Telecommun Syst (2008) 37: 71–84
DOI 10.1007/s11235-008-9073-5

Integrating heterogeneous network monitoring data

Chi Zhang · Bin Liu · Xun Su · Heidi Alvarez · Julio Ibarra

Published online: 15 February 2008

Abstract In this paper, we investigate the integration of heterogeneous network monitoring data. Specifically, we will synchronize and integrate flow-level records, exemplified by Cisco NetFlow, and packet-level traces, exemplified by NLANR PMA. The integration can facilitate cross-validation and complementary utility. However, finding the correspondences of timestamps/flows/packets between the PMA and Netflow is non-trivial, because they have different levels of granularity, different sampling strategy, different time sources, and different IP address masking. To integrate heterogeneous monitoring data, we first synchronize their timestamps, and then match their masked IP addresses. Our key observation is that although the IP addresses are masked, some other header fields can be exploited to match different types of monitoring data. In order to reduce the search space and the processing overhead, we have adopted a top-down approach to limit the search scope, and iterative algorithms to reduce the matching errors step by step.

Keywords Heterogeneous network monitoring data · NetFlow · PMA

This work is sponsored by the University Research Program of Cisco Systems Inc from 09/01/04 to 08/31/06.

C. Zhang (✉)
Juniper Networks, 1220 North Mathilda Ave, Sunnyvale, CA 94089, USA
e-mail: czhang@cs.fiu.edu

B. Liu
Microsoft, Redmond, WA 98052, USA
e-mail: bliu@microsoft.com

X. Su
Fulcrum Microsystems, Calabasas, CA 91302, USA
e-mail: xsu@fulcrummicro.com

H. Alvarez · J. Ibarra
Center for Internet Augmented Research and Assessment, Florida International University, Miami, FL 33199, USA

H. Alvarez
e-mail: heidi@fiu.edu

J. Ibarra
e-mail: julio@fiu.edu

1 Introduction

1.1 Background

While the end-to-end design principle of Internet enables new applications to flourish without modifications to the network, it also makes network management a significant challenge. Therefore, developing advanced techniques to monitor, analyze and understand the dynamics of the Internet traffic is crucial. Traditionally, IP networks do not provide sufficient fine-grained monitoring supports. For example, SNMP allows network operators to obtain the total number of packets/bytes traversing a network link within a time period (e.g. 5 minutes). Recently, a number of fine-grained network monitoring toolkits have been developed. Each of them has their own strengths and weaknesses. It would be interesting to integrate different monitoring tools and cross-validate heterogeneous measurement data, such as Cisco NetFlow (http://www.cisco.com/en/US/products/ps6601/products_ios_ protocol_group_home.html) and NLANR PMA (http://pma.nlanr.net/).

NetFlow in Cisco routers periodically samples packets and counts the total number of sampled packets/bytes, for each TCP/UDP flow it has seen. The seven tuple (`SrcIP`,

DstIP, SrcPort, DstPort, IP Protocol Number, ToS, Input Logical Interface) is used to uniquely identify a TCP/UDP flow. For each flow observed, it also maintains timestamps of the first/last sampled packets, cumulative OR of TCP flags, and routing information (such as address prefix masks, AS numbers, the IP address of the next hop router). In addition, ICMP traffic can also be recorded. According to a number of rules, flow-level records can expire, and are then encapsulated into UDP packets and exported to an external computer for offline analysis. As link speeds and the number of flows increase, keeping a counter for each flow is not scalable. This makes packet sampling a necessary compromise for performance and scalability. With packet sampling, not all packets of a long flow can be seen by Netflow, and some short flows might even not be recorded by Netflow. The selection of sampling interval is a tradeoff between the information accuracy and the processing overhead [6].

Table 1 shows the relevant flow attributes in a NetFlow record (Version 5). We categorize these attributes into three groups: (a) time- and sampling-related attributes. (b) flow attributes available from packet headers. These attributes are also available to sniffing-based packet traces, such as PMA. (c) routing-related attributes. These attributes are obtained from the routing mechanisms inside the router, and are thus unavailable to PMA packet traces.

