In-line Service Measurements: An IPv6-based Framework for Traffic Evaluation and Network Operations

D. P. Pezaros, D. Hutchison
Computing Department
Lancaster University
Lancaster, LA1 4YR, UK
{dp, dh}@comp.lancs.ac.uk

F. J. Garcia, R. D. Gardner
Agilent Laboratories, Scotland
Agilent Technologies
Edinburgh, EH30 9TG, UK
{frankie_garcia, robert_gardner}@agilent.com

J. S. Sventek
Department of Computing Science
Univeristy of Glasgow
Glasgow, G12 8QQ, UK
joe@dcs.gla.ac.uk

Abstract

The ability to measure, monitor and control the service quality experienced by network traffic is becoming increasingly important as multiple traffic types are aggregated onto IP networks. Assessing the real-time performance of the application flows is an essential requirement for network operations and service management, as well as for identifying how the different traffic types and transports interact and behave, when they are carried over the end-to-end Internet infrastructure.

This paper introduces a novel measurement technique for assessing the performance of IPv6 network flows. By exploiting IPv6 extension headers, measurement triggers and the instantaneous measurement indications are carried in the same packets as the payload data itself, providing a high level of probability that the behaviour of the real user traffic flows is being observed. The measurement mechanism is applied at the network layer and provides for a generic technique able to measure any type of traffic, without depending on particular transports nor on specific measurement architectures. A prototype implementation of this technique is also described and evaluated by measuring performance properties of application flows, over different-capacity IPv6 environments. End-to-end delay and jitter of video streams have been measured, as well as the goodput for services operating on top of reliable transport. This measurement technique can be the basis for low-overhead, scalable, transparent and reliable measurement of individual and aggregate network flows, and can be dynamically deployed where and when required in a multi-service IP environment.

Keywords

Active/passive measurements, IPv6, one-way delay/jitter/loss, throughput

1. Introduction

As the Internet grows larger and becomes increasingly heterogeneous, measuring the real-time properties and the service quality experienced by user traffic becomes even more challenging, and moreover, of critical importance. Traditional measurement techniques have focused on providing fault management support for the network, and usually adopt a link- or device-centric view in order to trace the sources of and reasons for service degradation. More recent measurement infrastructures specialise either in measuring the path properties of special-purpose, injected traffic (active measurements) [1, 6, 9, 11, 12, 13, 18] or in correlating and analysing one-point link traffic (passive measurements and standalone monitoring tools) [2, 3, 4, 8, 15].

Active measurement architectures produce results reflecting the performance experienced by the synthetic traffic (mainly based on ICMP or other special-purpose protocols), which is not necessarily identical to the performance experienced by the real traffic flows. Also, the additional traffic associated with active measurements obviously impacts the network and may itself be a factor in observing a poorer performance than the network would otherwise deliver [13].

Passive measurements suffer from continuous increases in network speeds that make the amount of measurement data that needs to be transferred across the monitored links substantial; the consequential difficulty in targeting specific services, identifying packets belonging to the same flows and inferring knowledge about them makes it difficult to prove that contractual agreements are being met [11]. The need for administrative access to network equipment limits the scope of passive measurements to single links and networks, and hence, end-to-end traffic behaviour is difficult to infer [2, 7, 8].

The Internet Protocol (IP) is evolving as the universal transport mechanism for the continuing convergence of telephony with data communications, where there is increasing aggregation of multi-service traffic onto IP networks carrying various equivalence classes and network flows. The different Quality of Service (QoS) requirements and different sensitivities to potential service degradation of these equivalence classes make it essential to determine timely and accurate measurements of the QoS of network flows. Measurements revealing the real service experienced by user traffic can prove valuable for long and short term network design decisions, for dynamic traffic engineering, for Service Level Agreement (SLA) negotiation and policing, and for network operations and service management.

Whereas IPv4 is the current ubiquitously-deployed version of the Internet Protocol, it is likely that IPv6 will increasingly be adopted as its advantages are revealed. Router and client system vendors are building IPv6 implementations into their products, and one of the biggest challenges will be the logistical problem of handover from v4 to v6. But it is still not universally accepted that IPv6 should be adopted, and there needs to be a convincing rationale for such acceptance: this paper investigates one possible feature of IPv6 that could be seen as an advantage for network operators and users, in the context of traffic evaluation and network operations.

