Service Quality Measurements for IPv6 Inter-networks

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

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

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

Abstract— **Measurement-based performance evaluation of network traffic is becoming very important, especially for networks trying to provide differentiated levels of service quality to the different application flows. The non-identical response of flows to the different types of network-imposed performance degradation raises the need for ubiquitous measurement mechanisms, able to measure numerous performance properties, and being equally applicable to different applications and transports. This paper presents a new measurement mechanism, facilitated by the steady introduction of IPv6 in network nodes and hosts, which exploits native features of the protocol to provide support for performance measurements at the network (IP) layer. IPv6 Extension Headers have been used to carry the triggers involving the measurement activity and the measurement data *in-line* with the payload data itself, providing a high level of probability that the behaviour of the real user traffic flows is observed. End-to-end one-way delay, jitter, loss, and throughput have been measured for applications operating on top of both reliable and unreliable transports, over different-capacity IPv6 network configurations. We conclude that this technique could form the basis for future Internet measurements that can be dynamically deployed where and when required in a multi-service IP environment.**

Keywords- active/passive measurements, extension headers, Internet Protocol version 6 (IPv6), performance metrics

I. Introduction

A major challenge for network operators and service providers is the ability to provide a stable service with consistently predictable performance characteristics to include high network reliability, low delay, low jitter and high availability [1]. Especially when considering introducing preferential treatment to some arbitrary amount of network traffic, as opposed to all traffic being treated as best-effort, then the necessary mechanisms need to be in place to provide feedback over the different service quality characteristics experienced by the different traffic flows.

Current measurement techniques do not provide a generalised framework for performance evaluation, because either their focus is on measuring specific properties of certain types of traffic, or their scope is limited within the boundaries of single administrative domains. In addition, they are quite tightly coupled with particular measurement infrastructures and applications, performing specific measurement tasks.

Recent discussions within IRTF's Internet Measurement Research Group [2] have shown a clear interest in defining another Internet measurement protocol to try and measure performance and path properties that current deployments do not address, due to their inherent design or due to the way the network itself treats the associated measurement traffic.

In this paper, a service-oriented measurement mechanism is presented, namely *in-line*, IPv6-based measurements; the mechanism is using native features of the IPv6 protocol to provide for accurate, transparent, low-overhead and reliable traffic flow measurements for any type of traffic. At the same time it surpasses the associated difficulties and shortcomings of existing measurement protocols and systems, by not being coupled to particular applications, transport protocols, or network topologies.

Due to the evolution of the Internet Protocol as a universal transport mechanism there is an increasing aggregation of multi-service traffic onto IP networks carrying various equivalence classes of network flows. As a consequence, the different Quality of Service (QoS) requirements and different sensitivities to potential service degradation of these equivalence classes of traffic make timely and accurate measurement of actual network flow QoS essential. Measurements revealing the real service experienced by user traffic can prove valuable for long and short term network design decisions, dynamic traffic engineering, as well as for Service Level Agreement (SLA) negotiation and policing, and advanced network and service management.

Dimitrios Pezaros' work has been funded by Agilent Technologies through an industrial fellowship

This paper is organized as follows: Section 2 selectively discusses measurement-related work, identifying strengths and weaknesses of existing measurement approaches and techniques. Section 3 introduces the concept of in-line, IPv6-based measurements, highlighting the major relevance of this work to theory and practice. Section 4 describes the implementation of an In-line measurement prototype, where measurement functionality has been built into systems' network protocol stack. Section 5 presents and discusses the results taken from instrumenting with measurement data different types of traffic, carried over different-capacity IPv6 configurations. Numerous service quality characteristics in the form of performance metrics have been documented. Section 6 contains concluding remarks and highlights future directions for this work.

II. Related Work

Existing measurement techniques and infrastructures fall into two main categories, namely active and passive. The primary focus of active and passive measurements has been on measuring end-to-end path properties, and on characterising network traffic mainly through backbone link monitoring, respectively. Both active and passive streams exhibit particular limitations reflecting the fact that they are independent and non-native mechanisms (protocols, tools, etc.) operating on top of or in parallel with the actual network traffic, while trying to measure and assess its characteristics.

A. Active Measurements

Active techniques are based on the concept of injecting additional traffic with known characteristics into the network to test particular attributes of a service [3]. Part of the measurement process is the generation/injection of this synthetic traffic into the network. This traffic usually consists of ICMP packets [4], [5] or variations [6], UDP packets [5], [7], [8], but also of dedicated measurement protocol traffic like NLANR's Internet Performance Measurement Protocol (IPMP) [9]. Active measurements projects have been focusing on measuring properties of end-to-end network paths between instrumented systems by implementing metrics defined within IETF's IP Performance Metrics (IPPM) Working Group, such as delay and loss, and also by defining (and implementing) higher-level metrics such as unreachability and quiescence [4].

Two major limitations of active measurements are first that the performance properties measured by the injected traffic do not necessarily reflect the performance experienced by operational network traffic; and second, the additional traffic associated with active measurements obviously impacts the network and may itself be a factor in measuring a poorer performance than the network would otherwise deliver. Also, periodic sampling and packet injection used by some architecture may fail to observe periodic network events [4]. Therefore, although active techniques are service oriented, their dependence on certain types of traffic and applications limits their ability to provide a generalized measurement framework.

