The Architecture of NG-MON: A Passive Network Monitoring System for High-Speed IP Networks[1]

Se-Hee Han[1], Myung-Sup Kim[1], Hong-Taek Ju[2], and James Won-Ki Hong[1]

[1] Department of Computer Science and Engineering, POSTECH, Korea
[2] Department of Computer Engineering, Keimyung University, Korea
{sehee, mount, juht, jwkhong}@postech.ac.kr

Abstract. This paper presents the design of a next generation network traffic monitoring and analysis system, called NG-MON (Next Generation MONitoring), for high-speed networks such as 10 Gbps and above. Packet capturing and analysis on such high-speed networks is very difficult using traditional approaches. Using distributed, pipelining and parallel processing techniques, we have designed a flexible and scalable monitoring and analysis system, which can run on off-the-shelf, cost-effective computers. The monitoring and analysis task in NG-MON is divided into five phases; packet capture, flow generation, flow store, traffic analysis, and presentation. Each phase can be executed on separate computer systems and cooperates with adjacent phases using pipeline processing. Each phase can be composed of a cluster of computers wherever the system load of the phase is higher than the performance of a single computer system. We have defined efficient communication methods and message formats between phases. Numerical analysis results of our design for 10 Gbps networks are also provided.

1 Introduction

Today, multi-gigabit networks are becoming common within or between ISP networks. The bandwidth of ISP's backbone networks is evolving from OC-48 (2.5Gbps) to OC-192 (10Gbps) to support rapidly increasing Internet traffic. Also, Ethernet networks are evolving from gigabit to 10 Gbps. Further, the types of traffic on these links are changing from simple text and image based traffic to more sophisticated and higher volume (such as streaming rich media, peer-to-peer). Monitoring and analyzing such high-speed, high-volume and complex network traffic is needed, but it lies beyond the boundaries of most traditional monitoring systems.

Sampling is a popular method that most monitoring systems adopted to overcome this problem [1]. However, the sampling method is neither accurate nor adequate for

[1] The authors would like to thank the Ministry of Education of Korea for its financial support toward the Electrical and Computer Engineering Division at POSTECH through its BK21 program.

some applications (e.g., usage-based billing or intrusion detection system). Another approach is by the adoption of purpose-built hardware [2]. Unfortunately, the development cost of such hardware approach is very high, and the hardware can get outdated quickly. ISPs would be required to replace them to meet the requirement as the network bandwidth increases. Therefore, we need a solution that is flexible, scalable, and cost-effective.

This paper suggests the design of such a solution. In our earlier work, we had developed a passive network traffic monitoring system, called WebTrafMon [3, 4]. It could monitor 100 Mbps links and was able to capture packets without any loss. When we used it to monitor faster links than 100 Mbps, we encountered several problems. The amounts of incoming packets were beyond the processing capacity of the probe. And the required storage space for flow data increased linearly as the link speed increased. Also, the analyzer took a long time to complete its tasks.

So, we had to come up with a new approach to solve the problem. At first, we subdivided the monitoring process into multiple phases, and distributed the processing load over them by allocating a system for each phase. If the distributed load in each phase is still beyond the capability of the system, it can be composed of a cluster of systems. By using this approach, we have designed a flexible and scalable network traffic monitoring and analysis system, called NG-MON (Next Generation MONintoring). NG-MON uses the passive monitoring method.

The organization of this paper is as follows. The requirements of the NG-MON are enumerated in Section 2 and the design of NG-MON is described in Section 3. Numerical analysis results of our design for 10 Gbps networks are provided in Section 4. In Section 5, we compare our approach with other approaches proposed thus far. Finally, concluding remarks are given and possible future work is mentioned in Section 6.

2 Requirements

The following are the requirements we have considered in designing NG-MON.

Distributed architecture: With a single general purpose PC system, it is hard to monitor and analyze all the packets on a multi-gigabit network. So it is required to divide monitoring task into several functional units and distribute processing loads. With respect to the distribution method, we considered the pipelined and parallel methods. And we also considered the packet distribution by using the functions which are provided by network devices.