While NetFlow provides *sampled flow-level* records, the Passive Measurement and Analysis (PMA) provides *unsampled packet-level* traces. An optical splitter is inserted into a network link to sniff the packets traversing the link. A computer connected to the splitter then captures the TCP/UDP/IP header of every packet into a log file. Due to the storage overhead, typically PMA captures all packet headers in a short period of time every several hours (e.g., 90 seconds every 2 to 3 hours). A timestamp in microseconds and the ingress interface number are associated with each packet header.

Table 2 shows the trace file format of PMA, which contains all the information in IP header and most of the fields in the layer 3 header (e.g. TCP/UDP header).

Table 1 NetFlow record format

NetFlow attribute	Description
Time- and sampling-related attributes	
UNIX_SECS	Seconds since 0000 UTC 1970
UNIX_NSECS	Residual nanoseconds since 0000 UTC 1970
SYSUPTIME	Time in milliseconds since this device was first booted
SAMPLING_INTERVAL	The sampling interval
FIRST	System up time at start of flow
LAST	System up time at the time the last packet of the flow was received
Flow attributes available from packet headers	
DPKTS	Packets in the flow
DOCTETS	Total number of Layer 3 bytes in the packets of the flow
SRCADDR	Source IP address
DSTADDR	Destination IP address
SRCPORT	TCP/UDP source port number or equivalent
DSTPORT	TCP/UDP destination port number or equivalent
PROT	IP protocol type
TCP_FLAGS	Cumulative OR of TCP flags
Routing-related attributes	
NEXTHOP	IP address of next hop router
INPUT	SNMP index of input interface
OUTPUT	SNMP index of output interface
SRC_MASK	Source address prefix mask bits
DST_MASK	Destination address prefix mask bits
SRC_AS	Source AS number
DST_AS	Destination AS number

1.2 Goal and approaches

The goal of this research is to compare and integrate the flow-level records, exemplified by NetFlow, and packet-level traces, exemplified by PMA. That is, given two monitoring data with heterogeneous formats and information, how to find the correspondences of timestamps/flows/packets, between the two? This is motivated by: (a) *Cross Validation*: With one type of monitoring data, we can examine the accuracy of network traffic represented by the other type of monitoring data. In fact, in Sect. 4, we will demonstrate that based on the PMA data, we are able to discover a problem of NetFlow Version 5 with multicast packets. On the other hand, using Cisco NetFlow data, we revealed in Sect. 2 that the PMA box may occasionally ignore some packets during a capturing period. (b) *Complementary Utility*: Once integrated, different monitoring tools can potentially complement each other. For example, PMA provides more details (packets vs. sampled flows) in its short capturing periods, outside of which are only covered by Netflow. As another example, Neflow records export routing information (such as AS numbers and network masks) available only inside the router. PMA packet traces collected on a link, on the other hand, cannot provide this information.

However, the task of comparison and integration is non-trivial, because it's difficult to find the correspondences between these two types of network monitoring data even collected at the same location. First, while PMA provides

Table 2 PMA trace file format

Timestamp (seconds) 4 bytes			
Interface # 1 byte		Timestamp (microseconds) 3 bytes	
IP Ver 4 bits	IHL 4bits	Type of service 1 byte	Total length 2 bytes
Identification 2bytes		Flag 4bits	Fragment Offset 12 bits
TTL 1byte		Protocol 1 byte	Header Checksum 2 bytes
Source IP Address 4 bytes			
Destination IP Address 4 bytes			
Source Port 2 bytes		Destination Port 2 bytes	
Sequence Number 4 bytes			
Acknowledgment Number 4 bytes			
DataOffset 4bits	Reserved	Flags 6 bits	Windows

packet-level traces, NetFlow provides flow-level records. More importantly, while NetFlow samples the incoming packet stream 24 hours a day, PMA attempts to capture all packets in a few short periods. Secondly, PMA and Netflow synchronize with different time sources with different precisions. Third, to protect privacy, the IP addresses from different monitoring data may be masked or anonymized using different algorithms. Even if the anonymization algorithms are the same, the actual mapping of IP addresses can be different, depending on different traffic observed (i.e. sampled vs. unsampled). Finally, the processing algorithm must be efficient, since the size of monitoring data is huge. Scanning the entire data set to find the correlation is time consuming.