B. Passive Measurements

Passive measurements provide for highly accurate results by observing and analysing real traffic on a link without disruption to the service. They are usually deployed within single administrative domains and are mainly concerned with monitoring link utilisation [10], [11], deriving network traffic demands [12] and characterising backbone traffic [13]. They are particularly useful in gathering measurements of user traffic at a single observation point in the network but may also be deployed for two-point measurements by employing additional correlation techniques.

Passive measurements suffer from continuous increases in network speeds that make the amount of measurement data, which often needs to be transferred across the monitored links substantial; there is consequential difficulty in targeting specific services and identifying packets that belong to the same flows. Also, correlating samples collected at two distinct observation points to yield one-way flow measures can be a challenging task, consuming significant resources and network bandwidth [11]. Hence passive techniques are more oriented towards network engineering and investigation of the overall traffic characteristics of backbone topologies, rather than measuring the service quality properties of the different application flows carried over the Internet.

III. In-line, IPv6-based Measurements

The main motivation behind defining and implementing in-line, IPv6-based measurements is the lack of a generic service-quality-oriented measurement mechanism to be equally applicable to different types of traffic, independent of particular transport protocols and applications; at the same time the mechanism should mainly offer accurate, flexible and transparent performance measurements of the real -rather than special-purpose- network traffic.

In-line measurements are an intrinsically multi-point mechanism. It makes use of piggybacked data, containing measurement-related indicators such as timestamps and packet counters, carried in a special header within regular user packets. The presence of such special headers acts as a trigger that invokes the measurement activity.

Underpinning this functionality is the programmable and extensible specification of IPv6 [14] and the use of IPv6 extension headers, which allow such a measurement mechanism to be deployed independently from the applications and from the intermediate network nodes between the two (or more) instrumented points.

Hence, the contribution of IPv6-based, in-line measurements is not only the fact that the measurement data is carried along within the user packets per se. Studies of carrying timestamps and tags within packets have been conducted in the past, targeted at specific traffic (e.g. modified ICMP) [6].

The main novelty of this technique is threefold:

- The user traffic carries the minimal measurement information, virtually guaranteeing that the measurements reflect the performance experienced by the payload data; at the same time

- the underlying network (IPv6) layer provides the space for measurement extensions to be defined as part of the protocol itself, hence making the mechanism applicable to any type of traffic.
- Selective processing of the measurement extensions only where and when required (at specific instrumented nodes/hosts) is guaranteed by the IPv6 specification.

Optional functionality in IPv6 has been designed as a set of *Extension Headers*. These are inserted between the main IPv6 header and the upper layer (transport) header. The *Destination Options* extension header is specified to carry Type-Length-Value (TLV)-encoded options to be processed only by the packet's destination. This can either be the ultimate destination or a pre-specified intermediate destination. Destination options are not examined by the nodes along the packet's route, and therefore intermediate nodes forward the packets normally, irrespective of whether they carry such options or not. Furthermore, the IPv6 specification provides space for additional TLV-encoded options to be defined as long as they conform to certain structural and alignment requirements.

The latter two properties of the destination options header make IPv6 particularly suitable for measurement extensions to be designed efficiently, cost-effectively, and be a native part of the protocol's operation without influencing the core of the forwarding mechanism.

This also demonstrates an important difference in the design between IPv4 and IPv6: in IPv4, there is a fixed set of options defined to be carried within the main IPv4 header. Moreover, options are processed en route, and all IPv4 modules need to implement all the standardised options. "What is optional is their transmission in any particular datagram, not their implementation" (RFC 791). Not only the need for standardisation would impose immense difficulties in designing new options for IPv4, but also the need for all IPv4 modules to support them, would make their seamless operation virtually impossible.

Moreover, the necessity of option processing en route, would alter the forwarding behaviour of packets, and hence even if measurement extensions were designed for IPv4, the measurement results would not reflect the performance experienced by the rest of the (un-instrumented) traffic.

Indeed, studies on IPv4 option support in the Internet have shown that IPv4 packets carrying options would either be discarded en route (high volume routers don't have the power to process options), or they would be put on different processing queues (fast/slow path) [15].

The incremental and selective option processing of IPv6 eliminates these concerns, and henceforth makes it feasible for a native measurement mechanism to be defined and targeted to any traffic type carried over IPv6.

As part of this work, numerous measurement destination options TLVs have been defined to carry timestamps, counters and trace information as well as other associated measurement system traffic. For a detailed description of the particular encoding of these options and their insertion between the main IPv6 header and the upper layer headers, the reader is referred to [16]. Figure 1 shows two destination options TLVs designed to measure one-way delay (a) and one-way loss (b). The rest of the paper will focus on the implementation of and the experimental results gathered from these particular measurement options.

The one-way delay TLV has been used to measure delay between two points along an Internet path, as well as more synthetic time-related parameters such as jitter and throughput. Its operation is based on the source and destination hosts inserting departure and arrival timestamps, respectively, with microsecond accuracy. The one-way loss TLV provides a means of IP-based sequencing of packets by having a source node inserting incremental sequence counters to packets belonging to the same flows, which are then observed at the destination. Packet loss as well as out of order delivery can be measured.

Figure 2 shows the different places along an end-to-end Internet path where in-line, IPv6-based measurements can be deployed; measuring properties of the real end-to-end path between hosts <A> and <D> needs the measurement functionality (extension header processing) be implemented at the two end-systems. In this scenario, nodes <B> and <C> will not process the measurement extension header options, rather they will forward the packet as if no extensions were present. An alternative measurement scenario is to have measurements conducted between host <A> and node <C>. In this case the encoding of the extension headers itself would explicitly specify node <C> as an intermediate destination to examine the optional header along the path from <A> to <D> [14], [16]. Upon arrival at node <C> the packet header would be inspected and the measurement data observed/amended; the packet would then be forwarded to the ultimate destination <D>.