Lossless packet capture: We need to capture all packets on the link without any loss so to provide required information to various applications.

Flow-based analysis: When analyzing, it is better to aggregate packet information into flows for efficient processing. By doing this, packets can be compressed without any loss of information.

Consideration of limited storage: The amount of captured packets in high-speed networks is more than hundreds of megabytes per minute even though being aggregated into flows [2]. An efficient method is needed for storing these large amounts of flows and analyzed data in the limited storage.

Support for various applications: It should be flexible enough to provide data to various applications in diverse forms. When a new application needs to use the system, it should be able to easily support the application without changing the structure of the system.

3 Design of NG-MON

In the design of NG-MON, the key features we have employed are pipelined distribution and load balancing techniques. In Fig. 1, traffic monitoring and analysis tasks are divided into five phases: packet capture, flow generation, flow store, traffic analysis, and presentation of analyzed data.

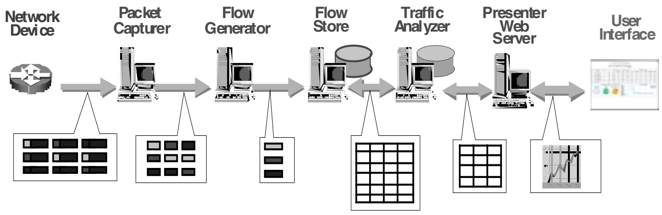

Fig. 1. Pipelined Architecture of NG-MON

These five phases are serially interconnected using a pipelined architecture. One or more systems may be used in each phase to distribute and balance the processing load. Each phase performs its defined role in the manner of a pipeline system. This architecture can improve the overall performance. And each phase is configured with a cluster architecture for load distribution. This provides good scalability. We have also defined a communication method between each phase. Each phase can be replaced with more optimized modules as long as they provide and use the same defined interface. In the following sections, we describe each phase in detail. This gives flexibility. Rather than using expensive server-level computers, we use inexpensive, off-the-shelf PC-level computers. Since our solution is all software-based, as more processing power is needed one can simply replace existing hardware or add more to wherever is needed. We believe this is a very cost-effective and scalable approach.

3.1 Packet Capture

In the packet capture phase, one or more probe machines (or packet capturer) collect the entire raw packets passing through the network link. By using the splitting function provided by an optical splitter [5], all the packets on the link are directed toward probe systems as illustrated in Fig. 2. We can also use the mirroring function provided in network devices such as switches and routers for distributing traffic to multiple probes. Each probe processes incoming packets and keeps the minimum packet header information that we are interested in. In Fig. 2, the output of each probe is a collection of the packet header information that is derived from raw packets.

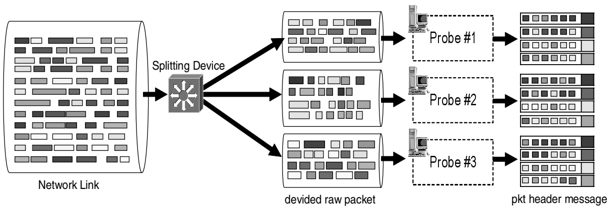

Fig. 2. Packet Distribution and Packet Capture

A single off-the-shelf PC cannot process all the packets coming from the high-speed links such as 10 Gbps networks due to performance limitations [6]. It is essential to use multiple systems for capturing all the packets without loss. Although processing loads are distributed, the packets in the same flow can be scattered. Each probe has as many export buffer-queues as the number of flow generators. Each export buffer-queue is for flow generators. The probe fills the buffers with header information using the 5-tuple based hashing over export buffer-queues. When this buffer-queue is full, the probe constructs a message containing captured packet headers and then sends it to the next phase, the flow generator. The destination of the constructed message is assigned among the addresses of flow generators as to the buffer-queues. Therefore, temporally scattered packets in the same flow would be put together and sent to the same flow generator. One message is composed of up to 50 packet header information. The format of raw packet header data is given in Fig. 3.