The main contributions of this paper are the introduction, design and implementation of a new, IPv6-based, in-line measurement technique, able to assess the performance properties experienced by real user traffic transparently at the IP layer. The technique exploits the inherent programmability of IPv6 [5], and performs two-point measurements for any type of traffic carried over end-to-end or intermediate Internet paths. The term ‘in-line’ implies that measurement triggers, which invoke measurement activity and the instantaneous measurement indications, are piggybacked onto real user packets. With low overhead and minimal impact on the network traffic, the technique provides a high level of probability that the real user experience is being measured and it is equally applicable to measuring aggregate flows as it is to particular applications or protocols. Additionally, it is well-suited to dynamic deployment and promises a scalable implementation. The technique is facilitated by the steady introduction of IPv6 in research and academic networks; the consumption of the IP address space primarily driven by the growth in mobile and wireless markets requiring IP connectivity, also makes a strong case for widespread operational deployment of IPv6.

This paper is organised as follows: Section 2, introduces in-line measurements and the motivation and design decisions are presented. Section 3 describes the prototype design and implementation. Section 4 discusses the results taken over different experimental, IPv6 configurations. Section 5 contains concluding remarks and indicates some of the future directions of this work.

2. In-line Measurement Technique

The acquisition of low-overhead, scalable, accurate, and service-oriented measurements of network performance has led to the investigation of mechanisms that can circumvent some of the inherent difficulties of active and passive techniques. Indeed, the in-line measurement technique may be viewed as a hybrid of the beneficial characteristics of active and passive measurement approaches, whereby the measurement data and triggering mechanisms are piggybacked onto real user packets. Through instrumentation of real network flows, it provides for accurate measurements reflecting the performance offered by the network infrastructure; additionally, it offers the flexibility of being able to target specific services, flows and/or aggregates.

In-line measurements are intrinsically multi-point measurements whereby packets are tagged with information at one point in the network and this information is retrieved and/or observed elsewhere.

Since it is the user traffic itself which carries the measurement and triggering mechanism, it is guaranteed that the same packet has been observed at both ends. Similarly, because any added data will be piggybacked onto real user traffic we can assume with a high degree of probability that it will receive the same treatment and follow the same path as the real user traffic. Unlike active measurements consisting of injected packets, two-point in-line measurement results will more accurately reflect the real characteristics of traffic flows with only a small additional systematic processing delay and marginally larger overall packet length compared to unaffected packets. This makes the reasonable assumption that, for appropriately selected packets, the marginal increase in the packet header length does not change how the packet is treated on its journey through the network.

Moreover, in contrast to two-point passive measurements, correlation of data from path endpoints is not necessary, reducing the complexity of the measurement system, potentially reducing the amount of measurement data that must be shipped across the network and speeding up the availability of the measurement results. In fact, the total amount of additional traffic transported across the network for measurement purposes can be kept to a minimum.

2.1 IPv6 Enhancements

Native features of Internet Protocol version 6 (IPv6) can be conveniently used to implement in-line measurements, potentially being fully integrated into the protocol stack operation using IPv6 extension headers [5]. The protocol has a common, 40-octet, fixed-sized protocol header that mainly deals with addressing, while the rest of the functionality is implemented via a set of optional extensions headers containing data structures called 'options'. Unlike IPv4, where the options must be implemented by all IP modules (host and gateways), IPv6 allows for certain options to be implemented and processed only where necessary, at the edges of the network, removing the complexity of option processing at every node en-route to the destination. (Whereas in IPv4, what is optional is the transmission of the options in any particular datagram, not their implementation - RFC 791.) In IPv6, intermediate nodes do not have to deal with option processing, and the concern of treating packets differently (fast/slow paths in routers) depending on whether they carry optional data or not, is thereby eliminated.

Several extension headers (e.g. Routing, hop-by-hop options, destination options) and corresponding options have already been defined in the IPv6 standard [5] and others are currently being defined for dedicated purposes - e.g. within recent Mobile IPv6 IETF drafts. Optional extension headers are encoded between the main IPv6 header and the transport layer header.

Each header is identified by the value of the Next Header field of the immediately preceding header. The rules and semantics of each extension header determine whether the receiving node will proceed to the next header or not. Figure 1(a) shows the general format of the Destination Options extension header which is defined to carry optional data to be examined by a datagram's destination, indicated by the destination address field of the main IPv6 header. The Next Header field contains a value that identifies the type of the subsequent header, which could either be another extension header or higher-layer protocol such as UDP or TCP. The Hdr Ext Len field gives the total length in octets of the destination options header excluding the 8 bits representing the Next Header Field.