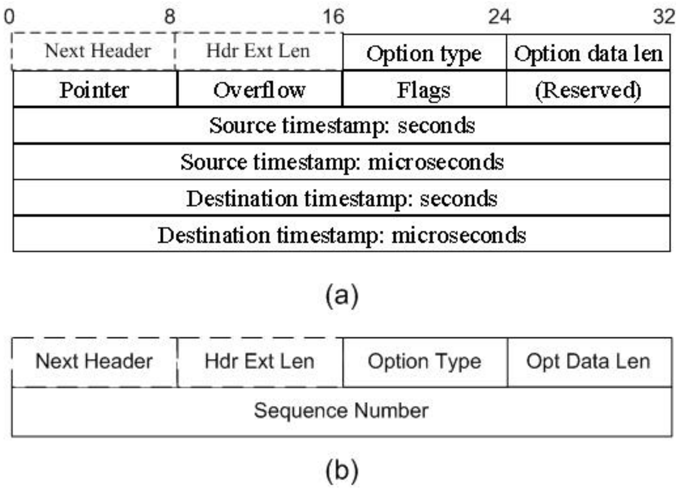

Figure 1. (a) One-way delay and (b) one-way loss TLVs

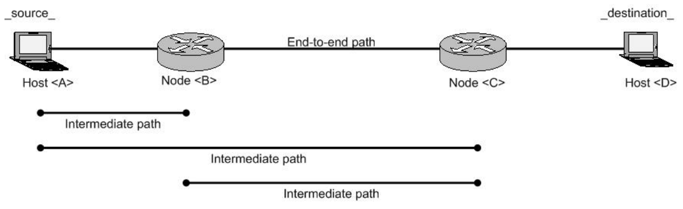

Figure 2. The different notions of end-to-end

Again, under this scenario, node <B> need not know anything about the measurements; it forwards packets normally. Equivalent principles hold true when measuring performance between host <A> and node <B>, and between nodes <B> and <C>. This demonstrates the important ability of the mechanism to enable end-to-end as well as intermediate path (e.g. between edge nodes of an Autonomous System) measurements.

In contrast to active measurements that are applied to certain types of traffic, two-point in-line measurements can be applied to the majority of traffic flows, with a small additional systematic delay and marginally larger packet headers. 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.At the same time, 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, if the addition of measurement data were to cause packet fragmentation and reassembly, then any resulting measurements might be highly questionable, especially in relation to delay. Also, the addition of measurement data to time- or length-sensitive packets, such as Voice-over-IP packets, may have detrimental consequences and, in any case, be difficult because of header compression techniques.

In-line techniques should thus be considered as another potentially favourable approach to measurement rather than as an omnipotent answer appropriate for all situations.

IV. PROTOTYPE IMPLEMENTATION

As was emphasised earlier in the paper, an important property of in-line measurements is the decoupling between the measurement mechanism that instruments packets with measurement data, and the higher-level architecture/tool that is used to gather and analyse the measurements. In this section the implementation details of the measurement instrumentation mechanism and the measurement system prototypes are treated separately.

A. In-line Measurement Modules

At a system level, the required functionality for instrumenting traffic with measurement-related information can be implemented in a variety of different ways, depending on the other roles of the system in question (e.g. network node, host), and its available resources and processing capabilities. A network element might implement separate measurements modules for different measurements that can be dynamically loaded onto a line interface as dynamically reconfigurable hardware, as software, or as a hybrid combination of both options.

For the purposes of the prototype, measurement modules are implemented as dynamically Loadable Kernel Modules (LKMs) running on Linux systems (kernels 2.4.x), that can be linked with a running Linux kernel at runtime. The modules provide insertion, manipulation, extraction and removal of IPv6 measurement destination options TLVs, as an integral part of the kernel's implementation of the network protocol stack. These typically provide better processing performance compared to external, user-space applications.

Separate modules have been used not only for the different types of measurement extension headers (e.g. figure 1 (a) and (b)), but also for the distinguished functionality between the "source" and the "destination" systems in an experiment.

Each measurement is implemented as a pair of source/destination modules that -once loaded- attach themselves to the IP6_output and IP6_input queues, respectively. The responsibility of the source module is to select packets to be instrumented before they are forwarded to the network interface; create the destination options extension header and the appropriate measurement TLV, and fill the corresponding fields with local measurement data. Then the module inserts the extension header between the main IPv6 header and the payload data, and returns the packet under the control of the IP6_output routine. If a packet to be instrumented already contains another extension header, then the source module creates and inserts the measurement TLV according to the IPv6 extension headers' processing rules [14].

The destination module observes packets as these come in from the network interface. If a packet contains a destination options header carrying the appropriate optional TLV, then the module, depending on the experiment, stores the measurement data, and sometimes adds further local measurement indicators to the TLV fields. Then the measurement header is extracted and the packet is passed back to the IP6_input routine which resumes processing control.

Figure 3 shows the operations of source and destination measurement modules when measurements are taken between hosts A and B. It can also be seen how the two modules can simultaneously operate on each system, hence providing for bi-directional measurements of traffic flows.