0	8	16	24	31
Timestamp				
Source Address				
Destination Address				
Source Port		Destination Port		
Packet Size		Ether_Type		
Flag and Offset		IP Identification		
Protocol Number	ToS	TCP Flags	Capture ID	

Fig. 3. Packet Header Data Format

The size of packet header data kept is 28 bytes for each packet. All the fields except Timestamp and Capture ID are extracted from IP and TCP/UDP headers of each packet. The Timestamp indicates the time when a packet is captured by a probe. The Capture ID indicates the system, which captured that packet for later use.

3.2 Flow Generation

There are various definitions about the flow [7, 8, 9]. In this paper, we use the traditional one, which defines the flow as a sequence of packets with the same 5-tuple: source IP address, destination IP address, protocol number, source port, and destination port.

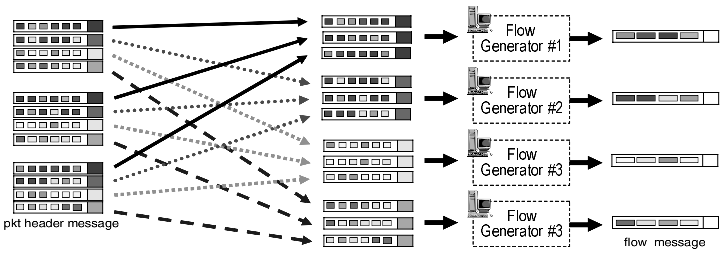

Fig. 4. Load Distribution in the Flow Generation Phase

In Fig. 4, the messages from the packet capture phase are distributed over flow generators by assigning their destinations to the corresponding flow generators.

A flow generator stores the flow data in its memory area for processing. When a message containing raw packet data arrives, the flow generator searches the corresponding flow data from its flow table and then updates it, or creates a new flow if one does not already exist. Packets in the same flow are aggregated into the same entry of the table by increasing the packet count and adding the length to the total packet size. The flow generator exports the flow data to the flow store when one of the following conditions is satisfied: when the flow is finished (if TCP, when a FIN packet received), the time has expired or the flow table is filled.

0	8	16	24 31
Flow Start Time			
Flow End Time			
Source Address			
Destination Address			
Source Port		Destination Port	
Packet Count			
Total Bytes in a Flow			
Protocol Type	Reserved	ToS	Protocol Number

Fig. 5. Flow Data Format

The flow data format is given in Fig. 5. Besides the 5-tuple information, the flow data has several other fields such as flow start time and flow end time. The flow start time indicates the time when the first packet of a flow is captured by a probe, and the flow end time means the capture time of the last packet of the flow. The size of the flow data format is 32 bytes. For our flow generator, it can send up to 40 flows in a single message of approximately 1350 bytes.

3.3 Flow Store

In our earlier work [4], we realized that one of the bottlenecks of the monitoring process is a storing of flow data. Therefore, when the flow data is stored to the flow store, the load balancing should be considered. In Fig. 6, the destination of the exported messages is assigned among the flow stores in turn by a round-robin algorithm. The assigning period is determined by the transfer rate of export flow data, capabilities of the flow stores, and the number of flow stores. In this way, the processing load to store the flow data is distributed over the flow stores.

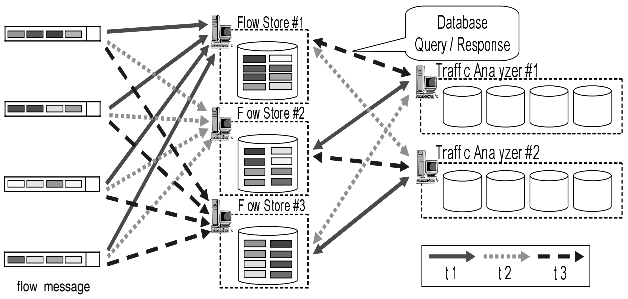

Fig. 6. Load Distribution in the Flow Store Phase

When several flow generators assign a destination flow store of the messages, there can be a time synchronization problem. But the components of NG-MON would tend to be deployed in a local area network, thus the required degree of time synchronization is not so high. Therefore, the time synchronization protocol like NTP [10] can be used to synchronize the system clocks of the components.