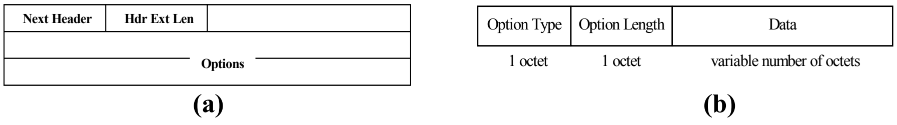


Figure 1: (a) Destination options extension header and (b) options in the form of TLV tuples.

The options field is a variable length field in which options are encoded as type-length-value (TLV) tuples, as shown in figure 1(b); these represent a suitable format for the transportation of opaque objects.

The type uniquely identifies the option. The length indicates the length of the option data field in octets, while the value represents the option specific data. Options are processed in sequence and the option type also has two bits that specify the action to be taken by an IPv6 node when it does not recognise a particular option in the sequence - e.g. skip the option, discard the packet, etc. Another bit in the type field is used to indicate whether the option data value may change en-route to its destination. Padding options also exist to help with alignment issues when constructing a destination options header to ensure that it represents a multiple of 8 octets in length.

2.2 IPv6 Extension Headers & Inline Measurements

In this paper, it is proposed that in-line measurements be implemented by exploiting the IPv6 extension headers, in particular the Destination Options extension header. Extension headers can be used within a measurement infrastructure to carry measurement information in the form of TLV-encoded options, such as timestamps, counters and trace information as well as other associated measurement system traffic. Destination options on their own can be used to perform end-to-end in-line service measurements along IPv6 network paths. With the addition of a routing header, it is possible to target specific nodes en-route to enable the implementation of more detailed service measurements as the user traffic crosses crucial points of a network cloud. It would also be possible to use some of the flow-label bits to easily identify and trigger measurement and monitoring behaviour as the user traffic with the in-line data visits nodes en-route to its destination.

Performing service-centric, multi-point measurements at the IPv6 layer has the great advantage allowing the instrumentation of various different types of traffic carried over the (IPv6) Internet. Also, a generic, minimal measurement mechanism within the protocol stack, which is decoupled from particular measurement architectures, can be applicable to different application domains, such as traffic engineering, protocol performance evaluation, dynamic SLA negotiation and policing, pricing and charging for different services.

The insertion of the extension headers into real user traffic for the purpose of measurement and monitoring can be dynamically controlled depending on a particular management application requirement. Thus not all user packets need to carry this data. Selection can be based on application and sampling can also be effectively used.

It is important to ensure that the addition of measurement data does not become intrusive and therefore adversely affect the traffic being measured.

For example, the addition of measurement data should not cause a packet to exceed the link/path maximum transmission unit (MTU), since the resulting fragmentation/reassembly would influence the performance, and would introduce bias into the measurement process.

2.3 One-Way Delay Destination Options Type-Length-Value (TLV)

Extensive experimentation towards the definition of measurements options (to be processed as part of the destination options extension header) has been carried out, and numerous TLVs have been designed to measure end-to-end one-way delay, delay variations, throughput, packet loss, response time. In this paper the concentration is on the one-way delay TLV-encoded option (Figure 2), and the associated metrics that can be measured and deduced. The TLV has the following special-purpose fields:

- *Pointer* – 8-bit unsigned integer. Used to indicate the location of the next unused slot in the option data, i.e. for storage of a timestamp.
- *Overflow* – 8-bit unsigned integer. Used to indicate if an attempt is made to store more timestamps than there are slots to accommodate them.
- *Flags* – Octet comprising eight binary flags, for example for indicating the nature of data stored elsewhere in the option data fields.
- *Reserved* – A zero-valued octet included for alignment purposes
- *Source timestamp* – Two 32-bit unsigned integers indicating time of forwarding of the packet from the interface of the node where the extension header or option was inserted. The two integers represent the seconds and microseconds portions respectively of the time elapsed since 0000 hrs on 1st January 1970 Universal Co-ordinated Time (UTC).
- *Destination timestamp* – Two 32-bit unsigned integers indicating time of receipt of the packet at the interface of the node where the extension header is detected.