To integrate NetFlow and PMA data, we first synchronize their timestamps, and then match their masked IP addresses. Our key observation is that although the IP addresses are masked, some other header fields are still available to facilitate finding the correspondences between different monitoring data. For example, the TCP/UDP port numbers can be leveraged even though different flows from different IP addresses may have the same source/destination port numbers. Also, although the NetFlow records only provide the timestamps for the first and the last sampled packets, the TCP SYN flag can be used to identify the actual first packet in a flow, with which a one-to-one correspondence can be established. In fact, our data-driven approach can even deal with clocks losing synchronization completely (Sect. 2.3).

In order to limit the processing overhead, we have adopted a top-down approach to reduce the matching errors step by step. We first use a coarse-grained but lightweight approach to narrow down the timestamp difference to 1 minute. We then use a fine-grained approach to accurately (within 10 ms) estimate the timestamp differences in a reduced search space. We further effectively match the anonymized IP addresses, based on synchronized timestamps.

1.3 Testbed

We conducted our experiments on a campus network, as shown in Fig. 1. The Cisco GSR router in Fig. 1 is a high-speed connection point between Internet2, a campus network, and 9 non-US research and education networks. The non-US R&E networks are connected to an ATM switch, which is further linked to the Cisco router via an OC-3 line. The configuration of the ATM switch provides a virtual circuit between each non-US R&E network and the Cisco router.

CISCO NetFlow Version 5 is enabled on the Cisco GSR router. The NetFlow sampling interval is 100 packets, and the expiration timer setting uses the default vales. Most of the traffic observed in the Cisco router is between Interface 1 (to Internet2) and Interface 3 (to ATM switch). A PMA box, provided and managed by NLANR, is installed on the OC-3 link between the router and the ATM switch. The IP addresses in the PMA traces are all anonymized, and the anonymization algorithm is inaccessible to us. The mapping to the internal addresses is one-to-one, and does not change within each PMA capture. Typically an Abilene PMA trace is 90-second long. With the help from NLANR, we obtained some long traces (10 minutes) on our PMA box for 10 days, with 8 captures per day, in March 2005. All our experiments are based on these long PMA traces. The total size of PMA data is 19.34 GB (in TSH format, see Table 2), while the total size of raw Netflow binary data is 3.4 GB.

The rest of the paper is organized as follows. Section 2 addresses the timestamp synchronization, and Sect. 3 presents algorithms for matching masked IP address. Section 4 describes some experiments based on the integrated

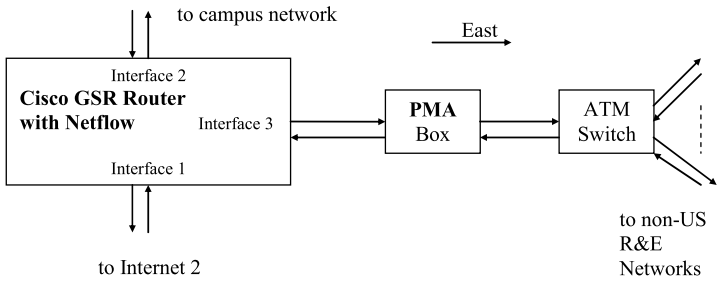


Fig. 1 Network topology and configurations

monitoring data. Related work is discussed in Sect. 5 and Sect. 6 concludes the paper.

2 Timestamp synchronization

In this section we present our approach to synchronize PMA and Netflow timestamps. To limit the processing overhead and reduce the synchronization errors step by step, this is divided into three major steps as described below in Sects. 2.1, 2.2, and 2.3, respectively.

2.1 Traffic aggregation and interface filtering

Both NetFlow and PMA traces provide information to identify the ingress or egress interface of packet/flow. To reduce the search space, we will compare the eastbound PMA traffic only with the those NetFlow records with Interface 3 as the egress interface (see Fig. 1). We will also compare the westbound PMA traffic only with those NetFlow records with Interface 3 as the ingress interface.