In our system, we separate write (i.e., insert) operations from database query operations performed by the analyzers. Insertion does not occur at the same time as other operations in a single flow store. Thus, traffic analyzers query databases of flow stores when they are not receiving flow data. An example is illustrated in Fig. 6. At time t1, the flow store #1 receives flow data from flow generators and the flow stores #2 and #3 process the query from traffic analyzers. That is, the flow store concentrates on operation requests of one side at a time. Flow stores discard the flow data table when they are finished with analysis by traffic analyzers. Only the most recent flow data is stored in the flow store, so the flow store only requires a small, fixed amount of disk space.

There can be various traffic analyzers for supporting various applications after the flow store phase. This means that the flow store should provide an analysis API to analyzers.

3.4 Traffic Analysis

In this phase, the traffic analyzer queries the flow data stored in the flow store according to the various analysis scopes. The analyzer sends query messages to the flow stores and makes various matrices and tables from the response. If all the scope of analysis is put into one table, the size of a single table will be too large to manage. Therefore, we place several reduced set of tables corresponding to each analysis scope. For example, the analyzer in Fig. 7, provides the details on network throughput with protocol and temporal analysis. And in order to provide temporal analysis, the analyzer has a set of tables according to every time-unit of minute, hour, day, month, and year. It is impractical to store all the flow data into the time-series tables because of voluminous data, and limitation of storage space. To reduce the storage requirement, we preserve tables with only the most significant *N* entries. Thus, the total size of database will have some degree of boundary. The analyzer fills up the tables simultaneously in a pipelined manner. If the reception time period of flow stores is 5 minutes, there can be 20 tables for storing every 5 minutes' analyzed flow data. After updating the 5-minute table, the corresponding hour table gets updated. There should be these kinds of time-series tables for each scope of analysis in the analyzer.

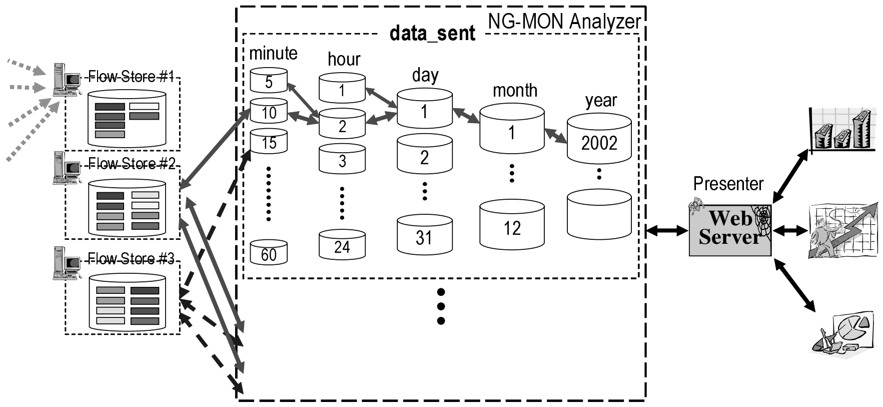

Fig. 7. Traffic Analyzer and Various Applications

The presentation phase can provide an analysis to users about the traffic in various forms using the Web interface. Before designing an analyzer, we have to determine analysis items to be shown in this phase. Then the traffic analyzer can generate the corresponding DB tables based on these items. That is, a different analyzer is required to support a different purpose application. Because tables contain analyzed data which is ready to be shown, the time needed to create reports and HTML pages is very short, typically less than a second.

3.5 Presentation

This phase provides analyzed data to corresponding applications. Because the header information of all packets has been stored to the flow store being compressed into flows, it can provide any information to applications in a flexible and efficient way. NG-MON can provide necessary information to the billing applications on IP networks, IDS systems, and so on.

4 Design Analysis

We have validated our design of NG-MON analytically for monitoring high-speed networks. We already described the flexibility and the scalability of our design. At each phase, we can assign a number of systems for load distribution, or can merge some phases into one system. The appropriate number of systems will be determined from this analysis.