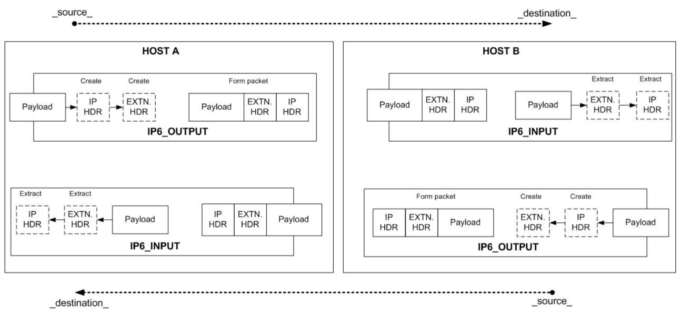

Figure 3. Source and destination measurement modules also showing bi-directional operation

The measurement modules use character devices in order to read data from and write data to the user space, through the input/output control (ioctl) system calls. The source module reads configuration commands from the user space, which include filtering and sampling parameters, so that traffic instrumentation can be tuned to be as fine or as coarse-grained as necessary. Filtering can be based on the source and destination addresses and ports, on transport protocol, traffic class, and on flow label values. Sampling can be configured to determine whether the module should instrument all packets (that match filtering criteria), 1-in-N packets, or act at a specific temporal sampling rate. The destination module writes to the character device the measurement-related information of packets belonging to instrumented flows. For each packet, this information contains copies of the main IPv6 header, the destination options measurement TLV, and the transport layer header of the packet, all encapsulated in the form of an opaque test-object. Test-objects can then be accessed from a user space measurement application for further data analysis.

Referring to the examples of the one-way delay and loss TLVs (figure 1), the corresponding source and destination modules perform the following measurement-specific tasks: the one-way delay source module inserts a 64-bit timestamp to the TLV marking the departure time from the system, and the destination module inserts an arrival timestamp to the TLV. The one-way loss source module builds a database of active flows (based on the source/destination addresses and ports tuple) since the time it was loaded. It then starts inserting a 32-bit incremental sequence number to the TLV of every departing packet that matches the filtering criteria. After insertion, the sequence number of each flow is stored in the corresponding flow record in the database. The module resets the counters of a flow if it has been inactive for more than a configurable number of seconds. The one-way loss destination module simply copies the extension header (together with the IPv6 and the transport layer header) to the user-space-accessible character device, without making any amendments to the loss TLV.

B. *Measurement tools*

The test-objects created by the destination module for a particular measurement are then read by a measurement system that decodes the data and extracts values of interest. Each object contains encoded information specifying the particular measurement class it belongs to (e.g. delay, loss, or trace measurement), and depending on that class, trace-files are created containing measurement-specific indicators (timestamps, counters) as well as network and transport layer values, such as e.g. packet payload length or TCP window size.

Further off-line and computational analysis of the measurement data is being conducted by protocol-specific engines that re-build the flows out of the individual packets and reveal their performance characteristics in the form of first (e.g. delay, loss) and second order (e.g. throughput, jitter) performance metrics. The tool also offers the possibility of combining unidirectional flow traces to build the entire TCP streams, and examine the relevance in performance between the forward the reverse paths of the connections. This is particularly applicable when investigating sessions where the data characteristics massively differ in each direction (e.g. file transfers), or when measuring the performance of traffic over asymmetric paths.

V. Experimental Results

The operation of the in-line measurements prototype has been initially demonstrated by instrumenting representative types of traffic end-to-end, over a variety of different-capacity network configurations. Future experimentation will also consider conducting measurements between selective intermediate nodes, transparently to the end-systems, essentially demonstrating a scenario of measurements being taken between edge nodes of an ISP.

In this section time-related and loss performance results are presented for bulk transfers running over TCP, and for streaming video traffic running over UDP.

Figure 4 shows an overview of the overall connectivity between the different IPv6 networks at Lancaster University and the rest of the Internet. Measurements were conducted over a variety of topologies, including private and public wired 100 Mb/s Ethernet networks, IEEE 802.11b operational topologies, and 512/256 kb/s ADSL links. Wireless connectivity is provided at major parts of the University campus and also at a number of places in the city centre. End-to-end IPv6 connectivity for the residential broadband links is provided through IPv6-in-IPv4 tunnels to/from the university's network.

In order to demonstrate the applicability of the technique and its inter-operation with current protocol stack implementations and higher level services, we chose to instrument traffic of existing off-the-shelf applications; a FTP server [17] generated bulk TCP transfers, and a streaming video client-server pair [18] was used to create the UDP streams.

