A Characterization of High-Performance Network Monitoring Systems and Workloads

Siddhartha Bunga* and Tilman Wolf†
*Cisco Systems, Inc., San Jose, CA
sbunga@cisco.com
†Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA
wolf@ece.umass.edu

***Abstract*— Measurement and monitoring functionality is widely deployed in the present Internet infrastructure to gather insight into the operation of the network. It is important to obtain a detailed understanding of the system architectures and workloads associated with packet measurement. We present the results of a *quantitative* performance analysis of a variety of existing measurement systems under different workloads. These results give us an understanding of how much system resources are necessary to support measurement in next-generation high-performance networks.**

I. Introduction

Next-generation computer network architectures will need to expand in functionality and flexibility to accommodate a range of novel technologies. Heterogeneous end-systems ranging from extremely resource-constrained devices (e.g., RFID tags) to high-end servers, mobile ad-hoc networks, and sensor networks will need to be supported and managed by the network. Part of the ability to integrate such technologies into the network is the ability to measure and monitor traffic, routers, end-systems, and the state of the network in general.

In this paper, we present an analysis of measurement and monitoring workloads, discuss different hardware architectures for passive measurement and monitoring, and compare system performance for different measurement tasks. The results highlight the system-level issues of different measurement and monitoring approaches. Using a detailed instruction-level analysis of measurement workloads, we can extrapolate the performance requirements of future systems. The conclusions from this work will help us determine the system cost of providing measurement and monitoring capabilities as part of the network infrastructure. In particular, we attempt to answer these questions:

1) What are the inherent processing characteristics of network measurement and monitoring workloads?
2) What are the capabilities of different measurement and monitoring systems in terms of throughput and complexity of measurement tasks?
3) What are the scalability issues when considering monitoring of packet payloads?
4) What are the capabilities and cost of different systems when extrapolating several years into the future?

The reason why studying these topics is relevant and timely, is that the networking community is currently in the early stages of defining the next-generation Internet architecture. New protocols, system designs, and network management mechanisms will be developed as part of this effort. The design decisions regarding these components will heavily depend on what information we can assume to be able to obtain from the network. For example, in the current Internet, TCP congestion control is based on end-system measurement with no information obtained from the network [1]. The decision of designing TCP congestion control was based on the assumption that the network could not provide explicit congestion notification or other measurement and monitoring information. If such capabilities would have been available, different–and possibly better–TCP congestion control mechanisms could have been developed (e.g., ECN-based congestion control [2]). Measurement and monitoring capabilities influence design decisions in other areas of network design:

- Planning and provisioning.
- Run-time monitoring of links (e.g., for routing).
- Security-related monitoring.
- Usage-assessment and billing.

Due to the breadth of these aspects, it is important that we explore what measurement and monitoring capabilities are feasible to implement for a given system cost and complexity. This will allow the community to make the best design choices for the next-generation Internet architecture.

Section II discusses related work. Section III introduces results from an instruction-level analysis of different measurement tasks. Section IV discusses the analysis of three existing passive network measurement systems. Section V discusses the implications of our results on measurement and monitoring in next-generation networks. Finally, Section VI summarizes and concludes this paper.

II. Related Work

Considerable research efforts in the networking community are focused on defining a new Internet architecture that not only solves some of the problems of the current design, but also meets future needs [3]–[5]. The design of today's

Internet has several shortcoming that are expressing themselves in the lack of flexibility, security, and manageability. Most of these problems are direct results of the original design goal to have a simple, packet-switched communication infrastructure, where the network itself was kept relatively simple and provided basic communication between the end-systems [6]. This led to networking protocols, where most of the complexity is implemented on the end-systems (e.g., retransmission of lost packets, congestion control based on round-trip time measurements). Yet, over time, several additional functionalities have been introduced inside the network. For example, Random Early Detection (RED [7]), Network Address Translators (NAT [8]), Firewalls [9], and Intrusion Detection Systems (IDS, e.g., Snort [10]) are all features that extend the capabilities of routers and the network, and cause the Internet architecture to diverge more and more from its original design. We expect that the Internet design will be adapted further to accommodate the expansion of network connectivity to mobile and ad-hoc sensor networks with end-systems having very low processing power. An increasing number of specialized services will be required in order to support the growing number of heterogeneous end-systems and network technologies. Some of these services will require measurement and monitoring capabilities inside the network.

In this paper, we consider measurement and monitoring (MM) to be *passive*; i.e., results are obtained by passively observing user traffic rather than by actively injecting probing traffic. This MM functionality is already used widely in the current Internet and is likely to be part of next-generation network architectures. Typical MM functions are:

- Packet-level monitoring [11]–[14].
- Traffic matrix estimation [15].
- Flow sizes estimation [16], [17].
- Heavy hitter detection [18], [19].
- Anomaly detection [20].
- Frequency analysis and sampling [21], [22].
- Round-trip time estimation [23].