4.1 Assumptions

The monitored network link speed is 10 Gbps, and our system captures all the packets inbound and outbound. The size of a single packet header data is 28 bytes, and that of a single flow data is 32 bytes. Then we calculate the size of data to be processed in a second in a probe, flow generator, and flow store. The average number of packets per flow (C_{avg}) is 16, which is derived from the flow generator test on our campus network. The average packet size (P_{avg}) is 550 bytes from the same testing. So in this numerical analysis we use the follow values as shown in Table 1.

Table 1. Symbols and its values

Symbol	Description	Value
L	Link speed	10 Gbps
d	Full duplex factor	2
H_p	A single packet header data size	28 bytes
H_f	A single flow data size	32 bytes
C_{avg}	Average packet count per flow	16
P_{avg}	Average packet size	550 bytes

4.2 The Amount of Data to Be Processed

The total raw packet (T_p) in the packet capture phase, the total raw packet header information (T_h) in the flow generation, and the total flow data (T_f) in the flow store processed in one second are as follows:

$$\text{Total raw packets}\,(T_p) = L \times \frac{d}{8} = 10 \times \frac{2}{8} = 2.5\ \text{Gbytes/sec}$$

Total raw packet header data (T_h)

$$= \frac{T_p}{P_{avg}} \times \left[H_p + \frac{\text{MAC header} + \text{IP header} + \text{UDP header}}{\text{the number of raw packet headers in an export UDP packet}} \right]$$

$$= \frac{2500}{550} \times \left[28 + \frac{14 + 20 + 8}{50} \right] = 131.1\ \text{Mbytes/sec}$$

$$\text{Total flow data (Tf)} = \frac{\text{Tp}}{\text{Pavg}} \times \frac{\text{Hf}}{\text{Cavg}} = \frac{2500}{550} \times \frac{32}{16} = 9 \text{ Mbytes/sec}$$

It seems meaningless to calculate the size of total flow data in a second (T_f) because it is too short to be used as an exporting period in the flow generation phase. So we observe the size of flow data in a minute (1 min-T_f), five minutes (5 mins-T_f), and an hour (1 hour-T_f):

1 min-T_f = 9(T_f) × 60 = 540 Mbytes

5 mins-T_f = 2.7 Gbytes

1 hour-T_f = 32.4 Gbytes

If we choose one minute as the exporting period, each flow store requires only 540 Mbytes of disk space. In the same way, if we choose 5 minutes, 2.7 Gbytes are required per flow store.

4.3 Allocation of Systems to Each Phase

The amount of data to be processed at each phase is typically beyond the processing capacity of a single off-the-shelf general-purpose PC system. For example, a single probe cannot process all the raw packets of 2.5 Gbytes in one second because of the limited network interface card (NIC) and PCI bus capacity in a PC.

During processing in a single system, there are several subsystems that affect the monitoring capacity: NIC bandwidth, PCI bus bandwidth, memory, CPU processing power, storage and so on. Let us consider a computer system with a gigabit Ethernet NIC and 66MHz/64bit PCI bus and 1 Gbyte RDRAM (800MHz/16bit) and 2 GHz Pentium 4. Then the theoretical max transfer rate of a gigabit Ethernet NIC is 125 Mbytes/sec, PCI bus is 533 Mbytes/sec, and dual channel RDRAM (800MHz/16bit) is 3.2 Gbytes/sec [6, 12]. The probe can have multiple NICs, one for sending raw packet header information to flow generators, the others for capturing. Then, there can be 4 NICs within the bandwidth of a PCI bus so far as the number of PCI slots permits. Therefore, it requires 7 probe machines to receive total raw packets of 2.5 Gbytes in a second for a full duplex 10 Gbps link.

