QUALITY OF SERVICE MONITORING AND ACCOUNTING

M. E. Culverhouse and B. V. Ghita

Network Research Group, University of Plymouth, Plymouth, United Kingdom
info@network-research-group.org

ABSTRACT

Network monitoring has been a topical research area for the past two decades, receiving substantial attention both from the research community when pursuing novel inference and observation alternatives, as well as and from the network administrators, when implementing the proposed solutions. In parallel, consumers demands evolved from service availability towards Quality of Service, imposing new requirements for monitoring service level agreements, as well as linking network usage and quality accounting. Various solutions exist on the market for traffic accounting but, increasingly, network performance is added to availability and traffic levels when assessing a network provisioning service. This paper proposes an enhancement to one of the leading traffic accounting architectures available - Netflow. The proposed expanded architecture allows Netflow-based monitoring to offer not only accounting information, but also performance related statistics in order to provide a perspective of the network health. The paper provides the framework for expanding the components of current monitoring architecture towards including performance related statistics, whilst also considering the aggregation and management of the data collected as an end result.

KEYWORDS

NETWORK MONITORING, NETFLOW, QOS, TCP PERFORMANCE, TRAFFIC ACCOUNTING

1. INTRODUCTION

Computer networks have evolved into large complex structures that can no longer be implemented and left to function with minimal human intervention. Networks are being introduced into environments where previously a single computer would suffice; homes and small offices are benefiting from the advantages of sharing resources and operating network services as well as sharing Internet connectivity. In parallel, end-users become more and more dependant on the network from an operational perspective, transforming Internet service provisioning from an alternative to the only choice, with more and more applications sharing the available bandwidth. Due to these changes, the underlying aggregation and core networks provided by the Internet Service Providers (ISP) have to adapt to the demand of their services by the consumers, to confidently deliver the required Quality of Service (QoS). Enterprise networks have faced a similar evolution - as the use of real-time and time-critical applications has increased, enterprise networks saw more and more the necessity to add quality provisioning measurers to ensure appropriate perceived performance.

In order for enterprise and core networks to ensure the level of their service, efforts have been made to address bandwidth management and QoS into network design. While management of a network has become a fundamental requirement, it is impossible to separate management from monitoring. A network administrator must have a complete view of the network performance and status at any point in time (including historical trends) in order to efficiently manage the network resources. Monitoring the performance of a network is not sufficient for an administrator to confidently manage a network. Accounting the behaviour of a network in terms of traffic types and usage patterns allow for informed adjustments to be made to the network configuration. Accountability of a network also has advantages from a security perspective, suspicious network usage can be identified and provide information leading to those responsible.

Network monitoring can introduce a number of obstacles for a network manager to consider prior to proposing and implementing an appropriate solution. Many traffic monitoring tools offer one of two extremes - either a generalised view of the network offering very limited details on network devices or, at the other end of the scale, a highly-granular detailed account of network activity which does not provide any means for aggregation. This paper proposes a solution that will combine the benefits of the above alternatives, providing both the ability to observe the overall network behaviour, with broad and general statistics, whilst offering the ability to examine more detailed statistics about the traffic flows on a network. The proposed approach also addresses a critical issue in historical monitoring – data storage; the collected information on the network would allow the generation of trends over considerably long periods of time without excessive storage requirements.

2. SUMMARY OF NETWORK MONITORING

The network monitoring area received substantial attention over the history of the Internet, its focus migrating from local monitoring over to Internet-wide measurement infrastructure and back to observing end-networks. From an output perspective, monitoring methods also migrated from providing results for overall networks towards per-flow analysis. The methods over time led to a wide range of implementations, currently providing a reasonably detailed view of the network topology, traffic, performance, and trends.