There are several ways of how to implement passive measurement and monitoring capabilities on a network node. We consider the following three: (1) a general-purpose processing system that runs a packet trace collection application (e.g., *libpcap* as used in *tcpdump* [24]), (2) a system with special-purpose trace collection hardware and a general-purpose processing capabilities (e.g., Endace DAG [25]), and (3) a system based on an embedded multiprocessor (e.g., Intel IXP network processor [26]). Details of these architectures are discussed in the following section. Comparisons of general-purpose processors and network processors for implementing networking functions have been performed in the context of TCP processing [27].

There are potentially two ways of operating a passive measurement and monitoring node:

1) Offline MM. In this method all the packets seen on the link are archived and then post-processed by running the application on the trace.
2) Online MM. In this method the packets are processed on transit and required statistics are collected/updated on a per packet basis.

The feasibility of a particular method depends on the nature of the application as well as the infrastructure available. The offline version requires lot of storage requirements and also the processing time is considerable. Also the offline version is not suitable if the application is time sensitive (e.g., anomaly detection). Therefore, we consider online measurement and monitoring in our work. An example of one such system is described in [28].

III. Network Measurement Workload Characterization

As in any computer system, the system workload is a dominating factor in influencing design choices. We show results from an instruction-level analysis of a broad set of measurement applications to illustrate processing requirements. In Section IV, we put these results into the context of specific system implementations.

A. Measurement Workload

To compare the performance of the above systems, we use five benchmark applications that stress different system components:

- **Packet Count**. This application maintains a simple packet counter. This requires that each packet is processed by the I/O interface of the MM system, but the processing demands are very low.
- **Flow Classification**. This application uses the typical 5-tuple of source and destination IP addresses and port numbers and protocol type to classify a packet to a flow. The classification is performed based on previously seen packets, not on pre-determined rules. The data structure used for maintaining flow information is a hash table with a linked list for collision resolution. The processing requirements of flow classification stress the processing component of the system.
- **HTTP Traffic Filtering**. This application simply checks the value of a header field (destination port, in this case) to identify packets that are destined for an HTTP server.
- **IP Address Anonymization**. In some measurement scenarios, it is important to protect the privacy of users when observing traffic. A common technique to remove information about source and destination of a particular packet is to "anonymize" the IP addresses. This is typically done in a prefix-preserving manner, which maintains the subnet relationship between anonymized addresses while hiding the information on the actual addresses. The process of anonymization is typically processing intense and stresses the processing capabilities of a measurement system. We use a version of anonymization that has been optimized for implementation on embedded systems [**?**].
- **Intrusion Detection**. Detecting anomalous traffic is an important function of MM systems in order to detect DDoS attacks, intrusion, malware, etc. This application is

TABLE I

INSTRUCTION-LEVEL ANALYSIS RESULTS (AVERAGES PER PACKET).

Application	Instructions	Packet memory accesses	Non-packet memory accesses
Packet Count	13	0	6
HTTP Traffic Filtering	24	4	5
Flow Classification	153	23	53
IP Address Anonymization	909	21	91
Intrusion Detection	3174	110	1124

based on Snort [10], a commonly used intrusion detection software. The processing requirements are significantly higher than any other MM application in our tests since intrusion detection scans the payload of packets.

These applications not only stress processing capabilities of MM systems, but also their memory subsystems.

B. Instruction-Level Analysis

Both processing and memory access demands present potential bottlenecks in measurement and monitoring systems. In order to explore this issue, we perform a detailed analysis of MM applications at the instruction-level. Using PacketBench [29], we are able to obtain simulation results that yield the number of RISC instructions and memory accesses executed per packet.

When analyzing applications in the networking domain, it is important to distinguish between two types of memory accesses: accesses to packet memory and accesses to non-packet memory [30]. The inherent orientation towards intensive I/O in networking applications requires this separation. Packet data, which changes between different iterations of the same application, cannot easily be cached. Also, on high-performance network processing systems, often packet headers and packet payload are stored in different types of memory (i.e., SRAM for headers and SDRAM for payload). Non-packet memory for program state, on the other hand, is equivalent to what is considered data memory in the computer architecture domain.

Table I shows the average number of instructions, packet memory accesses, and non-packet memory accesses that are performed when processing one packet. The results were derived from processing a trace of 100,000 packets that was collected on our university LAN. The packet sizes of this trace represents typical Internet packet size distribution (i.e., dominated by 40-byte, 1518-byte, and 576-byte packets). As expected, the number of instructions that need to be processed per package differ considerably from 13 for Packet Count to 3174 for Intrusion Detection. Accordingly, the number of both types of memory accesses roughly increases with the complexity of the application. Intrusion Detection is particularly memory intensive as it requires the scanning of packet payloads (thus many packet memory accesses) and comparison with a finite state machine (thus many non-packet memory accesses).