In the flow generator phase, it receives 131.1 Mbytes of raw packet header per second (T_h), so theoretically 1 flow generator which has 2 NICs is needed. In the flow store phase, though it is sufficient for processing the rate of flow data with one system, the execution time of queries affects the required number of flow stores. Such an execution time varies as to the kind of database system, DB schema, query construction, and so on. Therefore, the number of flow stores is flexible regarding to those kinds of factors. In our previous work [4], it took about 4 seconds to insert 20 Mbytes of raw packet headers into MySQL database running on an 800 MHz Pentium 3 with 256 Mbytes of memory. If we assume the runtime of an insert is *O(N)*, it will take 150 seconds to insert 1 min-T_f data into the database. Here we assume the analysis system takes about 2 minutes for querying. Then it will take 4 minutes and 30 seconds to insert and analyze an 1-minute flow data. As we have to finish all these operations within 1 minute, it requires 3 systems for inserting, and 2 systems for analyzing in the flow store phase.

Table 2. The required number of systems in each phase

	Packet Capture	Flow Generation	Flow Store	Total
100 Mbps	1			1
1 Gbps	1		2	3
10 Gbps	7	1	5	13

Therefore, it requires approximately 13 systems (7 in the packet capture phase, 1 in the flow generation phase, 5 in the flow store phase) to provide flow data to analysis systems in a fully-utilized, full-duplex 10 Gbps network. In a 100 Mbps network, the amount of flow data in a minute is less than 10 Mbytes. Thus, three phases can merge into one system. In a 1 Gbps network, packet capture and flow generation phase can merge into one system which has 3 NICs. And the flow store phase can be composed of 2 systems if the time to insert and query the 1min-T_f of 1 Gbps network is less than a minute per each operation. Table 2 summarizes the required number for systems in each phase for 100 Mbps, 1 Gbps and 10 Gbps links.

5 Related Work

Table 3 compares NG-MON with other passive network monitoring systems. Ntop [13] is a monitoring software system that provides detailed protocol analysis and a graphical user interface. However, Ntop is not suitable for monitoring high-speed networks from our experience in deploying in an operational network. It cannot scale beyond monitoring a fully-utilized 100 Mbps link.

FlowScan [14] is a NetFlow analysis software system that uses cflowd [15], RRD Tool [16], and arts++ [17]. It can only process the NetFlow [18] data format and is not suitable for monitoring high-speed networks either. Ntop and FlowScan are appropriate for analyzing relatively low-speed networks such as a WAN-LAN junction or a LAN segment. Our approach of distributing the processing load may be applied to Ntop and FlowScan in improving their processing capabilities.

CoralReef [19] is a package of library, device driver, class, and application for network monitoring developed by CAIDA [20]. CoralReef can monitor up to OC-48 network links. With respect to the load distribution, only CoralReef suggests a separation of flow generation and traffic analysis, but without consideration of clustering of processing systems in each stage.

Sprint's IPMon project [21] developed a probe system for collecting traffic traces, which is used for off-line analysis after transferring to a laboratory. Their approach uses purpose-built hardware to assist the packet capturing and processing.

Table 3. A Comparison of NG-MON with Related Work

	Ntop	FlowScan	CoralReef	Sprint IPMon	NG-MON
Input	Raw Traffic, NetFlow	NetFlow	Raw Traffic	Raw Traffic	Raw Traffic
Output	Throughput	Throughput	Throughput	Packet Trace	Throughput
Speed	<<100Mbps	<<155Mbps	<<622Mbps	10Gbps	10Gbps
Solution	Software	Software	Hardware+ Software	Hardware+ Software	Software
Sampling used	No	Yes(device)	Configurable	No	No but Configurable
Analysis	On-line	On-line	On-line	Off-line	On-line

Our NG-MON has been designed for monitoring high-speed IP networks. It takes raw traffic packets as input, and analyzes captured data online and then generates various throughput related data. NG-MON is a software-based solution (i.e., does not depend on any specific hardware), which can be easily installed and run on a variety of Unix and Linux platforms. NG-MON does not use sampling for capturing packets without any loss. However, the system configuration user interface allows the system to be configured to capture packets using sampling if needed.

6 Conclusion and Future Work

In this paper, we have presented the design of NG-MON, a scalable and flexible monitoring and analysis system for high-speed IP networks. NG-MON adopted a pipelined and parallel architecture for achieving our goal. NG-MON with its pipelined and parallel architecture can process packets without any loss. Multi-gigabit networks generate a lot of traffic and thus the amount of data generated from packet capturing is incredibly large. Our packet and flow processing method requires small, fixed amount of disk space on each flow store. We have also presented a numerical analysis of our design and have presented the theoretical number of systems for monitoring 10 Gbps networks as an example.

