHOULD be reported individually for each application protocol present within the traffic mix.

4.2. Application Throughput

4.2.1. Objective

To determine the maximum rate through which a device is able to forward bits when using application flows as defined in the previous sections.

4.2.2. Setup Parameters

The same parameter reporting procedure as described in Section 4.1.2 SHOULD be used for all tests.

4.2.3. Procedure

This test will attempt to send application flows through the device at a flow rate of 30% of the maximum, as observed in Section 4.1. This procedure MAY be repeated with the results from each iteration averaged together.

4.2.4. Measurement

The following metrics MAY be determined from this test, and SHOULD be observed for each application protocol within the traffic mix:

4.2.4.1. Maximum Throughput

The test tool SHOULD report the minimum, maximum and average application throughput.

4.2.4.2. Maximum Application Flow Rate

The test tool SHOULD report the maximum rate at which application flows were completed, as defined by RFC 2647 [5], Section 3.7. This rate SHOULD be reported individually for each application protocol present within the traffic mix.

4.2.4.3. Application Flow Duration

The test tool SHOULD report the minimum, maximum and average application duration, as defined by RFC 2647 [5], Section 3.9. This duration SHOULD be reported individually for each application protocol present within the traffic mix.

4.2.4.4. Application Efficiency

The test tool SHOULD report the application efficiency as defined in Section 4.1.4.3.

4.2.4.5. Packet Loss

The test tool SHOULD report the number of packets lost or dropped from source to destination.

4.2.4.6. Application Flow Latency

The test tool SHOULD report the minimum, maximum and average amount of time an application flow member takes to traverse the DUT, as defined by RFC 1242 [3], Section 3.13. This value SHOULD be reported individually for each application protocol present within the traffic mix.

4.3. Malformed Traffic Handling

4.3.1. Objective

To determine the effects on performance and stability that malformed traffic may have on the DUT.

4.3.2. Setup Parameters

The same parameters SHOULD be used for Transport-Layer and Application Layer Parameters previously specified in Section 4.1.2 and Section 4.2.2.

4.3.3. Procedure

This test will utilize the procedures specified previously in Section 4.1.3 and Section 4.2.3. When performing the procedures listed previously, the tester should generate malformed traffic at all protocol layers. This is commonly known as fuzzed traffic. Fuzzing techniques generally modify portions of packets, including checksum errors, invalid protocol options, and improper protocol conformance.

The process by which the tester SHOULD generate the malformed traffic is outlined in detail in Appendix B.

4.3.4. Measurement

For each protocol present in the traffic mix, the metrics specified by Section 4.1.4 and Section 4.2.4 MAY be determined. This data may be used to ascertain the effects of fuzzed traffic on the DUT.

5. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see the update of RFC 2434 [15] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

6. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints RFC 2544 [2].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or mis-route traffic to the test management network

7. References

7.1. Normative References

[1] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[2] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

[3] Bradner, S., "Benchmarking terminology for network interconnection devices", RFC 1242, July 1991.

[4] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, April 2003.

[5] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, August 1999.

[6] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[7] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[8] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, May 2008.

[9] Handelman, S., Stibler, S., Brownlee, N., and G. Ruth, "RTFM: New Attributes for Traffic Flow Measurement", RFC 2724, October 1999.

[10] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[11] Duke, M., Braden, R., Eddy, W., and E. Blanton, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 4614, September 2006.

[12] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

[13] Constantine, B., Forget, G., Geib, R., and R. Schrage, "Framework for TCP Throughput Testing", RFC 6349, August 2011.

7.2. Informative References

[14] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[15] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

7.3. URL References

[16] Sandvine Corporation, "http://www.sandvine.com/general/document.download.asp?docID=58&sourceID=0", 2012.

Appendix A. Example Traffic Mix

This appendix shows an example case of a protocol mix that may be used with this methodology. This mix closely represents the research published by Sandvine [16] in their biannual report for the first half of 2012 on North American fixed access service provider networks.