2.1 Traditional, Current and Future Trends of Network Monitoring

Historically simple tools such as *ping* (Packet Internet Groper) and *traceroute* were used to actively monitor a network; providing “one-time” results that would give an indication of a networks performance. This kind of network monitoring is no longer sufficient to provide an administrator enough detail as already mentioned. Taking one step further, Simple Network Management Protocol (SNMP) enables the exchange of management information about network devices. and provides figures based on interface utilisation, bandwidth statistics and interface status (uptime). However SNMP does not offer a network administrator the ability to collect highly detailed statistics regarding traffic on the network and so cannot be used for accounting purposes, it also does little to characterise traffic [1]. From a different perspective, packet capturing tools can collect and store all passing traffic in a packet trace file for later analysis. *tcpdump* is probably the most well-known such tool, using pcaplib [2] to capture the packets and producing a text output with the packet details either to the display or to a file for later analysis. *tcptrace* [3] provides an in-depth view of a TCP connection, able to extract detailed network, data transfer, and connection performance information from packet traces. *tcptrace* produces text results, but can be linked to external tools for generating graphical output of the inferred performance characteristics.

Due to their market leading position and the simplicity and flexibility of the concept, Cisco imposed the Netflow architecture as the preferred alternative for network monitoring and accounting. . Netflow uses the *flow* as the atomic unit for monitoring for which the architecture gathers associated information; a Netflow collector defines a flow as unidirectional stream of IP datagrams sharing the same values for Source and Destination address and port numbers, IP Protocol type, Type of Service and input interface [1]. From a generic network monitoring perspective, Netflow offers a level of monitoring detail between SNMP and packet parsing – per-flow data is far more detailed than the generic SNMP statistics, but the limited amount of information stored provides off-line analysis with less detail than a *pcaplib* packet trace. Being a proprietary but open standard [4] has facilitated the development of Netflow-based implementations by both competitor network equipment manufacturers and also by software developers. A Netflow system typically has four essential components: Flow caching, Data Export, Flow Collection, and Data Analysis [5]. There is an abundance of commercial and open-source alternatives that provide one or more of the key component functions. The complexity of tools available varies greatly, for example flowd [6] is an open-source Netflow collector for BSD/Unix/Linux platforms. More often tools perform more than just one function, Netflow Analyzer from AdventNet [7] is one such tool for both Microsoft Windows and Unix/Linux platforms that can collect exported Netflow data, correlate it and produce graphs and reports to provide a visual interpretation of the Network. Flow-tools [8] is a collection of Netflow tools for Unix/Linux platforms which currently consist of 17 different tools for Netflow collection, manipulation and analysis. A Netflow (or Netflow-based) system can provide adequate network accountability, providing relevant statistics primarily for user and application monitoring and billing services, but also for observing traffic analysis and network monitoring. A number of journal and conference papers have been published relating to the use of Netflow-based monitoring techniques, amongst them Kim *et al* [9] and Sommer *et al* [10] highlight the benefits to a Netflow-based monitoring system over alternative methods. Zang *et al* [11] and Estan *et al* [12] pay particular attention to improving and optimising Netflow implementations by proposing modification to Netflow's operation in terms of the proportion of network traffic measured and also the rate at which traffic is sampled, both evidencing the flexible nature of Netflow-based monitoring.

As mentioned, storage of the collected data represents another potential difficulty when considering a network monitoring system. When collecting data from networks, the amount of information collected can become an issue if not appropriately managed. Collected flows are typically stored in a database and subject to further aggregation automatically by the program or tool used to manage the collected data. Stager [13] is an example of a Netflow-based architecture, performing aggregation of the stored data over time, reducing the level of detail over time to maintain control over the storage (and ultimately reducing the cost) of the monitoring system. In addition, Stager performs collecting and reporting on flows, providing a single-host monitoring system alternative. A valuable feature of using open-source tools is the ability to bind tools together to form a tailored system. RRDTool [14] offers an alternative to choosing a tool such as Stager. RRDTool is an open-source data-logging and real-time graphing application which maintains a backend database (The Round Robin Database – RRD.). RRDTool can be bound to data collection tools using Perl and/or Python, of particular interest to network monitoring, FlowScan from [15] is an open-source Linux tool that provides a complete solution for the analysis of collected Netflow-based data by binding itself to a Netflow collection tool and also RRDTool, utilising the RRD and web-browser visualisation features of RRDTool.