The Option Type identifier in the header is set to a value allocated to identify "one-way end-to-end delay measurement". In this prototype, this octet has been set to a value 00100001$_2$ (33 in decimal), also ensuring that the option is skipped if it is an unrecognised option and indicating that the data may change en-route.
This option can be used to measure one-way delay within a section of a network, such as between ingress and egress points of a packet flow across the boundaries of a section under the control of a single operator. However, this would involve modification of the standard processing procedures for Destination options or a new extension header to be defined. Note also that the option is equally applicable to "end-to-end" measurements of one-way delay, such as from a server (e.g. serving website) to a client (e.g. a wireless connected personal digital assistant (PDA) running a web browser application).

		Option type	Option data len
Pointer	Overflow	Flags	(Reserved)
Source timestamp: seconds			
Source timestamp: microseconds			
Destination timestamp: seconds			
Destination timestamp: microseconds			

Figure 2: One-way delay TLV-encoded option.

In the next section, the design, implementation and operation of such options is described, as part of a working prototype in-line measurement framework that facilitates various traffic measurements.

3. Prototype Design and Implementation

A core set of application frameworks have been developed to handle the initialisation, control, co-ordination, and scheduling of tests in a distributed network. They provide capabilities for collection, wire-format representation and streaming of measurement samples to registered consumers. Underpinning this design philosophy is the notion of *telemetry modules* which are the realisation of particular IPv6 measurement options; these are the basic components employed in the prototype to instrument nodes to facilitate in-line measurement techniques through the addition, modification and removal of data in the extension headers. These modules are remotely configured, managed and controlled. At a system level, the required functionality for performing delay and other measurements can be implemented, for example, by using dynamically loadable modules to provide additional processing logic for the manipulation of packet extensions in headers and other supporting functions such as the storage, retrieval, correlation and forwarding of measurement-related data. By modularising the set of monitoring and measurement tasks it is possible to dynamically load only those modules that are needed at a particular time and then unload them once they are no longer required. The loadable modules may be remotely delivered to the nodes, loaded and configured and, whilst in use, effectively become an integral, embedded part of the node's operating software.

A modular approach can reduce memory usage, minimise the actively used processing logic, speed up processing time, and reduce overall subsystem complexity. Another significant advantage of this approach is that it eliminates the need for physical access to the links between network nodes in order to monitor passing data.

It instead makes use of spare programmable logic or processing capability within the routers or other network devices, providing a more integrated, inherently powerful solution. Upgrades involve delivering new modules to nodes (for example over the network itself), which can either be directly loaded on delivery or be temporarily stored on some form of local media (e.g. hard disk storage) for later use.

Figure 3 shows an illustrative architecture of a single network element with a number of line interfaces and a controller comprising a processor, memory and program software or firmware. The figure illustrates three different example integration points where, depending on the design of the network element, dynamically loadable modules can be accommodated:

- The modules may be loaded onto a line interface as dynamically reconfigurable hardware (e.g. using Field Programmable Gate Arrays – FPGAs), as software or as a hybrid combination of both options.
- The modules may exist as loadable software in the software operating system 'kernel' or equivalent for the controller;
- The modules may exist as loadable applications in 'user' space provided by the controller's operating system.

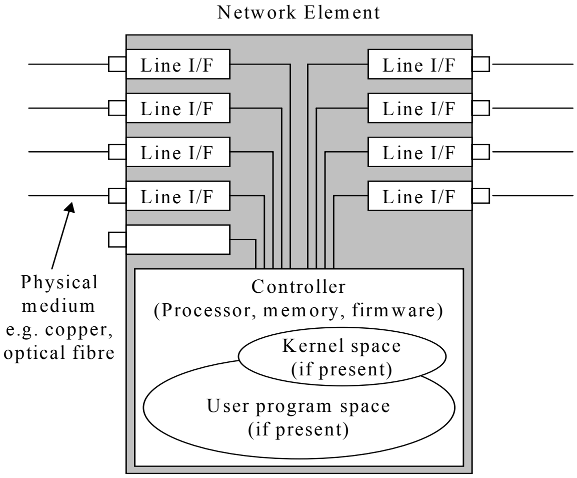


Figure 3: Abstract model of a network element.