NG-MON can play a major role in providing necessary information to multitudes of applications. For example, it can be used as a basis for billing on IP-based applications (such as VoIP, Internet access). It can be also used as a basis for intrusion detection where capturing all packets is essential. Customer relationship management (CRM) is a hot topic for ISPs these days. NG-MON can provide useful user usage information for such purpose.

References

[1] K. C. Claffy, G. C. Polyzos and H. W. Braun, "Application of Sampling Methodologies to Network Traffic Characterization," Proc. of ACM SIGCOMM, Hamilton, New Zealand, May 1993, pp. 194-203.

[2] G. Iannaccone, C. Diot, I. Graham, N. McKeown, "Monitoring very high speed links," Proc. of ACM SIGCOMM Internet Measurement Workshop, San Francisco, USA, November 2001, pp. 267-271.

[3] James W. Hong, Soon-Sun Kwon and Jae-Young Kim, "WebTrafMon: Web-based Internet/Intranet Network Traffic Monitoring and Analysis System," Computer Communications, Elsevier Science, Vol. 22, No. 14, September 1999, pp. 1333-1342.

[4] Soon-Hwa Hong, Jae-Young Kim, Bum-Rae Cho, James W. Hong, "Distributed Network Traffic Monitoring and Analysis using Load Balancing Technology," Proc. of 2001 Asia-Pacific Network Operations and Management Symposium, Sydney, Australia, September 2001, pp. 172-183.

[5] Aurora, Optical Splitter, http://www.aurora.com/products/headend-OP3xSx.html.

[6] J. Michael, H. Braun and I. Graham, "Storage and bandwidth requirements for passive Internet header traces," Proc. of the Workshop on Network-Related Data Management 2001, Santa Barbara, California, USA, May 2001.

[7] Siegfried Lifler, "Using Flows for Analysis and Measurement of Internet Traffic," Diploma Thesis, Institute of Communication Networks and Computer Engineering, University of Stuttgart, 1997.

[8] J.Quittek, T. Zseby, B. Claise, K.C. Norsth, "IPFIX Requirements," Internet Draft, http://norseth.org/ietf/ipfix/draft-ietf-ipfix-architecture-00.txt.

[9] CAIDA, "Preliminary Measurement Spec for Internet Routers," http://www.caida.org/tools/measurement/measurementspec/.

[10] David L. Mills, Network Time Protocol, RFC 1305, IETF Nework Working Group (March 1992), http://www.ietf.org/rfc/rfc1305.txt.

[11] K. Thompson, G. Miller, and M. Wilder, "Wide-area internet traffic patterns and charateristics," IEEE Network, vol. 11, no. 6, November-December 1997, pp. 10-23.

[12] Rambus, RDRAM memory, http://www.rambus.com/technology/rdram_overview.html.

[13] Luca Deri, Ntop, http://www.ntop.org.

[14] Dave Plonka, "FlowScan: A Network Traffic Flow Reporting and Visualization Tool," Proc. of 2000 LISA XIV, New Orleans, USA, December 2000, pp. 305-317.

[15] Daniel W. McRobb, "cflowd design," CAIDA, September 1998.

[16] RRDtool, http://www.rrdtool.com.

[17] CAIDA, ARTS++, http://www.caida.org/tools/utilities/arts/.

[18] Cisco, "NetFlow," http://www.cisco.com/warp/public/732/Tech/netflow/.

[19] K. Keys, D. Moore, Y. Koga, E. Lagache, M. Tesch, and K. Claffy, "The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite," Proc. of Passive and Active Measurement Workshop 2001, Amsterdam, Netherlands, April 2001.

[20] CAIDA, http://www.caida.org.

[21] Sprint ATL, "IP Monitoring Project," http://www.sprintlabs.com/Department/IP-Interworking/Monitor/.