Netflow-based software implementations represent a convenient alternative and will produce comprehensive results for the tasks of accounting and monitoring. Softflowd [16] is an open-source software implemented network traffic analyser capable of caching and exporting versions 5 and 9 of Cisco Netflow data. Softflowd operates on BSD/Unix/Linux platforms, providing capability to passively monitor a network interface or use a dump file as its data input source. At the other end of the spectrum, per-flow monitoring and analysis tools, such as tcptrace, offer the ability to identify performance related issues such as excessive delay, availability, congestion patterns, and packet loss.

3. EXTENDING NETFLOW

In a typical scenario, routers or enterprise switches will include functionality for caching and exporting Netflow packets; this is facilitated through a hardware module that can be integrated in high-end Cisco routers and switches [17] which does the associated processing and avoids overloading the router CPU from any additional processing. The actual format and content of packets exporting Netflow information changed over time as the technology progressed through several versions. The latest version, Netflow v9 [18], forms the basis for the IETF IP Flow Information Export (IPFIX) working group standard [18]. A key feature of version 9 is its extensible nature which permits a high level of customisation to the statistics being cached and exported. The Netflow packet header informs the collector of the export version and allows the collector and analyser to expect the correct packet structure.

This paper proposes an architecture that combines the TCP/IP statistics from tcptrace into an extended Netflow packet structure. This enhancement to the Netflow packet structure requires that each component of the network monitoring system architecture is modified to support the new Netflow data structure. Figure 1 presents a block diagram of the proposed architecture, this block diagram is used later in this paper to illustrate the integration of the specific tools that are being used by the end design.

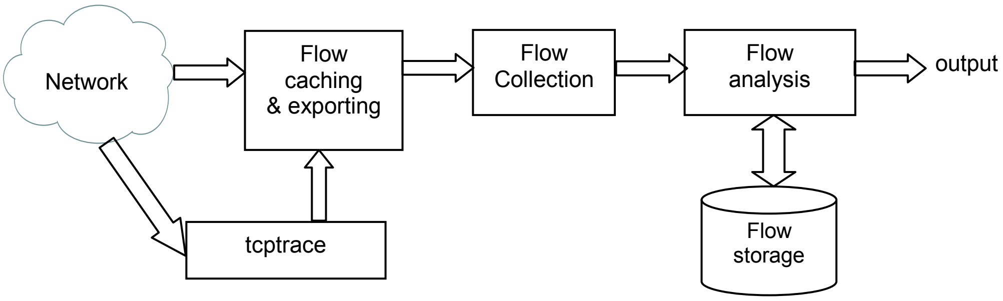

Figure 1: Proposed Component Architecture.

One primary advantage that Netflow has over other monitoring architectures is its balance between the provided detail and the amount of storage data for each flow/connection. Netflow aggregates only the required parameters for each flow transiting the monitoring point. A typical scenario would require exporting the data from the collector to a storage device, but, with the storage placed topologically near the collector, the architecture will not produce additional traffic. This potential problem can be completely removed if a single node will act as the Netflow exporter, analyser and reporter, which is the recommended implementation for the proposed architecture.

3.1 Desired Performance Statistics

Monitoring network performance through TCP analysis involves a substantial amount of inferred events, implying assumptions, and possible estimation errors. *tcptrace* can produce over 70 different statistics for each analysed TCP connection [19], all of them relevant for examination of the network performance, including details about the evolution of the data transfer, inferred network parameters, as well as the correctness of the connection. For the purpose of reducing the amount of stored data while facilitating a comprehensive analysis on each TCP connection the combined Netflow-tcptrace solution will include the following parameters:

- connection parameters: *SYN/FIN packets sent, Initial Window bytes*
- network parameters: *Retransmitted data packets, Retransmitted data bytes, Out of Order packets, RTT Samples, Minimum, Maximum and Average RTT values*
- data transfer parameters*, , Maximum Idle Time, Throughput, Minimum, Maximum, Mean, and Weighted Average Congestion Window Values.*

The addition of these parameters allows Netflow to complete its generic traffic accounting capabilities with functionality for recording and storing quality of service parameters. As mentioned, *tcptrace* will provide comprehensive statistics for each connection, but further analysis should determine the accuracy of results [20] when applying the method in a particular scenario. Regardless of the amount of inference, a key factor in when evaluating the accuracy of the measurements is the placement of the monitor in relation to the endpoints. Having a single monitoring terminal introduces discrepancies when determining the location of problems on the network; for instance a packet may pass the monitor and subsequently get dropped between the monitor and the receiving terminal, in this case the monitor may be calculating a RTT value and include unexpected delay incurred whilst the sender resends the packet following a timer expiration at the sender end. Distributed monitoring would clearly offer a better solution when accuracy of the statistics is paramount, the benefits of a distributed monitoring system are highlighted by Habib *et al* [21], but adopting a distributed design would render the solution either too expensive or unusable when monitoring the traffic between an end network and the Internet.