3.1 Linux-based Prototype Implementation

A prototype implementation of this design philosophy has been realised on Linux systems running kernel version 2.4.x; telemetry modules are implemented as dynamically Loadable Kernel Modules (LKMs) that can be linked with a running Linux kernel. These typically provide better processing performance compared to applications executing in user space, and can easily be configured to add, remove and modify extension headers as an integral part of the kernel's implementation of the network protocol stack. LKMs are loaded and unloaded using device-specific scripts, and virtual devices are used for configuring the modules' parameters and also for obtaining the results from their operation. Filtering and sampling mechanisms that can be initially and/or dynamically configured are in place, so that the modules can instrument specific traffic of interest at certain rates. These mechanisms are particularly useful when analysing aggregate or continuous media flows. The one-way delay destination options header has been implemented as a pair of telemetry modules, running at the source/ingress and destination/egress nodes of an instrumented path, respectively (Figure 4). The source module operates on the IPv6 output routine, inspecting packets before they are forwarded on the network. For packets that satisfy the filtering and sampling criteria in place, the one-way delay option is generated, time-stamped and either appended to an existing destination options header or one is created for it.

The destination module operates alongside the IPv6 input routine and looks for IPv6 packets whose protocol (next header) field is set to the value 60 indicating a Destination Options header [5]. Once found, it processes the option with value 33 by adding the second time-stamp. A copy of the IPv6 header, extension header, and transport layer header is then added to a FIFO message queue, which can be accessed from the user space. If required, the extension header may also be stripped from the

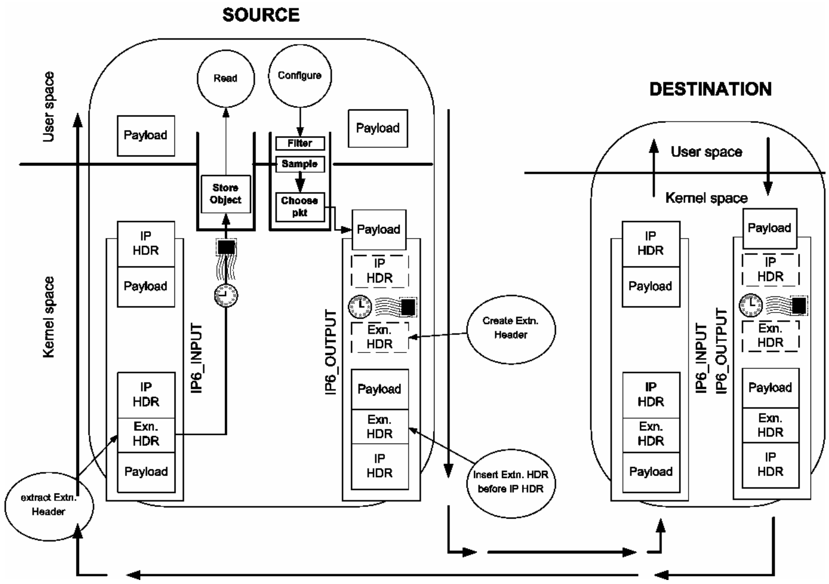


Figure 4: One-way delay source and destination modules in operation.

packet, or the particular option removed, returning the packet to its original form. Figure 4 also shows how the source and destination telemetry modules can be exploited to measure bi-directional properties of Internet paths, by combining two unidirectional deployments.

4. Experimental Results

An experimental measurement testbed has been set up at Lancaster University comprising a number of real PCs and virtual machines connected to several IPv6 networks with different characteristics. The testbed spans a public and two private ethernet topologies, and also multiple operational wireless 802.11b networks. Lately, the size and scope of the testbed was enhanced by the addition of IPv6-enabled residential ADSL links. Representative types of traffic have been chosen to run over the wired and wireless topologies; various indicative properties have been measured and computed for traffic operating over connection-oriented and connectionless transports, using the prototype one-way delay option. End-to-end one-way delay and Inter-Packet Delay Variation (IPDV) are presented for video streaming on top of UDP whereas application goodput is presented for interactive TCP applications, as a more interesting property for reliable transports.

All systems synchronise using NTP [14]. One system synchronises with external stratum 1/2 servers and the rest synchronise with this, now a stratum 3 NTP server. Experimentation showed that this “hierarchy” produced better synchronisation for the testbed than using (separate) external NTP servers for all the systems. Millisecond accuracy was achieved by having a system clock as the preferred reference for the rest of the workstations. This was empirically verified by experiments over delay-free private IPv6 networks (100Mbps Ethernet). Including the clocks' offset from the single reference in the calculation was also feasible for better precision.

Improved synchronisation can potentially be provided by Global Positioning System (GPS)-based timing at around 10 μs, but GPS installation can prove difficult and costly. Also, software solutions for improving the clock rates can be investigated [17].