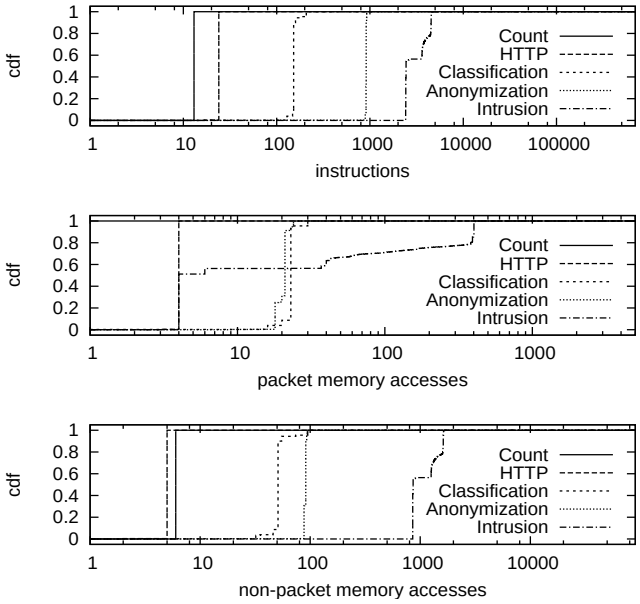


Fig. 1. Cumulative Density Function of Instructions, Packet Memory Accesses, and Non-Packet Memory Accesses.

Clearly, some MM applications contain data-dependent processing sequences. Thus, the averages presented in Table I may not be entirely representative. Figure 1 shows the cumulative density function for the three metrics that we consider. The only significant variation from the average is observed for Intrusion Detection, which is dependent on the size of the packet. Intrusion detection consists of processing the packet header and processing the packet payload. Header processing requires few packet memory accesses (four 32-bit word accesses to get most of the IP header), but lots of instructions ("baseline" of around 2200 instructions). Payload processing creates additional packet memory accesses depending on the packet size and a proportional number of processing instructions (note log scale).

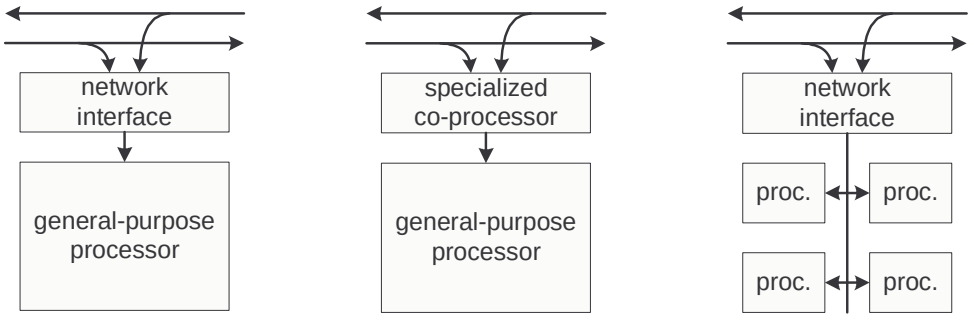
The other applications require exactly the same amount of processing and memory access for a large majority of packets. For applications that only process packet headers, we can therefore use the averages from Table I as a suitable approximation.

IV. NETWORK MEASUREMENT SYSTEMS

The workload results from Section III give some insight into MM applications. In this section, we put these results into context of three types of MM systems. Analyzing these combinations of systems and application workloads, we can obtain performance results that show the impact of I/O and processing demands on different measurement and monitoring architectures.

A. Measurement Systems

When considering measurement and monitoring system, we need to consider the different system components that influence the overall performance. Measurement and monitoring



(a) System A (b) System B (c) System C

Fig. 2. Three Measurement and Monitoring Architectures.

systems need to continuously process traffic and thus need to be able to sustain a certain bandwidth and packet rate. The bottlenecks that limit the amount of traffic that can be handled are:

- **I/O Processing**. The network interface ports have a limitation on the maximum bandwidth they can handle. These limits are often below the nominal link speed because of inefficiencies in the driver and operating system software. If the traffic rates that are fed into the system exceed the I/O processing capabilities, packets need to be queued and eventually dropped.
- **System Interconnect**. The system interconnect that is used to transmit data from the I/O subsystem to the MM processing system can present a bottleneck because of bandwidth limitations.
- **Measurement and Monitoring Processing**. Each packet requires to be processed in order to extract the information that is desired for a particular MM task. The speed and capabilities of the general-purpose processor(s) that perform this task determine the rate at which packets are processed. Clearly, more complex MM tasks require more processing power and thus reduce the rate at which packets can be processed.