3.2 tcptrace output synchronisation issues

The success of the resulting Netflow-tcptrace performance accounting architecture heavily depends on the ability to combine statistics from tcptrace into the correct corresponding Netflow record. Typically, tcptrace operates on offline packet traces (most of them using pcap-based [2] tools such as *tcpdump*); as the only outputs are the network and connection parameter files, tcptrace does not have any self-synchronisation issues. However, Netflow caching and exporting by default operates in real-time. Two alternatives can be employed to reach the desired synchronisation of the two sets of outputs. A possible solution is to have tcptrace running in real-time mode, with current TCP connections stored in a temporary memory location until they are closed (gracefully or not). When the caching of a flow is complete and ready to be exported, the new system will query and retrieve the appropriate performance statistics from tcptrace. At this stage the combination of the TCP performance statistics and the Netflow data is achieved prior to exporting the Netflow-tcptrace flow. Alternatively to this approach, tcptrace and the software caching and exporting the Netflow data can operate in a timed interval manner, where the two applications regularly receive and process packet traces. The preferred alternative is to modify tcptrace so that it will maintain a rolling-record of TCP connections, which will then be fed to the Netflow exporter. The reason behind working with this model relates to the compromise between long-lived connections and real-time delivery of information. At one extreme, short duration of each packet trace may lead to fragmenting the long-lived connection. This would result in a connection appearing over multiple flows and incorrectly illustrating the data transfer behaviour. At the other extreme, extending the duration of each trace will lead to non-real-time results and possible storage-and-retrieval issues.

4. DESIGN AND IMPLEMENTATION

The block architecture of the proposed solution detailed on the component function required at each stage of the Netflow system and the integration of tcptrace into the system, followed by an overview the wide range of available alternatives to perform one or more functions of a Netflow-based system. This section describes the specific tools that will be responsible for performing each of these functions in the new monitoring system.

4.1 Flow caching, exporting, collection and analysis.

Softflowd was chosen as the preferred alternative for the caching and exporting of the Netflow-based packets in the proposed model; the main reason for this choice was the flexibility of the implementation and its Linux availability. Softflowd is to be modified to request the performance statistics collected by tcptrace and combine these parameters with the collected monitoring and accounting based details. To eliminate the potential problem of transmitting the collected data back across the network if the system's elements were distributed, all the components will be operating on a single monitoring node. *FlowCapture* [8] taken from the *FlowTools* collection will provide the function of Netflow collection and storage-management to the system, automatically compressing, storing and managing disk space for the collected flows. *FlowScan* from the Caida open-source toolkit [22] will be used to provide analysis and visualisation of the collected flows through the use of its bindings to RRDTool, which allows for web-browser access to the RRDTool graphs and reports.

4.2 System Implementation

Figure 2 builds upon the Figure 1 to include the specific tools that are to contribute towards the complete system.

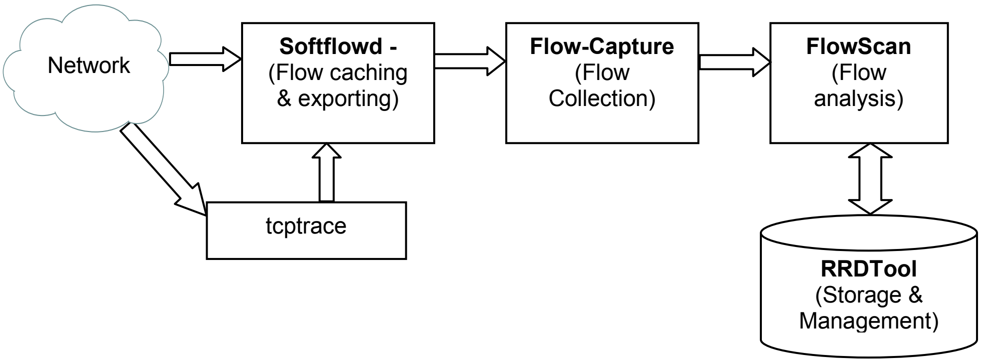

Figure 2: Design of the new system.