In order to emphasise the generalisation and applicability of in-line measurements, and the decoupling of the technique from particular measurement infrastructures, it was decided not to use measurement traffic from (even comparable) measurement tools. Rather, the traffic for the experiments was generated by existing, open-source application software. In the generation of all results that follow sampling was turned-off, instrumenting all packets that satisfied the filtering criteria; filtering was applied on the transport protocol (i.e. all TCP or all UDP traffic).

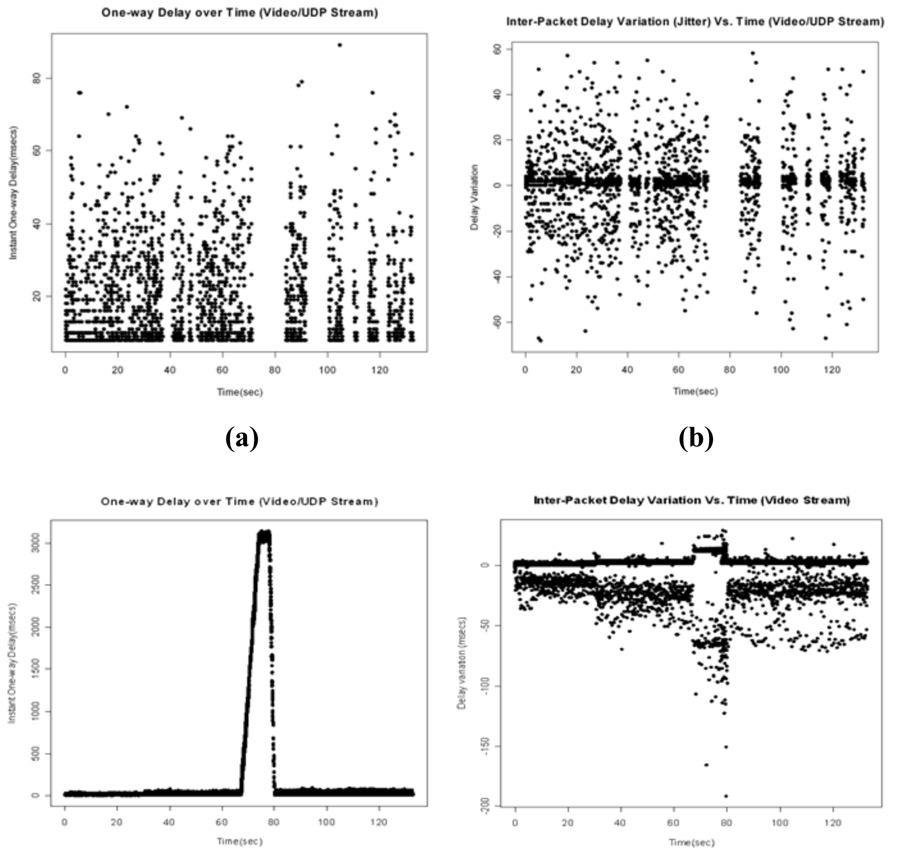
4.1 UDP Measurements

Streamed video data was monitored over both the wired and wireless topologies, using the in-line measurement techniques and the prototype implementation described previously. The main focus was on the properties of the wireless portion of the network, since the private Ethernet networks constitute a relatively delay-free topology and therefore useful for time synchronisation purposes.

The VideoLAN [19] server/client pair was used to stream MPEG video over IPv6 between the different nodes in the testbed. The software sent 1348-byte packets from source to destination. One-way delay was measured as the difference between the timestamps recorded at the source and the destination of the packet, respectively, carried within every TLV-encoded option. Jitter was measured as the difference between the delay indications of two successive packets, i.e. the Inter-Packet Delay Variation (IPDV), as defined in a draft of the IETF IPPM working group [10].

Figure 5 shows a common case scenario of the one-way delay (a) and jitter (b), measured for a video stream spanning through two wireless IPv6 networks. The instantaneous one-way delay for this video stream varied from 8 to 89 milliseconds and the average delay was 20.97 milliseconds. IPDV varied from -68 to 58 and its average value was -0.00049. There is a high concentration of delay values close to zero, but at the same time quite frequent sporadic delays, mainly in the interval 0-40 milliseconds. Figure 5 (c) and (d) shows a more interesting situation that arose during a video streaming session (part of the same set of experiments): There is a huge increase in the delay indications from less than 100 milliseconds up to more than 3 seconds, followed by an even more sudden decrease back to the flow's normal values. This fluctuation possibly occurred due to configuration changes taking place at the time of the experiment, at the infrastructure of the wireless network. The one-way delay varied from 5 to 3138 milliseconds and the average value was 148.16 milliseconds. IPDV took values from -192 to 29 with an average of 0.000443.