2.2 Coarse-grained timestamp synchronization

In PMA traces, a timestamp is associated with each captured packet header. In NetFlow, each flow is recorded with only two timestamps for the first and the last sampled packet of that flow, respectively. Based on the related Netflow record fields (i.e. `UNIX_SEC`, `UNIX_NSEC`, `SYSUPTIME`, `FIRST`, `LAST`, see Table 1), we can obtain the corresponding Unix time (in milliseconds) of the first and the last sampled packet (denoted by `NF_First_UnixTime` and `NF_Last_UnixTime`, respectively) for each flow, in terms of the router's clock. The question is, what is the difference between the NetFlow time and the PMA time? Or, given `NF_First_UnixTime` or `NF_Last_UnixTime` of a flow in Netflow, what is the corresponding PMA time? This is an important issue, since knowing the timestamp difference can significantly reduce the search space of our IP matching algorithm (see Sect. 3). As we will show below, in some cases the difference between the two can be very large, and increases/decreases slowly. Note that since the Netflow internal timestamps for the first/last flow packet are used, the Netflow setting of expiration timer is irrelevant here.

Fig. 2 Dividing NetFlow trace to 1-min segments

We first use a coarse-grained algorithm to obtain the approximate time difference (accurate to 1 minute). We observe that although IP addresses are masked, some other header fields, such as the source/destination port and the IP protocol number, are not masked (masking these fields is not necessary from the privacy point of view, and can make the monitoring data less useful), and exist in both monitoring data. Thus we have

Observation 1 Different flows from different IP addresses may have the same source or destination port numbers. However, within *different* time periods, a backbone link is likely to see *some* flows with *unique* source and destination port pairs. This information can be exploited to identify the timestamp difference between Netflow and PMA.

For example, for the Web traffic, although the server side port number is often 80, the client side port number depends on the available ports on different machines at the time of connection establishment. Since Web traffic consists of many short-lived flows, this can be exploited to differentiate different time intervals.

The coarse-grained algorithm can be illustrated by Fig. 2. NetFlow has flow records for the entire 24 hours of one day. A PMA trace captures the packets in an interval `T` of 10 minutes. Assume that `T=[PMA_t1, PMA_t2]`. That is, it starts at `PMA_t1` and ends at `PMA_t2` in terms of the PMA clock. The question is how to align `T` with the corresponding NetFlow time. We have:

Observation 2 Within `T`, all flows observed by NetFlow should also be observable in the PMA trace; Outside `T`, *not* all NetFlow records can be also observed in the PMA trace, especially on a highly multiplexed backbone link.