Based on these observations, we have selected three system architectures that differ in the way I/O, MM processing, and data transfers are implemented. While we describe each of these systems here in abstract terms, Section IV-B shows that each architecture corresponds to an existing measurement system. The systems we consider are shown in Figure 2 and can be categorized as follows:

- **System A – Conventional Uniprocessor**: This system represents the simplest scenario of a measurement and monitoring setup – a network interface card that collects traffic data connected to a general-purpose processor (e.g., workstation or server processor) that processes the information. The connection between the network interface and the processor is typically the system bus (e.g., PCI bus).
- **System B – Uniprocessor with Specialized Hardware**: This system utilizes specialized hardware (e.g., co-processor, programmable logic devices) to aid in the high-speed capture of packets. Possible functions of this hardware are to pre-process packets and/or more efficiently move them between the interface and the main processor (e.g., capturing the packet header only and performing a DMA transfer).
- **System C – Multiprocessor**: This system uses a multiprocessor architecture to provide more processing power for MM tasks. The partitioning of MM processing can be done such that each processor performs the complete set of MM tasks and packets are processed in parallel. Another partitioning is such that packets are pipelined through the processors, each of which perform a subset of the required processing. There are a variety of tradeoffs between these approaches [31].

From the above list, it is apparent that a combination of specialized capture hardware and a multiprocessor is possible. We are not aware of any currently available MM system that uses that architecture, and thus do not consider this combination any further. Nevertheless, it is a possible design that may be used in future MM systems.

B. Representative Systems

To obtain realistic results, we have chosen the following specific systems to represent each of the three MM architectures shown in Figure 2:

- **System A – PC with NIC**: This setup is a simple server system based on a Intel Pentium and a Gigabit Ethernet network interface. The system uses a conventional PCI bus as interconnect. The processor is a 3.0GHz Pentium 4 with 256MB of RAM running a Linux 2.4.20 kernel.
- **System B – Endace DAG**: This setup is a high-performance, commercial network measurement system that uses an FPGA to handle high-speed packet handling tasks and a host processor for more complex packet processing functions. The FPGA can transfer measurement data into memory without requiring any resources on the host processor. The CPU can thus be dedicated to analyzing packet data without interruption. We use a DAG 3.7GP card placed in the PCI slot of server with a 3.0GHz Pentium processor.
- **System C – Intel IXP2400**: This setup uses a network processor that is programmed to perform measurement

tasks. Network processors are embedded multiprocessor systems-on-a-chip that are optimized for packet processing tasks. By using multiple simple processor cores, packets can be processed in parallel and high data rates can be achieved. Commercial examples of network processors are the Intel IXP systems [26], the Agere FPP [32], and the Hifn PowerNP [33]. For our experiments, we use the Intel IXP2400 with three 1Gbps Ethernet interfaces and 8 embedded processor cores. In our implementation, four processor cores are dedicated to handling the I/O of packets and simple pre-MM tasks and the remaining four processors are available for MM tasks.

We use two testbeds to generate the traffic load for measuring the performance of these systems. For Systems A and B, we aggregate traffic from four Pentium-based workstations through a Gigabit Ethernet switch. For System C, we use a second IXP2400 network processor system to generate traffic that is then fed into the measurement system. The reasons for using two different setups are differences in the physical layer (optical vs. copper) between measurement systems.

C. System Performance Measurement

In this set of experiments, we compare the performance of the different systems for Packet Count and Flow Classification. We inject a stream of minimum size UDP packets at varying data rates. The results are shown in Figure 3. On the x-axis, we show applied throughput measured in kilo packets per seconds (kpps). On the y-axis to the left, we show the throughput that was sustained by the MM system (solid line, also in kpps). On the right y-axis, we show percentage of packets that were dropped (dashed line). We have chosen to present the results in packets per second rather than bits per second since measurement tasks are per-packet and thus the results shown are independent of the packet size.

For System A, the PC with a NIC, the plots show a maximum data rate of 180 kpps. Beyond that rate, packet drops increase considerably and the overall throughput even declines. The reason for this trend is that a standard network interface card cannot cope up with high traffic rates even for a simple application like Packet Count. This is because the goals of an operating system and associated hardware are often in direct contradiction with the needs of measurement. Much of the CPU utilization is wasted in I/O and operating system tasks, and thus the MM application gets a smaller share of the CPU. A similar trend has been observed previously in [34]. This could be improved by using interrupt coalescing techniques, which would amortize interrupt overhead over multiple packets.

A PC with specialized hardware, i.e., the DAG card, does not interrupt the CPU for I/O operations. Thus, the application gets a better share of the CPU. This can be seen in the case of System B and the Packet Count application where there are no packet drops. In this case, System B achieves the highest data rate of all systems because the specialized hardware component can handle measurement activities with trivial processing requirements best. However, in the Flow Classification case, there are packet drops seen above approximately 800 kpps. This is because the system processor becomes a bottleneck due to the processing complexity of the application.