As expected, experiments over the two private networks revealed no particularly interesting results. The one-way delay always assumed values between one and two milliseconds and hence the jitter varied between -1 and 1.



(a) **(b)**

(c) **(d)**

Figure 5: (a, c) One-way Delay and (b, d) IPDV for video stream, over the wireless topology.

The measurements take into account clock drifts in each system from the appropriate NTP server. While running, NTP improves the clock's accuracy, and as the polling intervals increase (e.g. 1024 seconds), the clock drifts tend towards a fairly static and stable value.

4.2 TCP Measurements

The TCP results were collected by running bi-directional delay source and destination modules as previously illustrated in Figure 4. As aforementioned, the instrumented sessions consisted of shortlived SSH and Telnet flows. All the running protocol stacks within the testbed adhered to TCP with the NewReno and SACK extensions - see RFCs 793, 1122 and 2001. From the collected traces, conversation dictionaries were built based on source/destination addresses and port numbers. These show conversation set-up time (the time between a SYN packet sent from the client and a SYN+ACK packet received from the server) and “Completeness” of a flow, signifying that at least one FIN packet, sent in either direction, was observed.

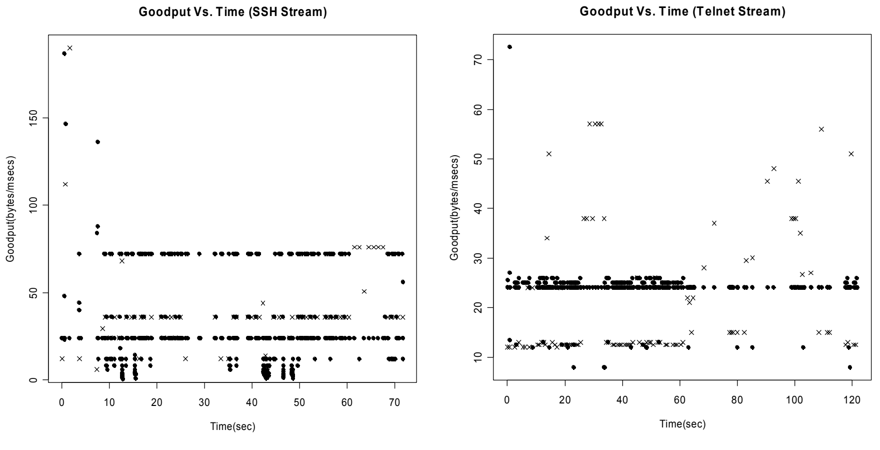
Table 1 shows the output of the TCP conversation dictionary for the three applications, running across the wireless networks. Application sessions are identified by their well-known service ports; the client ports, conversation set-up times and packets belonging to each flow can then be deduced.

Figure 6 shows the real goodput in bytes per millisecond experienced by the different TCP applications. Dots indicate values for packets from the client to the server, and ‘x-points’ represent values for packets from the server to the client. Goodput is defined as the number of payload bytes received per unit time in each direction, as opposed to throughput, which measures the number of packets sent, regardless of their eventual fate [16].

It is not envisaged that in-line measurements should instrument and be carried along with every single packet across the network. Rather, they should serve as an additional measurement tool deployed where and when required to perform service-oriented measurements along Internet paths. For this reason, target application domains need to be carefully addressed and evaluated.

Service Port	*Client Port*	*Conv. Setup Time (μsec)*	*Packets*	*Completness*
22	32790	3166	638	True
23	32789	3102	751	True

Table 1: Conversation Dictionary output for the measured TCP applications over the wireless networks



(a) **(b)**

Figure 6: Goodput measured for (a) SSH and (b) Telnet over the wireless topology.

For example, for bulk TCP transfers, adding measurement options to the data packets could cause MTU violation, since the protocol maximises its performance (throughput) by sending the maximum possible packets sizes for data. It should be up to the application and/or transport mechanism to judge, according to its requirements, whether extra space for measurement instrumentation should be accommodated or not.