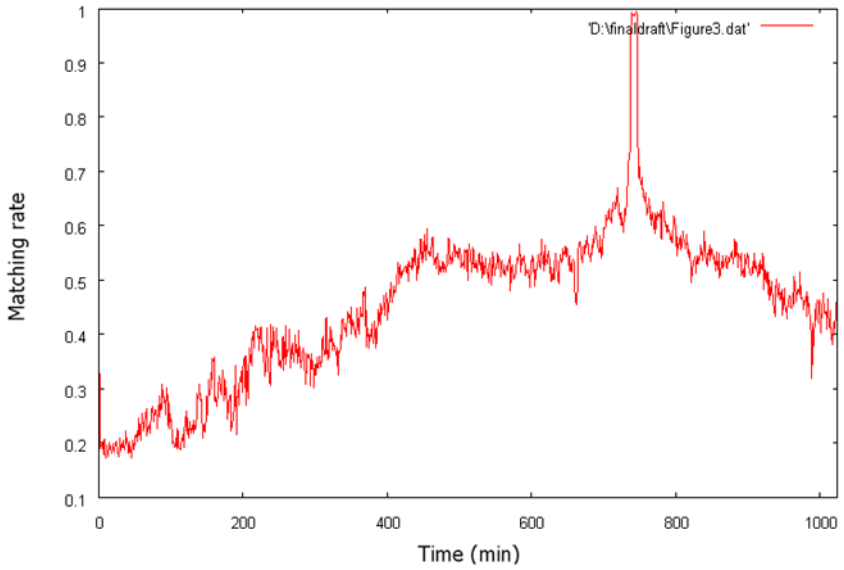


Fig. 3 Segment matching rate with one PMA trace

We thus divide the NetFlow data of one day into 1440 one-minute segments (Fig. 2), according to NetFlow timestamps. For each TCP/UDP flow record of NetFlow, a tuple (SrcPort, DstPort, IP Protocol Number) is placed into the Netflow segments covering `NF_First_UnixTime` or `NF_Last_UnixTime`. This tuple from NetFlow is said to be shared by PMA, if any packet anywhere inside the 10-minute PMA trace has a packet with the same source port, destination port and protocol number. For each 1-minute segment, we calculate the matching rate of this segment, defined by:

$$\textit{Segment\_Matching\_Rate} = \frac{\textit{Number of Tuples in the Segment Shared by PMA}}{\textit{Number of Tuples in the Segment}}.$$

The 11 *consecutive* segments with the highest average matching rate will be selected by the coarse-grained algorithm. Note that 11, instead of 10, consecutive segments are selected, because `PMA_t1`, the starting time of `T`, is not necessarily aligned with the boundary of NetFlow segments. The first and the last selected segments can have relatively lower matching rates, since they may contain flow records outside `T`. However, all the other 9 internal segments should have a matching rate close to 100%. The timestamp synchronization error, determined by difference between the starting time of the first selected segment and `PMA_t1`, is bounded by 1 minute.

Figure 3 shows the matching rates of NetFlow segments with one PMA trace of westbound traffic. The x-axis is the relative time of NetFlow in one day, while the y-axis gives the corresponding segment matching rate. A peak period of approximately 10 minutes, from 738 minute to 748 minute, can be clearly distinguished from the other segments.

The peak period in Fig. 3 is amplified in Fig. 4. We can see that the matching rates of selected segments are very close but not equal to 100%. The 9 internal segments have matching rates lower (around 98.5%) than what we have expected. To discover the reason behind this, we plotted in Fig. 5 the unmatched Netflow records in the peak period. Each dot in the figure represents an unmatched flow. The x-axis is the *NF_First_UnixTime* of the unmatched flow, while the y-axis is the protocol number in the IP header. Most of the unmatched flows have a protocol number of 1 (ICMP). The ICMP header does not have the transport layer port number. PMA and NetFlow have different (undefined) ways to fill the port number fields in their records.[1] We recalculated the matching rates after removing ICMP records. Figure 6 shows the results. Compared to Fig. 4, the peak in Fig. 6 is flat at the top, and the matching rate is very close to 100%.

However, there are still a number of TCP (protocol number 6) and UDP (protocol number 17) flows unmatched even in the 9 internal segments, as shown in Fig. 5. Note that the westbound traffic (see Fig. 1) travels through the PMA box before it arrives at the Cisco router. And yet a flow record found in NetFlow can be unmatched by PMA. This might relate to the fact that PMA uses a splitter to "divert" a small percentage of the optical energy to its detector, which in rare cases might not be able to decode the thus "artificially attenuated" packets. This finding confirms one value of our study: we may use sampled flow-level data to discover operational problems in the more-detailed packet-level traces.

[1] We found that in Cisco NetFlow Version 5, the source port number field is actually filled with ICMP message type and subtype, which are not as diverse as port numbers and cannot be used to differentiate ICMP packets.