To further explore the issue of CPU utilization, Figure 4 shows the CPU utilization of three different systems/workload scenarios. Figure 4(a) shows System A with the Packet Count application. Due to the lack of hardware support for bringing packets into the system, the majority of CPU load is caused by interrupt handling. This is further aggravated by the scenario in Figure 4(b), where System A is used for Flow Classification. Even though Flow Classification is only slightly more complex in terms of processing (see Table I), considerably more user-space processing for the MM application is necessary. As a result, the packet drop rate increases significantly. These problems can be avoided by System B, which uses specialized hardware to bring packets into the system. As Figure 4(c) shows, practically the entire CPU can be used for MM applications, and packet drops are limited to much higher data rates.

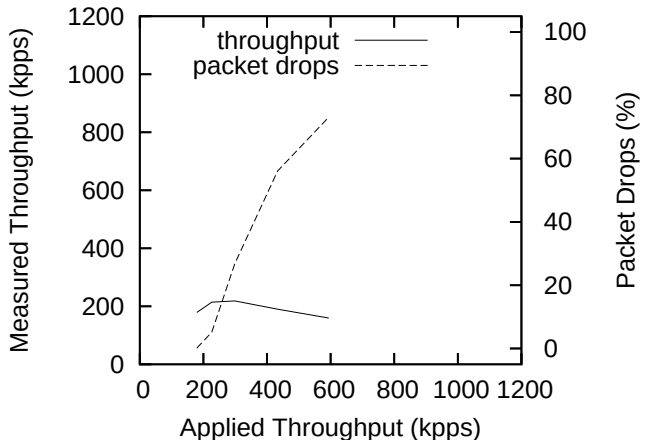
In the case of System C, the network processor with multiple processing engines, there are no packet drops for either Packet Count or Flow Classification (Figures 3(e) and 3(f)). This is because the system takes advantage of its parallel processing capabilities to handle high data rates and complex processing tasks. To further illustrate the impact of parallel processors on measurement, we have performed throughput measurements for System C with different numbers of processors active for handling Flow Classification (in addition to the four processors used for handling packet I/O). Figure 5 shows the performance that is achieved for 1–4 processors. Clearly, a single processor shows the same bottleneck behavior as System A and B did. With two or more processors, this problem is resolved.

Overall, the system with the lowest hardware cost, the conventional uniprocessor, can only handle small amounts of measurement traffic. The multiprocessor performs better, especially when the complexity of processing tasks increases. While the hardware cost of a network processor is comparable to that of a high-end workstation, it is very difficult to program and thus incurs a high initial software development cost. Thus, for simple measurement tasks (e.g., packet counting) at high data rates, the uniprocessor with specialized hardware is the best choice.

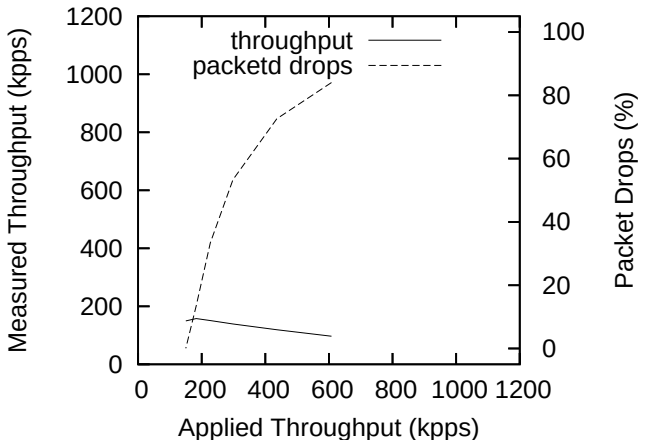
V. Implications for Next-Generation Networks

From the results in Sections III and IV, we can extrapolate several observations pertaining to the performance requirements of measurement and monitoring systems for next-generation networks:

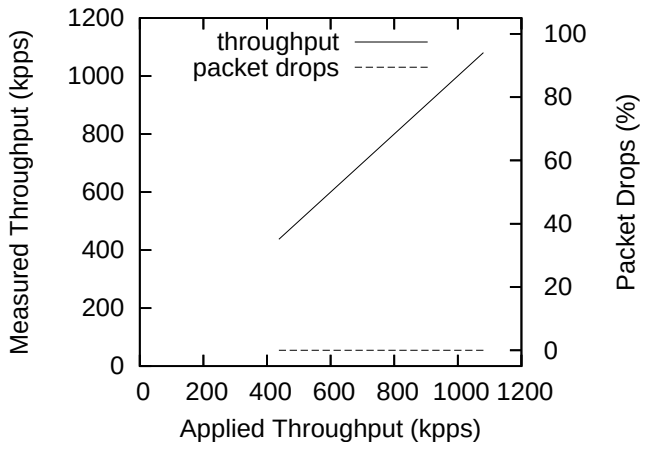
- Increasing link speeds will require either specialized hardware support for measurement systems or multiprocessor (e.g., network processor) support in order to handle the necessary I/O into the measurement system. Note that this clearly can be achieved since future routers will have to handle the same data rate. In the case of MM, however, packet will need to be handled by processing systems,