5. Conclusions and Future Work

This paper introduced and demonstrated a novel in-line measurements technique, which exploits the IPv6 extension headers to assess the performance properties of application flows across the Internet. The benefits of in-line over traditional measurement techniques were highlighted. The design of a prototype implementation has been also presented, showing how the technique can be provisionally realised and exploited to measure unidirectional and bi-directional traffic properties. Measurement results have been presented, documenting properties of UDP and TCP data, over IPv6 topologies.

It is anticipated that the technique would be found useful by network operators in assessing traffic performance, and in particular as part of their network operations and management. The diversity and complexity of Internet traffic implies that network operations need to include measurement-based techniques such as the one introduced in this paper in order to monitor and control (literally, to manage) the behaviour of the network and the services running on it.

Further work will concentrate on enhancing the scope and size of measurements and performance metrics. Different measurement TLVs are currently being evaluated. Experimentation also concentrates on filtering and sampling mechanisms to address scalability and overhead issues, and the applicability of in-line measurements for particular application domains and network operations.

ACKNOWLEDGMENTS

We are grateful to Agilent Technologies for the support of Dimitrios Pezaros' work through an industrial fellowship. Stefan Schmid, Steven Simpson, Joe Finney and Andrew Scott provided invaluable support throughout this work. The authors are deeply indebted to Barry Rowlingson for advising on the use of statistical packages.

References

[1]. Active Measurement Project (AMP), http://watt.nlanr.net//active/intro.html

[2]. Apsidorf, J., Claffy, K., C., Thompson, K., Wilder, R., OC3MON: Flexible, affordable, high performance statistics collection, in *Proc. of the seventh annual conference of the Internet society (INET'97)*, Kuala Lumpur, Malaysia, 1997

[3]. Claffy, K., C., Miller, G., Thompson, K., The nature of the beast: recent traffic measurements from an internet backbone, in *Proc. of the eighth annual conference of the Internet society (INET'98)*, Geneva, Switzerland, 1998

[4]. Cisco IOS NetFlow, http://www.cisco.com/warp/public/732/Tech/nmp/index.shtml

[5]. Deering, S., Hinden, R., Internet Protocol version 6 (IPv6) specification, IETF, IPNG Working Group, RFC 2460, December 1998

[6]. Downey, A., B., Using pathchar to estimate Internet link characteristics, in *Proc. of ACM SIGCOMM'99*, Cambridge, MA, pp. 241-250, September 1999

[7]. Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F., Deriving traffic demands for operational IP networks: methodology and experience, in *Proc. of ACM SIGCOMM'00*, Stockholm, Sweden, 2000

[8]. Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F., Design and deployment of a passive monitoring infrastructure, in *Proc. of Passive and Active Measurement Workshop (PAM2001)*, Amsterdam, NL, 2001

[9]. Georgatos, F., Gruber, F., Karrenberg, D., Santcroos, M., Susanj, A., Uijterwaal, H., Wilhelm, R., Providing active measurements as a regular service for ISP's, in *Proc. of Passive and Active Measurement Workshop (PAM2001)*, Amsterdam, NL, 2001

[10]. IETF IPPM Working Group,

http://www.ietf.org/html.charters/ippm-charter.html

[11]. Jain, R., Routhier, S., Packet trains – measurements and a new model for computer network traffic, *IEEE Journal on Selected Areas in Communications*, Vol.4, No.6, September 1986, pp. 986-995

[12]. Kalidindi, S., Zekauskas, M., J., Surveyor: an infrastructure for internet performance measurement, in *Proc. of the ninth Annual International Networking Conference (INET'99)*, San Jose, CA, 1999

[13]. Matthews, W., Cottrell, L., The PingER project: active internet performance monitoring for the HENP community, *IEEE Communications Magazine*, May 2000

[14]. Mills, D., Internet time synchronisation: the Network Time Protocol, *IEEE Trans. Communications*, 39(10):1482-1493, October 1991

[15]. Multi-Router Traffic Grapher, http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

[16]. Padhye, J., Firoiu, V., Towsley, D., Kurose, J., Modelling TCP throughput: a simple model and its empirical validation, in *Proc. of ACM SIGCOMM'98*, Vancouver, Canada, 1998

[17]. Pásztor, A., Veicht, D., PC based precision timing without GPS, in *Proc. of ACM SIGMETRICS 2002*, Marina Del Rey, California, 2002

[18]. RIPE NCC Test Traffic Measurements (TTM) Project, http://www.ripe.net/ripencc/mem-services/ttm/index.html

[19]. VideoLAN Open Source Streaming Solution Homepage, http://www.videolan.org