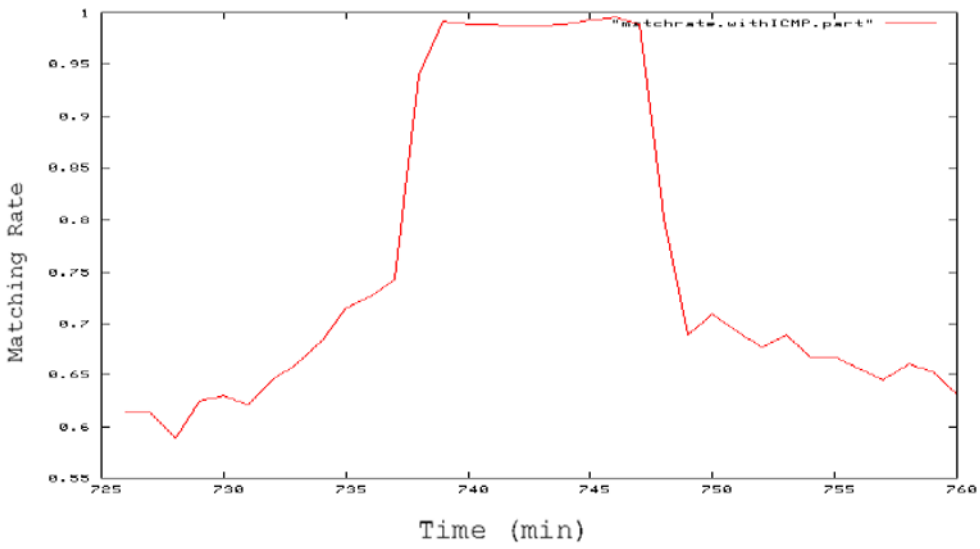


Fig. 4 Zoom in the peak period

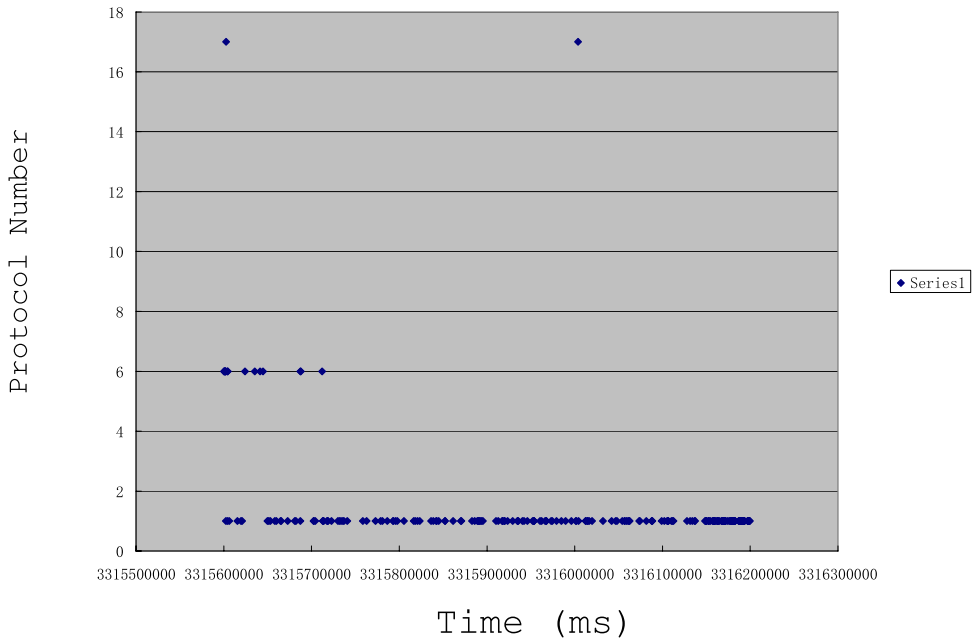


Fig. 5 The protocol number of the unmatched flows

To further validate the matching algorithm we performed another analysis. We have 8 PMA trace files captured per day, each of which is compared with the NetFlow data using our algorithm. For each NetFlow segment, Fig. 7 shows its maximum matching rate among the 8 PMA traces in one day. The x-axis is the relative time of NetFlow in one day, and the y-axis is the maximum matching rate. It can be seen that the corresponding 8 peak matching-periods can be easily identified for the 8 PMA traces.