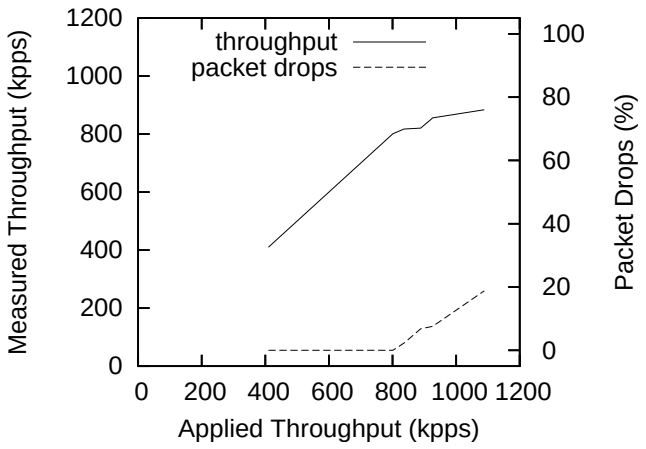
(a) System A - Packet Count



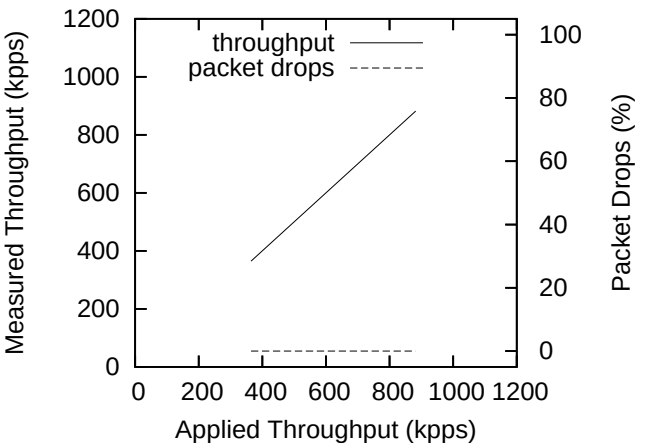
(b) System A - Flow Classification



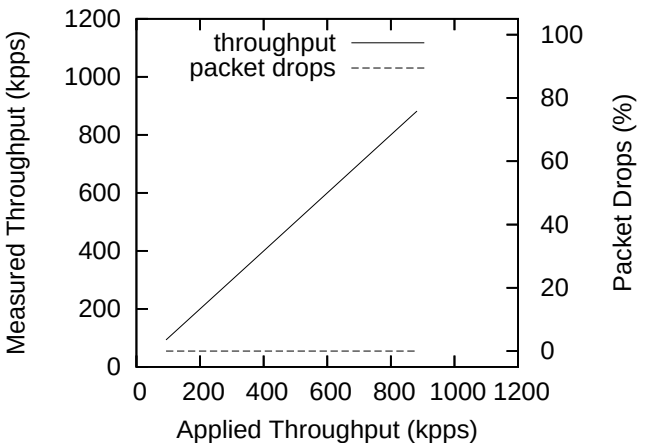
(c) System B - Packet Count



(d) System B - Flow Classification

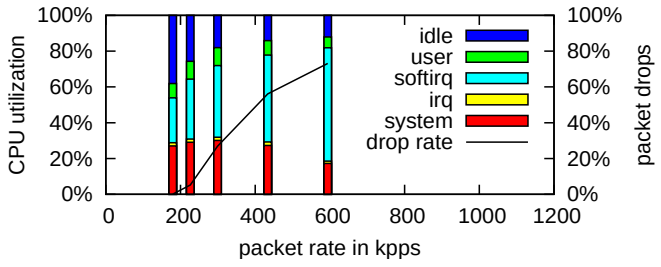


(e) System C - Packet Count

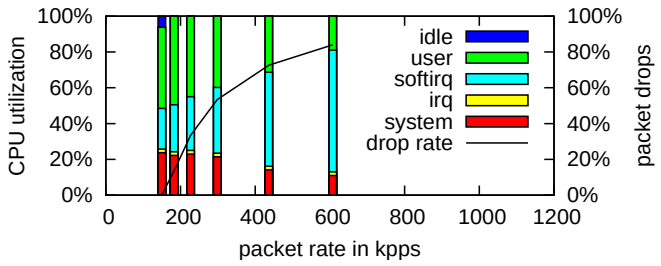


(f) System C - Flow Classification

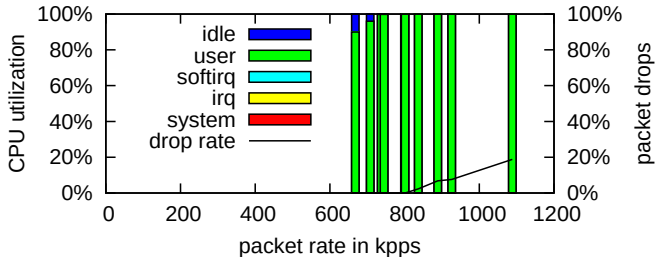
Fig. 3. Comparison of Performance of Different Measurement and Monitoring Systems. Each subfigure shows the throughput and drop rate for different offered loads.