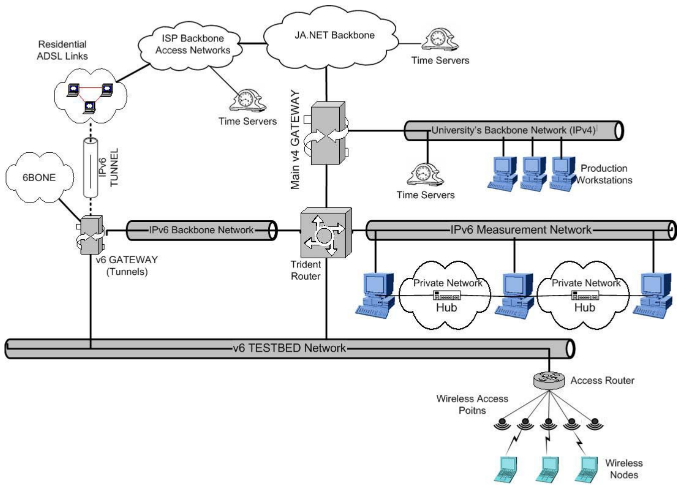

Figure 4. Overall IP connectivity at Lancaster

Measurements were coarse-grained, instrumenting every packet of the flows of interest with no selective sampling. Filtering has been based on the transport protocol, piggybacking with measurement data "all TCP" or "all UDP" traffic, respectively. For the time-related measurements the systems synchronised using NTP [19] with a stratum 1 server. The offset of each system from the NTP server was included in the timestamp calculation and it was empirically validated that this was sufficient to always produce positive delays, even over 100 Mb/s delay-free private wired Ethernet topologies. Of course, for test cases revealing delays at the order of hundreds of milliseconds, a NTP synchronisation error/offset of a few milliseconds is not significantly affecting the accuracy of the measurements.

The remainder of this section focuses on the presentation and discussion of measurement results taken over the experimental IPv6 configurations, and demonstrating the operation of the one-way delay and loss TLVs (figure 1) to assess a variety of performance metrics.

End-to-end paths over the IEEE 802.11b networks as well as over the 512/256 kb/s ADSL links have been considered as a set of more attractive topologies to measure, mainly for two reasons. First, they are relatively low capacity, operational configurations where there is a significant amount of cross-traffic present that creates higher contention along the path, and influences the performance of the instrumented flows. On the wireless side, traffic traverses a 4-hop path over university's wireless networks into the IPv6 Testbed network, and then routed within the private IPv6 measurement networks; more interestingly, IPv6 traffic to/from the residential ADSL links is tunnelled in IPv4 and traverses a 15-hop path over UK ISPs' backbone topologies, before it is de-capsulated at the tunnel end-points and routed within the IPv6 configurations (figure 4). Second, these topologies provide a better approximation of access networks and of more complex end-to-end paths, where there is no administrative access to network nodes and no control over the path followed by the traffic. Two-point, in-line measurements are used to provide an insight on the performance properties experienced by user traffic.

The overhead of the in-line technique was 20 and 8 bytes per instrumented packet for the time-related and loss measurements, respectively. For TCP that uses the full space (indicated by the MSS option) to carry data, this meant that more packets were needed (and more time) to complete the data transfer. For UDP, the overhead was mainly the additional processing delay of adding the extension header to each measured packet.

A. TCP Measurements

Bulk file transfers were generated over the wireless and ADSL topologies, and measurements were collected for both the forward (data) and reverse (ack) path by running simultaneously the source and destination measurement modules at the two end-systems of interest, as illustrated in figure 3. For TCP, the performance properties of interest were the absolute application goodput and packet loss. In this sense, 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 [20]. Measurements show the number of payload data bytes (excluding all network and transport layer headers) received at the destination over the end-to-end, one-way transit delay of each packet. The measurement system had to guarantee that packet fragmentation due to the addition of measurement data was strictly avoided, since fragmentation would not only impact the performance experienced by the traffic flows, but could also have cause instrumented packets to be treated differently by the network nodes than the rest of the traffic.

For TCP traffic this was achieved by decreasing the TCP Maximum Segment Size (MSS) option to accommodate space for the extension headers within the MTU boundaries. The in-line measurement source modules altered the MSS option in the header of TCP SYN messages and re-calculated the checksum of the packet. This approach is similar to the one used within the Netfilter project [21]. Another approach would have been to inform the applications about the space requirements of the in-line measurements, but this would be more tightly-coupled with the systems' kernel implementation.

Figure 5 is a kernel density plot [22] showing the estimation of the probability density function (pdf) of the application goodput (both forward and reverse paths) for three different bulk transfers over the wireless network. There is a clear similarity in the goodput of the three different transfers shown. The upper plot suggests that the application goodput of the data path mainly takes values between 8 and 15 KB/s; the lower plot shows the reverse path that has some peaks mainly around 4, 5 and 6 KB/s. The data paths of the transfers follow a distribution, such that the goodput tends to get larger values than the mode (around 10 KB/s). The shape of the lower plot reflects the more "static nature" of the goodput in the reverse path, which consists of small-sized acknowledgements that do not make optimal use of the bandwidth capacity. Reference [23] explains how TCP throughput, and hence goodput too, is directly proportional to the packet size.