We then further calculate the time intervals between successive PMA captures (in PMA timestamps) and the time intervals between successive peak matching periods (in NetFlow timestamps), shown in Table 3. The latter should be a multiple of 60 seconds, since the segment length is 1 minute. By comparing the column 2 and the column 3 in Table 3, we can see that the difference between the two columns is always less than 45 seconds. This verifies that our coarse-grained algorithm has an error upper-bound of 60 seconds, in highly-multiplexed backbone links. Results with the other 10 days of monitoring data are similar.

2.3 Fine-grained timestamp synchronization

Next we will reduce the estimation error in a reduced search space. Based on the coarse-grained algorithm, we are able to reduce the search space of NetFlow data to 11 segments. We further want to find a timestamp `pma_ts` associated with a PMA packet `pma_packet`, and a timestamp

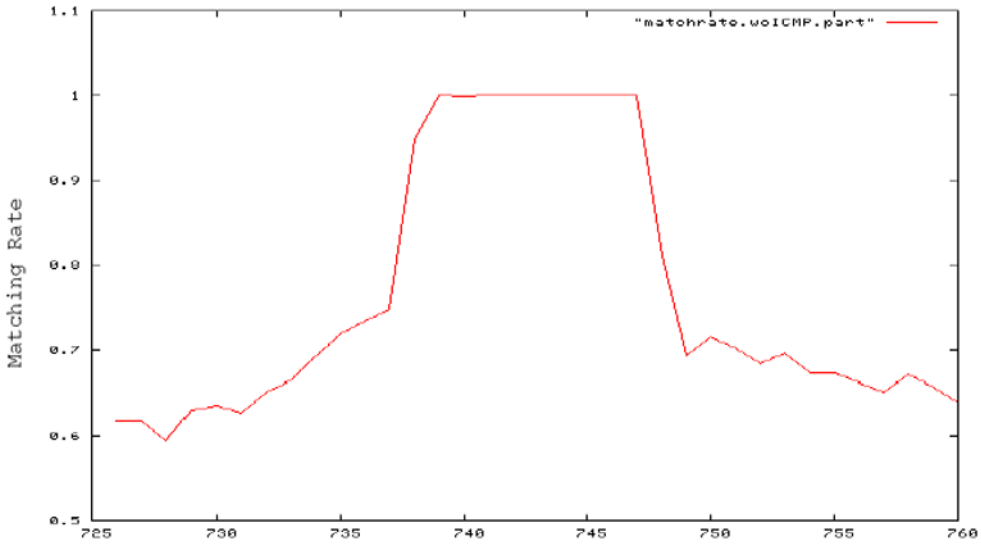


Fig. 6 Zoom in the peak without ICMP records

Table 3 The time intervals between successive PMA captures of one day

Successive PMA traces	Intervals between successive PMA Traces in PMA timestamps	Intervals between successive peak matching periods in NetFlow timestamps
1 → 2	9404 s	9420 s
2 → 3	9405 s	9360 s
3 → 4	13007 s	13020 s
4 → 5	9404 s	9420 s
5 → 6	13005 s	13020 s
6 → 7	9405 s	9360 s
7 → 8	9405 s	9420 s

`nf_ts` in NetFlow associated with a packet `nf_packet` (either the first or last sampled packet of a flow), such that `pma_packet` and `nf_packet` actually are the same packet. Then the timestamp difference between Netflow and PMA will be `nf_ts - pma_ts`. However, identifying the same packet in both PMA and NetFlow traces is difficult due to the reasons outlined in Sect. 1. The good news is, to synchronize timestamps, we just need to identify some but not all of the packets that appear in both traces.

The first/last packet of a flow *sampled* by NetFlow is not necessary the first/last packet of the flow observed by PMA. However, for TCP flows, whenever a TCP SYN segment (a TCP segment with its SYN flag set) is sampled by NetFlow, the SYN flag in the corresponding flow-level record will be turned on (see Table 1). A TCP SYN segment is the first segment of a flow in each direction (either the forward or reverse path). Thus, when the SYN flag of a NetFlow record is set, the actual first segment of the TCP flow has been sampled by NetFlow. The packet that `NF_First_UnixTime` is associated with, is actually the first packet of that flow in that direction observed by PMA. Hence we can establish an exact correspondence between the two timestamps from NetFlow and PMA, respectively. Following [3], we call these flows "*TCP SYN flows*". Note that since a TCP SYN segment might be retransmitted, we simply skip those TCP SYN segments occurring more than once in the PMA trace.