Softflowd is shown above extracting the TCP connection statistics from the cache of tcptrace prior to aggregating and exporting the Netflow packets. As Figure 2 highlights, there is more than just softflowd handling the Netflow data, this introduces the requirement that FlowCapture and FlowScan will also face modification to accommodate the irregular Netflow packets being exported by softflowd. Due to the new Netflow packet structure Flow-capture and FlowScan require modification to prevent rejection of the received packets. Figure 3 presents the structure of the customised Netflow packet and includes the TCP performance characteristics which will be provided per connection by the system (indicated by a '*').

<table>
<tr><td colspan="4">Source IP Address</td><td colspan="4">Destination IP Address</td><td colspan="4">Next Hop IP Address</td></tr>
<tr><td colspan="6">Input Interface Index</td><td colspan="6">Output Interface Index</td></tr>
<tr><td colspan="6">Packets</td><td colspan="6">Bytes</td></tr>
<tr><td colspan="6">Start Time Of Flow</td><td colspan="6">End Time Of Flow</td></tr>
<tr><td colspan="6">Source Port</td><td colspan="6">Destination Port</td></tr>
<tr><td colspan="3">Pad</td><td colspan="3">TCP Flags</td><td colspan="3">IP Protocol</td><td colspan="3">TOS</td></tr>
<tr><td colspan="6">Source As</td><td colspan="6">Destination As</td></tr>
<tr><td colspan="4">Src Netmask Length</td><td colspan="4">Dst Netmask Length</td><td colspan="4">Padding</td></tr>
<tr><td colspan="4">Retransmitted Data Packets*</td><td colspan="4">Retransmitted Data Bytes*</td><td colspan="4">Out Of Order Packets*</td></tr>
<tr><td colspan="3">SYN/FIN *</td><td colspan="3">Initial Window*</td><td colspan="3">Max Idletime*</td><td colspan="3">Throughput*</td></tr>
<tr><td colspan="3">RTT Samples*</td><td colspan="3">RTT Minimum*</td><td colspan="3">RTT Maximum*</td><td colspan="3">RTT Average*</td></tr>
<tr><td colspan="3"><i>Max CWND*</i></td><td colspan="3">Min. CWND*</td><td colspan="3">Mean CWND*</td><td colspan="3">Weighted Avg. CWND*</td></tr>
</table>

Figure 3: Proposed Extended NetFlow v5 Flow Record Information

4.3 Further improving network performance reporting

Whilst the proposed system above presents a valid enhancement of a traditional Netflow-based collection system, some additional modifications can be made to the calculation methods used by tcptrace in order to further optimise the systems effectiveness.

4.3.1 Round Trip Time calculation

Calculating the RTT can, if purely derived from timestamp differences on corresponding data segments, be subject to errors. tcptrace obeys Karn's algorithm [23] to prevent performing RTT calculations on retransmitted packets, however tcptrace cannot accurately determine which packets are lost based purely on interarrival timestamp analysis, this introduces errors in its estimations. The accurate measurement of RTT values is not a subject to have been ignored in the past; RFC1323 [24] details an extension to the TCP header to include a *TCP timestamp* option. TCP timestamps may be enabled on Internet hosts, but this is a not compulsory, according to RFC 1122 [25]. TCP timestamp options provide a method of increasing the precision of RTT measurements, stamping TCP segments when leaving the sender, the TCP receiver increments the timestamp value reflective on the delay incurred during transmission, and sends the increased value back along with the ACK. Ramadas comments on that tcptrace does not implement timestamp options in its calculations of the RTT [23]. It is beyond the scope of this study to recommend changes to tcptrace in order to include this facility, despite its associated accuracy increase. This is because, apart from the changes in tcptrace, such analysis requires enabling the TCP timestamps option at both end-points. By default, recent Microsoft Windows operating systems do implement the TCP timestamp options, but have them disabled by default [26]. As a result, a Windows TCP server will respond to data segments with attached TCP timestamps, but default Windows TCP clients would not include TCP timestamps when initiating a connection. As a generic conclusion, TCP timestamps are currently enabled only in Unix/Linux systems [27]. This clearly limits the number of TCP connections that can be observed to have Timestamp fields in use and so this contributes towards the reasoning behind not extending tcptrace to support this option. A secondary but less decisive factor towards not implementing timestamp options is from a security perspective. There have been recent reports that malformed TCP timestamps can result in the end-host ultimately crashing [28]. Whilst patches and resolutions have been released to counteract the problems faced with vulnerable TCP timestamps implementations, the generic associated concern remains if any future developments of the proposed tool were to include such functionality.