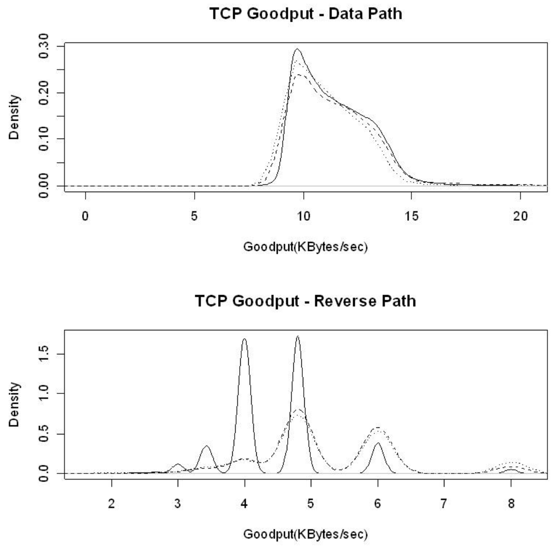

Figure 5. Density plots of TCP Goodput over the wireless networks

Figure 6 shows the density estimation of the distribution of application goodput of the data path for the ADSL downlink (512 kb/s) and uplink (256 kb/s). It can be seen that the data path of the ADSL line in each direction follows a very similar distribution, with similar peaks. The fact that the downlink graph (solid line) is shifted to the right (i.e. larger goodput values) can among others, also illustrate the difference in bandwidth capacity in each direction. Again, the distribution here implies values larger than the mode, but these seem much smaller and less smooth than what is shown at the upper plot of figure 5. These spikes at the tails of the distribution can be attributed to the high contention (50:1) of the path and the significant presence of competing traffic. Also, the IPv6-in-IPv4 tunnelling might have influenced the performance experienced by the flows; native IPv6 mode could have resulted in more optimistic goodput figures for the ADSL paths. However the difference of the peak values between the upper plot of figure 5 and figure 6 does not reflect the difference neither in capacity nor in utilisation between the two networks (11 Mb/s vs. 512 kb/s) in addition to the tunnelling that took place for experiments over the ADSL. Hence, signal strength of the wireless network might have also been an influencing factor of performance degradation.

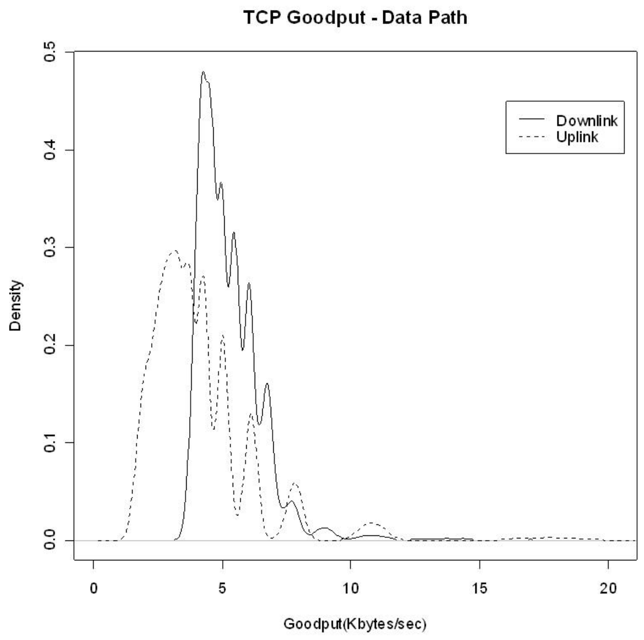

Figure 6. Density plots of TCP Goodput over the ADSL links

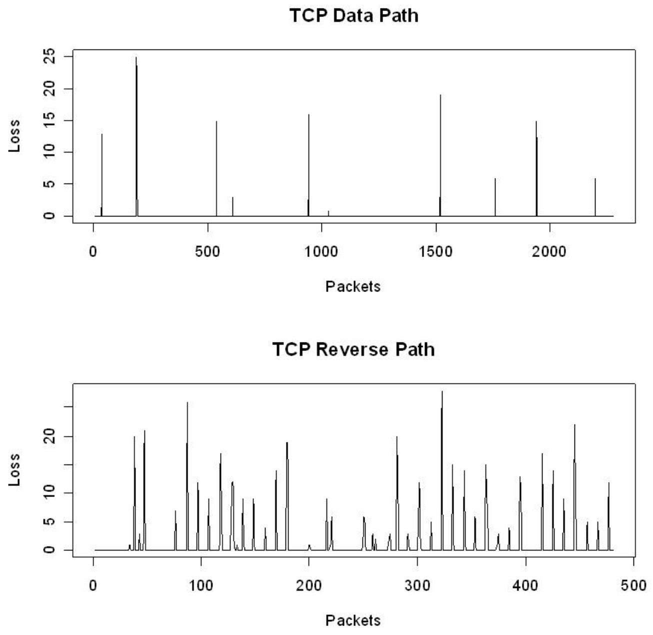

Figure 7. TCP Loss over the wireless networks

Figure 7 shows an interesting case of packet loss experienced by a TCP bulk transfer over the wireless network, where there were burst occurrences of lost packets, mainly in the reverse path. Figure 8 shows a more normal case of packet loss over the ADSL link, where occasionally some packets are lost/dropped. Packet loss is being measured as the difference in the sequence numbers in the packet loss TLV (figure 1), between successive packets received by the destination measurement module.

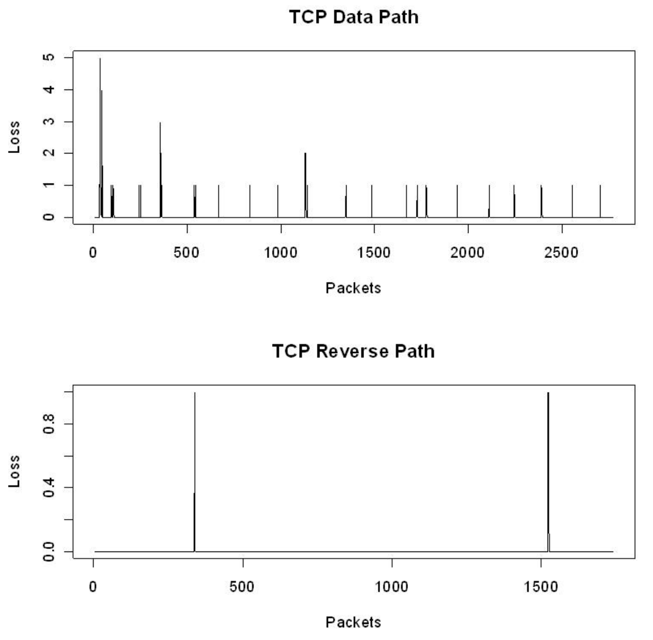

Figure 8. TCP Loss over the ADSL link

B. UDP Measurements

UDP traffic consisted of packets adequately-sized (1348 bytes) to accommodate the measurement information. For the video streaming sessions, measurements of end-to-end one-way transit delay, inter-packet delay variation (jitter), and packet loss are presented, as a set of more appealing performance properties for UDP traffic.