(a) System A - Packet Count



(b) System A - Flow Classification



(c) System B - Flow Classification

Fig. 4. CPU Utilization and Packet Drop Rates for Different Uniprocessor Systems and Workloads.

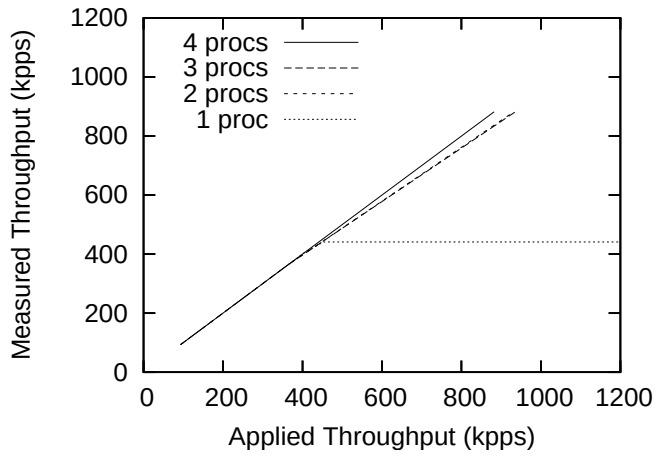


Fig. 5. Performance of Multiprocessor System with Flow Classification Application and Different Numbers of Processors.

which are different from the typical ASIC-based router implementations.

- Increasing complexity in the network may require more complex measurement and monitoring tasks. As an immediate result, the processing requirements will increase and possibly exceed the capabilities of a uniprocessor system.
- Advances in CMOS technology will make it possible to build embedded multiprocessors with more processor cores for less money and thus make multiprocessor-based MM system design a scalable solution.

It is clearly possible to design measurement and monitoring systems that meet the future performance requirement. But the cost of such systems needs to be put into the context of the cost of the networking equipment that provides the fundamental infrastructure. In order to monitor a Gigabit link (which needs to be able to handle several hundreds of thousands of packets per second) using a non-trivial monitoring application practically requires a multiprocessor system. When designing new network architectures, it is important to keep this in mind since the cost of such a processing system could easily exceed that of the entire router. As a rough comparison, a Gigabit LAN switch can be purchased for less than one hundred US dollars. A network processor from the IXP family with a link rate in the Gigabit per second range costs over one thousand US dollars. Even when considering the difference in market volume, the processing system is still relatively expensive. For next-generation network designs, this implies that it will not likely be possible to have intrusion detection or similar applications running at line rate on every edge, border, or access router.

Nevertheless, there are some solutions that can be considered to balance the need for measurement and monitoring and the cost of MM infrastructure:

- **Reduction of measurement speed**. Sampling can be used to obtain partial view of the actual traffic. Several techniques have been proposed to interpolate between samples in the context of network measurement [22].
- **Reduction of measurement density**. Inferencing techniques (e.g., network tomography in the context of active measurement [23]) can yield information with few measurement locations.
- **Hardware support for measurement**. Certain measurement tasks are well defined and could be implemented in custom logic. If such "measurement chips" could be mass-produced, measurement and monitoring could be widely deployed. However, for intrusion detection and similar applications it will probably be necessary to continue providing general-purpose programmability to support new algorithms that are continuously developed.
- **Integration with routing equipment**. Some router systems already employ network processors as port processors. These systems could simply be programmed to perform additional measurement tasks.

VI. Summary and Conclusions

This paper presents a discussion of three different network measurement and monitoring system architectures and presents their respective performance for different workloads. Our results show that multiprocessor-based measurement systems provide most performance and scalability. However, their system cost is high and therefore we consider a number of alternatives on how to deploy measurement capabilities in next-generation networks. The quantitative performance results obtained in our work can help researchers and practitioners understand performance tradeoffs in high-performance measurement and monitoring systems and aid in future designs.

Acknowledgements

This research was supported in part by National Science Foundation grant CNS-0325868. This work was performed while Siddhartha Bunga was at the University of Massachusetts and while interning at Bell Labs India (under the guidance of Jeyashanker S.R.). The PacketBench statistics for Snort were graciously provided by Chia-Hui Tai.

References

[1] V. Jacobson, "Congestion avoidance and control," in *Proc. of ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 314–329.

[2] S. Floyd, "TCP and explicit congestion notification," *SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[3] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proc. of ACM SIGCOMM 90*, Philadelphia, PA, Sept. 1990, pp. 200–208.

[4] M. S. Blumenthal and D. D. Clark, "Rethinking the design of the internet: the end-to-end arguments vs. the brave new world," *ACM Transactions on Internet Technology*, vol. 1, no. 1, pp. 70–109, 2001.