4.3.2 Congestion Window Calculation

In addition to RTT statistics tcptrace also exports another set of values of high importance for the data transfer, detailing the size of the congestion window during a connection. The choice of tcptrace is to provide reliable congestion window statistics, which are based on the number of unacknowledged packets currently in a connection. Whilst this technique provides a reliable method of estimating how well a connection utilises the available bandwidth, an alternative approach exists, based on the previously calculated RTT values. A monitor can compare the inter-arrival timestamps for all data packets within a flow with the RTT values in order to estimate the number of packets within the congestion window [29]. All packets being transmitted within the same congestion window (or *train*, as defined by Allman and Paxson [20]) have timestamps within close succession of each other. The first segment within a new *train* of packets will have an interarrival delay comparable to the RTT value. . This increased difference between timestamp values is caused by the sender having to wait for an acknowledgment from the receiver before commencing the transmission of the next congestion window train. A limitation of this mechanism is that, for long-lived connections, the sender is likely to make full use of the available bandwidth and render this method unusable; the majority of the flows however will get reliable estimates via this inference. This paper will recommend enhancing tcptrace to support the above method of measuring the congestion window to improve upon the "unacknowledged packet" method that currently operates to further add to the usefulness of the proposed system.

5. CONCLUSIONS

This paper proposed a monitoring architecture solution which allows integration of existing traffic accounting methods with network performance and data transfer analysis. Cisco Netflow provided an excellent basis to build upon through its passive monitoring style and tcptrace provided the required support for TCP analysis. The combination of the two leads to a single node caching, exporting, collection, and analysis.

In addition to describing the processes followed when combining the output of Softflowd and tcptrace, this paper further develops the system by introducing enhancements to the operation of tcptrace. This paper proposes modification to tcptrace to achieve two main goals, the first to incorporate real-time collection and analysis of TCP connections, and the second to improve the accuracy of tcptrace's estimation of the Congestion window by use of timestamp derived RTT values.

Further work on the described architecture will concentrate on considering multi-node monitoring abilities as well as adapting the TCP analysis to the characteristics of the monitored environment. It is envisaged that the resulting methods will cover the whole spectrum of requirements for network managers – from information about overall usage to evaluation of network performance.

6. REFERENCES

[1] Cisco (2004) Cisco IT Case Study – Netflow, http://www.cisco.com/warp/public/732/Tech/nmp/docs/cisco_it_case_study_netflow.pdf , (2nd February 2007)

[2] TCPDUMP, (2007) – TCPDUMP Public Repository, http://www.tcpdump.org, (20th February 2007)

[3] Ostermann, S. (2003) Shaun Ostermann, tcptrace, http://www.tcptrace.org, (20th February 2007)

[4] Apoapsis (2004) Apoapsis Limited, 2004, Cisco NetFlow Briefing Paper, http://www.netusage.net/pdfs/Netflow_Briefing_Paper_2_2.pdf, (January 30th, 2007)

[5] McGibbon (2002) Terry McGibbon, 2002, NetFlow in the Enterprise Network, Apoapsis Consulting, http://www.apoapsis.com/pdf/ Netflow_Briefing_Paper.pdf, (January 31st 2007)