Figure 9 shows the density estimation of the distribution of the one-way delay experienced over the wireless network and the ADSL down and up-links, respectively. The video stream over the higher capacity 11 Mbps wireless link experiences relatively small delay values having a mean of 19.25 milliseconds and the 3rd quantile of the distribution is 23 ms, hence 75% of the observations fall below this value.

Over the ADSL link the delays are clearly much larger with means 679.6 and 1558 ms for the down and up-link respectively. Between the two asymmetric ADSL channels the difference in the delays is considerably large, reflecting much better the difference in bandwidth (512 vs. 256 kb/s) than the TCP goodput experiments did.

Figure 10 shows UDP Jitter density estimation for the three different-capacity channels. Over the wireless link, jitter is mainly concentrated on small positive values, although some sudden decreases in delay can also be seen. Although the mean is 0.00172 ms, which is influenced by the large negative values, and the standard deviation is approximately 6.56, the plot itself shows that most values are not very widely-spread.

This can imply that the delays experienced by the UDP flow tend to slightly increase with time, being influenced by the delays experienced from the previous packets. On the ADSL side the means are not much larger, but values are clearly more spread, with standard deviations approximately 36.8 for the downlink, 43.8 for the uplink, respectively.

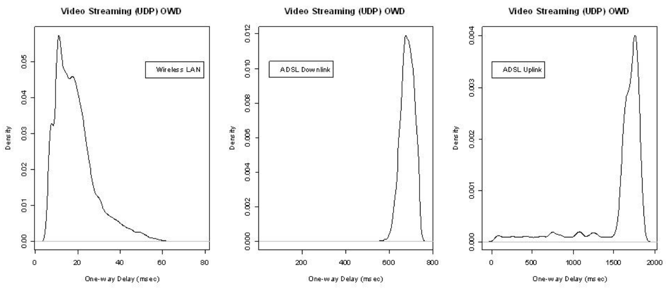

Figure 9. Density plots of the end-to-end one-way delay for UDP over the wireless and the ADSL networks

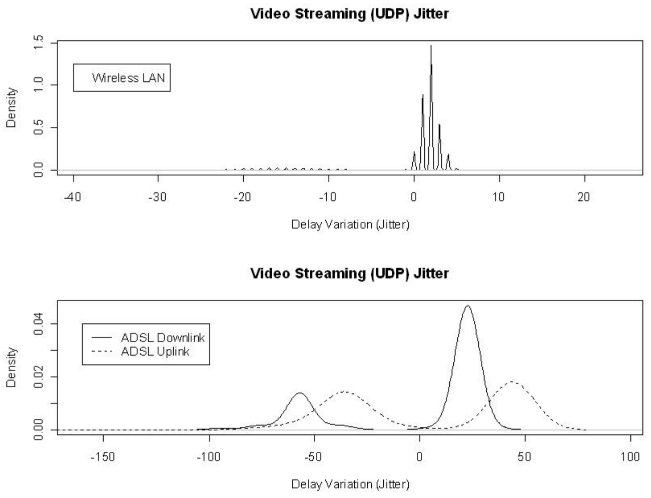

Figure 10. Density plots of Jitter for the UDP streams over the wireless and ADSL links

A very interesting situation with severe packet loss is shown in figure 11, where 1.5 Mb/s MPEG video was streamed over the wireless and ADSL topologies. Frequent bursts of up to 60 lost packets have been experienced during a video experiment over the wireless link. It can be implied that packet loss almost follows a pattern of tens of packets being lost approximately every 15 hundred packets being streamed, with only a few occasional losses in the order of 1-5 packets.

An even more severe situation arose during the same video experiment over the ADSL uplink, where bursts of losses occasionally reached a few thousands and more frequently a few hundreds of packets. It is indicative that from the more than four thousand packets sent from the server, only around 360 packets were received by the client. Needless to say that at the application front-end there was no video playback, although the end-system kept receiving packets occasionally.

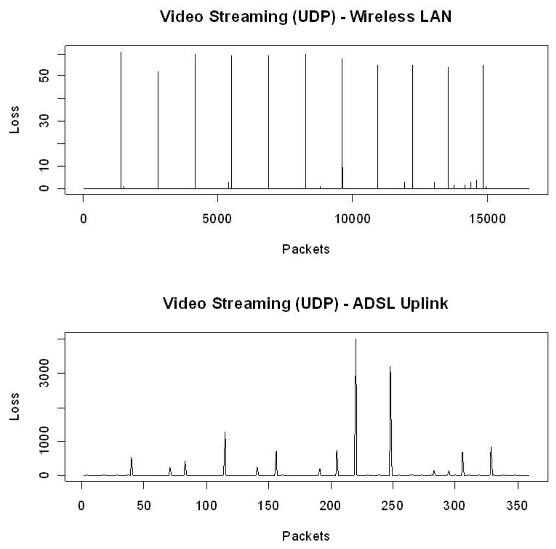

Figure 11. Packet loss of the UDP stream over the wireless and ADSL links

VI. Conclusions

This paper presented a novel measurement technique for assessing the performance properties of any type of traffic carried over IPv6 networks. Whereas IPv4 is the current ubiquitously-deployed version of the Internet Protocol, it is likely that IPv6 will increasingly be adopted as its advantages of providing scalable, transparent, and manageable services are revealed. This work investigated one such property of the protocol that can be exploited by all network users to offer feedback over the service quality properties of the various network flows. Among other benefits of this approach, end-users can verify practically the service that their traffic experiences, network operators can understand how the different flows react to specific events and configurations, and ISPs can evaluate the practical considerations of deploying and running multi-service networks.