Direction	Application Flow	Options	Value
Upstream	BitTorrent		
		Avg Flow Size (L7)	512 MB
		Flow Percentage	44.4%
	HTTP		
		Avg Flow Size (L7)	128 kB
		Flow Percentage	7.3%
	Skype		
		Avg Flow Size (L7)	8 MB
		Flow Percentage	4.9%
	SSL/TLS		
		Avg Flow Size (L7)	128 kB
		Flow Percentage	3.2%
	Netflix		

```
   |           |                  | Avg Flow Size (L7) | 500 kB |
   |           |                  |   Flow Percentage  |  3.1%  |
   |           |     PPStream     |                    |        |
   |           |                  | Avg Flow Size (L7) | 500 MB |
   |           |                  |   Flow Percentage  |  2.2%  |
   |           |      YouTube     |                    |        |
   |           |                  | Avg Flow Size (L7) |  4 MB  |
   |           |                  |   Flow Percentage  |  1.9%  |
   |           |     Facebook     |                    |        |
   |           |                  | Avg Flow Size (L7) |  2 MB  |
   |           |                  |   Flow Percentage  |  1.9%  |
   |           |      Teredo      |                    |        |
   |           |                  | Avg Flow Size (L7) | 500 MB |
   |           |                  |   Flow Percentage  |  1.2%  |
   |           |  Apple iMessage  |                    |        |
   |           |                  | Avg Flow Size (L7) |  40 kB |
   |           |                  |   Flow Percentage  |  1.1%  |
   |           |     Bulk TCP     |                    |        |
   |           |                  | Avg Flow Size (L7) | 128 kB |
   |           |                  |   Flow Percentage  |  28.8% |
   | Downstream|      Netflix     |                    |        |
   |           |                  | Avg Flow Size (L7) | 512 MB |
   |           |                  |   Flow Percentage  |  32.9% |
   |           |      YouTube     |                    |        |
   |           |                  | Avg Flow Size (L7) |  5 MB  |
   |           |                  |   Flow Percentage  |  13.8% |
   |           |       HTTP       |                    |        |
   |           |                  | Avg Flow Size (L7) |  1 MB  |
   |           |                  |   Flow Percentage  |  12.1% |
   |           |    BitTorrent    |                    |        |
   |           |                  | Avg Flow Size (L7) | 500 MB |
   |           |                  |   Flow Percentage  |  6.3%  |
   |           |      iTunes      |                    |        |
   |           |                  | Avg Flow Size (L7) |  32 MB |
   |           |                  |   Flow Percentage  |  3.8%  |
   |           |    Flash Video   |                    |        |
   |           |                  | Avg Flow Size (L7) | 100 MB |
   |           |                  |   Flow Percentage  |  2.6%  |
   |           |       MPEG       |                    |        |
   |           |                  | Avg Flow Size (L7) | 100 MB |
   |           |                  |   Flow Percentage  |  2.0%  |
   |           |       RTMP       |                    |        |
   |           |                  | Avg Flow Size (L7) |  50 MB |
   |           |                  |   Flow Percentage  |  2.0%  |
   |           |       Hulu       |                    |        |
   |           |                  | Avg Flow Size (L7) | 300 MB |
   |           |                  |   Flow Percentage  |  1.8%  |
   |           |      SSL/TLS     |                    |        |
```

		Bulk TCP	Avg Flow Size (L7)	256 kB
			Flow Percentage	1.6%
			Avg Flow Size (L7)	500 kB
			Flow Percentage	21.1%

Table 1: Example Traffic Pattern

Appendix B. Malformed Traffic Algorithm

Each application flow will be broken into multiple transport segments, IP packets, and Ethernet frames. The malformed traffic algorithm looks very similar to the IP Stack Integrity Checker project at http://isic.sourceforge.net.

The algorithm is very simple and starts by defining each of the fields within the TCP/IP stack that will be malformed during transmission. The following table illustrates the Ethernet, IPv4, IPv6, TCP, and UDP fields which are able to be malformed by the algorithm. The first column lists the protocol, the second column shows the actual header field name, with the third column showing the percentage of packets that should have the field modified by the malformation algorithm.

Protocol	Header Field	Malformed %
Total Frames		1%
Ethernet		
	Destination MAC	0%
	Source MAC	1%
	Ethertype	1%
	CRC	1%
IP Version 4		
	Version	1%
	IHL	1%
	Type of Service	1%
	Total Length	1%
	Identification	1%
	Flags	1%
	Fragment Offset	1%
	Time to Live	1%
	Protocol	1%
	Header Checksum	1%
	Source Address	1%
	Destination Address	1%
	Options	1%
	Padding	1%
UDP		
	Source Port	1%
	Destination Port	1%
	Length	1%
	Checksum	1%
TCP		
	Source Port	1%
	Destination Port	1%
	Sequence Number	1%
	Acknowledgement Number	1%
	Data Offset	1%
	Reserved(3 bit)	1%
	Flags(9 bit)	1%
	Window Size	1%
	Checksum	1%
	Urgent Pointer	1%
	Options(Variable Length)	1%

Table 2: Malformed Header Values

This algorithm is to be used across the regular application flows used throughout the rest of the methodology. As each frame is emitted from the test tool, a pseudo-random number generator will indicate whether the frame is to be malformed by creating a number between 0 and 100. If the number is less than the percentage defined in the table, then that frame will be malformed. If the frame is to be malformed, then each of the headers in the table present within the frame will follow the same process. If it is determined that a header field should be malformed, the same pseudo-random number generator will be used to create a random number for the specified header field.

Authors' Addresses

Mike Hamilton
Ixia
Austin, TX 78730
US

Phone: +1 512 636 2303
Email: mhamilton@ixiacom.com

Sarah Banks
Aerohive Networks
San Jose, CA 95134
US

Email: sbanks@aerohive.com