[6] Miller, D. (2006) Damien Miller, flowd – fast, secure and flexible Netflow collector, http://www.mindrot.org/projects/flowd/, (February 20th 2007)

[7] AdventNet (2007) Bandwith Monitoring, Bandwidth Reporting, Bandwidth Meter: NetFlow Analyzer, http://origin.manageengine.adventnet.com/products/netflow/index.html, (February 20th 2007)

[8] Fullmer, M. (2007) Splintered.net Website, Flow-tools, http://www.splintered.net/sw/flow-tools/docs/flow-tools.html, (February 20th 2007)

[9] Kim, M.S, Won, Y. J and Hong, J.W (2006), Characteristic analysis of internet traffic from the perspective of flows, *Computer Communications*, Volume 29, Issue 10, 19th June 2006, Pages 1639 – 1652

[10] Sommer, R. and Feldmann, A. (2002) Netflow: information loss or win?, *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement.*

[11] Zang, H. and Nucci, A. (2005) Optimal Netflow Deployment in IP Networks, Proceedings of the *19th International Teletraffic Congress, 2005*

[12] Estan, C., Keys, K., Moore, D. and Varghese, G. (2004) Building a better Netflow, *Proceedings of the 2004 conference on Applications, technologies, architectures and protocols for computer communications.*

[13] Uninett (2006), UNINETT: Testnett – Stager:News, http://software.uninett.no/stager/, (20th February 2007)

[14] Oetiker, T. (2006) RRDTool website, http://oss.oetiker.ch/rrdtool/, (20th February 2007)

[15] Dave Plonka, "FlowScan: A network traffic flow reporting and visualization tool," in *Proceedings of the Fourteenth Systems Administration Conference (LISA-00)*, 2000, pp. 305–318.

[16] Miller, D. (2006) Damien Miller, Softflowd – Fast Software Netflow Probe, http://www.mindrot.org/projects/softflowd/, (20th February 2007)

[17] Cisco (2007) Netflow Feature Card II – Catalyst 5000 Series Switch, http://www.cisco.com/en/US/products/hw/switches/ps679/products_data_sheet09186a0080092607.html, (20th February 2007)

[18] Claise, B. (2004) Cisco Systems NetFlow Services Export Version 9, Request for Comments 3954.

[19] Ostermann, S. (2003) Shaun Ostermann, TCPTRACE Manual, http://jarok.cs.ohiou.edu/software/tcptrace/manual/index.html, (20th February 2007)

[20] Allman, M. and Paxson, V. (1999), "On Estimating End-to-End Network Path Properties", SIGCOMM 99, Cambridge, Massachusetts, September 1999

[21] Habib, A., Khan, M. and Bhargava, B. (2004) Edge-to-edge measurement-based distributed network monitoring, *Computer Networks*, Volume 44, Issue 2, 2004, Pages 211-233

[22] McRobb, Daniel W. McRobb, "cflowd configuration", http://www.caida.org/tools/measurement/cflowd/

[23] Ramadas, M (2003) Tcptrace Users List: Re: tcptrace RTT measurement method, http://irg.cs.ohiou.edu/software/tcptrace/archive/0289.html, (9th February 2007)

[24] Jacobson, V., Braden, R. and Borman, D. (1992) TCP Extensions for High Performance, Request for Comments 1323

[25] Braden, R. (1989) Requirements for Internet Hosts – Communication Layers, Request for Comments 1122

[26] Microsoft (2003), Microsoft Windows Server 2003 TCP/IP Implementation Details - Capabilities and Functionalities, http://technet2.microsoft.com/WindowsServer/en/library/823ca085-8b46-4870-a83e-8032637a87c81033.mspx?mfr=true, (19th February 2007)

[27] P Sarolahti, A Kuznetsov (2002) Congestion Control in Linux TCP*, USENIX 2002 Annual Technical Conference*

[28] United States – Computer Emergency Readiness Team Website, (2005), US-CERT Vulnerability Note VU#637934: https://www.kb.cert.org/vuls/id/637934, (9th February 2007)

[29] Ghita, B - Performance Characterisation of IP Networks, Network Research Group, PhD Thesis, University of Plymouth, UK, 2004