[5] D. Clark, K. Sollins, J. Wroclawski, D. Katabi, J. Kulik, X. Yang, B. Braden, T. Faber, A. Falk, V. Pingali, M. Handley, and N. Chiappa, "New Arch: future generation internet architecture," Defense Advanced Research Project Agency, Tech. Rep., Dec. 2003.

[6] D. D. Clark, "The design philosophy of the DARPA internet protocols," in *Proc. of ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 106–114.

[7] S. Floyd and V. Jacobson, "Random early detection (RED) gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[8] K. B. Egevang and P. Francis, "The IP network address translator (NAT)," Network Working Group, RFC 1631, May 1994.

[9] J. C. Mogul, "Simple and flexible datagram access controls for UNIX-based gateways," in *USENIX Conference Proceedings*, Baltimore, MD, June 1989, pp. 203–221.

[10] *The Open Source Network Intrusion Detection System*, Snort, 2004, http://www.snort.org.

[11] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft, "Pop-level and access-link-level traffic dynamics in a tier-1 POP," in *Proc. of the First ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001, pp. 39–53.

[12] k. claffy, G. Miller, and T. Kevin, "The nature of the beast: Recent traffic measurements from an internet backbone," in *Proc. of 1998 INET Conference*, Geneva, Switzerland, June 1998.

[13] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network Magazine*, vol. 11, no. 6, pp. 10–23, Nov. 1997.

[14] C. Fraleigh, S. B. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, Nov. 2003.

[15] K. Papagiannaki, N. Taft, and A. Lakhina, "A distributed approach to measure IP traffic matrices," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Italy, Oct. 2004, pp. 161–174.

[16] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *Proc. of Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, New York, NY, June 2004.

[17] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better NetFlow," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, OR, 2004.

[18] L. Golab, D. DeHaan, E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Identifying frequent items in sliding windows over on-line packet streams," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, Miami Beach, FL, Oct. 2003, pp. 173–178.

[19] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Italy, Oct. 2004, pp. 101–114.

[20] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, Berkeley, CA, Oct. 2005.

[21] E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in *Proc. of the 10th Annual European Symposium on Algorithms (ESA 2002)*, ser. Lecture Notes in Computer Science, vol. 2461, Rome, Italy, Sept. 2002, pp. 348–360.

[22] N. Hohn and D. Veitch, "Inverting sampled traffic," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, Miami Beach, FL, 2003, pp. 222–233.

[23] Y. Tsang, M. Yildiz, P. Barford, and R. Nowak, "Network radar: tomography from round trip time measurements," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Italy, Oct. 2004, pp. 175–180.

[24] *TCPDUMP Public Repository*, TCPDUMP, 2003, http://www.tcpdump.org.

[25] *DAG4.3E Network Monitoring Interface Card*, Endace Measurement Systems, 2005, http://www.endace.com/dag4.3GE.htm.

[26] *Intel Second Generation Network Processor*, Intel Corporation, 2005, http://www.intel.com/design/network/products/npfamily/.

[27] L. Zhao, Y. Luo, L. Bhuyan, and R. Iyer, "SpliceNP: a TCP splicer using a network processor," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, Princeton, NJ, Oct. 2005, pp. 135–143.

[28] T. Wolf, R. Ramaswamy, S. Bunga, and N. Yang, "An architecture for distributed real-time passive network measurement," in *Proc. of 14th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Monterey, CA, Sept. 2006, pp. 335–344.

[29] R. Ramaswamy and T. Wolf, "PacketBench: A tool for workload characterization of network processing," in *Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6)*, Austin, TX, Oct. 2003, pp. 42–50.

[30] R. Ramaswamy, N. Weng, and T. Wolf, "Analysis of network processing workloads," in *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, Mar. 2005, pp. 226–235.

[31] N. Weng and T. Wolf, "Profiling and mapping of parallel workloads on network processors," in *Proc. of The 20th Annual ACM Symposium on Applied Computing (SAC)*, Santa Fe, NM, Mar. 2005, pp. 890–896.

[32] *PayloadPlus^{TM} Fast Pattern Processor*, Lucent Technologies Inc., Apr. 2000, http://www.agere.com/support/non-nda/docs/FPPProductBrief.pdf.

[33] J. Allen, B. Bass, C. Basso, R. Boivie, J. Calvignac, G. Davis, L. Frelechoux, M. Heddes, A. Herkersdorf, A. Kind, J. Logan, M. Peyravian, M. Rinaldi, R. Sabhikhi, M. Siegel, and M. Waldvogel, "IBM PowerNP network processor: Hardware, software, and applications," *IBM Journal of Research and Development*, vol. 47, no. 2/3, pp. 177–194, 2003.

[34] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson, "Design principles for accurate passive measurement," in *Passive and Active Measurement Workshop (PAM 2000)*, Hamilton, New Zealand, Apr. 2000.