Future work will concentrate on performing enhanced sets of end-to-end and intermediate path experiments and identifying particular target areas where in-line techniques can prove most useful. Filtering and sampling mechanisms will be investigated to address overhead issues; the systematic processing overhead, as well as scalability of the technique with respect to the number of instrumented flows will be examined.

Acknowledgment

The authors would like to express their very special gratitude to Theodore Kypraios from the Department of Mathematics and Statistics, Lancaster University, for his kind assistance and comments on the presentation of the measurement results.

References

[1] Ferguson, P., Huston, G., Quality of Service on the Internet: Fact, Fiction, or Compromise?, in Proceedings of the eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, July 21-24 1998

[2] IRTF IMRG Discussion Group, discussion on the need for defining a new Internet Measurement Protocol (IMP), available at: http://login.caida.org/pipermail/imrg-dist/2003-November/000129.html

[3] Paxson, V., Towards a Framework for Defining Internet Performance Metrics, in Proceedings of the sixth Annual Conference of the Internet Society (INET'96), Montreal, Canada, June 24-28 1996

[4] Matthews, W., Cottrell, L., The PingER project: Active Internet Performance Monitoring for the HENP Community, IEEE Communications Magazine, Volume 38, Issue 5, May 2000, pp. 130-136

[5] Kalidindi, S., Zekauskas, M., J., Surveyor: An Infrastructure for Internet Performance Measurements, in Proceedings of the ninth Annual Conference of the Internet Society (INET'99) INET'99, San Jose, California, June 22-25 1999

[6] Claffy, K., C., Polyzos, G., C., Braun, H., Measurement Considerations for Assessing Unidirectional Latencies, Internetworking, Research and Experience, Volume 4, No 2, John Wiley & Sons, 1993, pp. 121 - 132

[7] 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 Proceedings of Passive and Active Measurement Workshop (PAM2001), Amsterdam, NL, April 23-24 2001

[8] RIPE NCC Test Traffic Measurements Project Homepage: http://www.ripe.net/ripencc/mem-services/ttm/index.html

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

[10] Apsidorf, J., Claffy, K., C., Thompson, K., Wilder, R., OC3MON: Flexible, Affordable, High Performance Statistics Collection, in Proceedings of the seventh Annual Conference of the Internet Society (INET'97), Kuala Lumpur, Malaysia, June 24-27 1997

[11] Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F., Design and Deployment of a Passive Monitoring Infrastructure, in Proceedings of Passive and Active Measurement Workshop (PAM2001), Amsterdam, NL, April 23-24 2001

[12] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F., Deriving Traffic Demands For Operational IP Networks: Methodology And Experience, in Proceedings of ACM SIGCOMM'00, Stockholm, Sweden, August 28 – September 1 2000

[13] Claffy, K., C., Miller, G., Thompson, K., The Nature Of The Beast: Recent Traffic Measurements From An Internet Backbone in Proceedings of the eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, July 21-24 1998

[14] Deering, S., Hinden, R., Internet Protocol Version 6 (IPv6) Specification, IETF, IPNG Working Group, RFC 2460, December 1998

[15] IRTF IMRG Discussion Group, discussion on the different treatment of IPv4 packets containing options at the network nodes, available at: http://login.caida.org/pipermail/imrg-dist/2003-September/000122.html

[16] Pezaros, D., P., Hutchison, D., Garcia, F., J., Gardner, R., D., Sventek, J., S., In-line Service Measurements: An IPv6-based Framework for Traffic Evaluation and Network Operations, in Proceedings of IEEE/IFIP Network Operations and Managements Symposium (NOMS 2004), Seoul, Korea, April 19-23, 2004

[17] PureFTPd file transfer server, available at: http://www.pureftpd.org/

[18] VideoLAN streaming solution, "STREAMING: Overview of the VideoLAN streaming solution", available at: http://www.videolan.org/streaming, 18th April 2004 [date accessed]

[19] Mills, D., Internet time synchronisation: the Network Time Protocol, IEEE Transaction on Communications, Volume 39, Issue 1, October 1991, pp. 1482-1493

[20] Padhye, J., Firoiu, V., Towsley, D., Kurose, J., Modeling TCP Throughput: A Simple Model and its Empirical Validation, in Proceedings of ACM SIGCOMM'98, Vancouver, British Columbia, August 31-September 4 1998

[21] Netfilter Project Homepage, http://www.netfilter.org/

[22] Silverman, B., W., Density Estimation for Statistical Data Analysis, Chapman & Hall/CRC, April 1986, ISBN: 0412246201

[23] Mathis, M., Semske, J., Mahdavi, J., Ott, T., The Macroscopic Behavior Of The TCP Congestion Avoidance Algorithm, ACM SIGCOMM Computer Communications Review, volume 27, number 3, July 1997