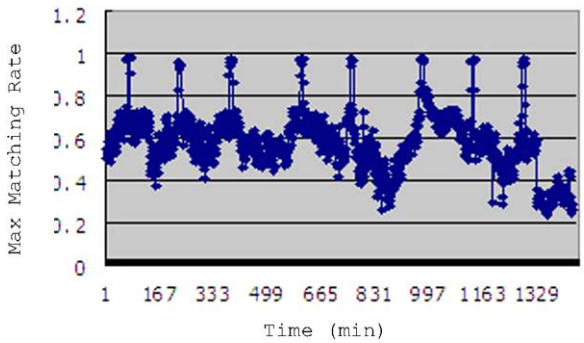
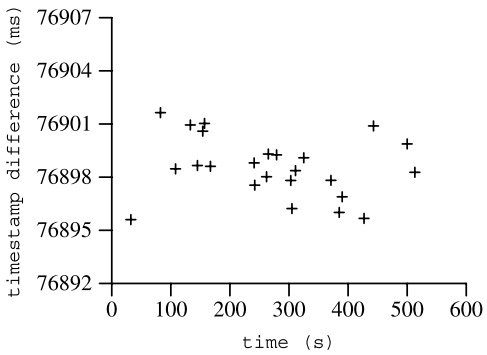
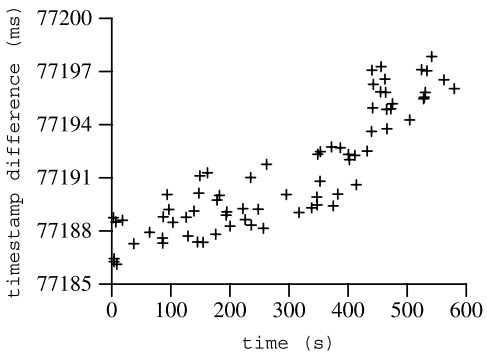


Fig. 7 Max matching rate of NetFlow segments with 8 PMA traces

The remaining question is: how can we tell whether the two flows in NetFlow and PMA, respectively, are the same flow when IP addresses are masked differently? Since we

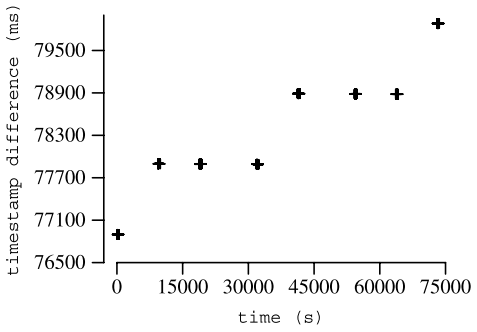


(a) west-bound traffic

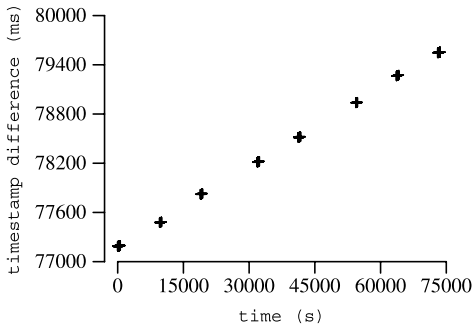


(b) east-bound traffic

Fig. 8 Timestamp differences for 1 PMA trace



(a) west-bound traffic



(b) east-bound traffic

Fig. 9 Timestamp difference for 8 PMA traces

just need to identify some but not all the packets appearing in both traces, we can look for those TCP flows who has a unique tuple (source port, destination port) in both PMA and NetFlow traces. For these flows, we can be sure that the two flows are the same flow.

The Fig. 8 below shows the timestamp differences of the two directions of traffic, respectively. Each dot in the figures represents a TCP SYN segment selected by our algorithm. The x-axis is the relative time of a 10-minute PMA trace, and the y-axis is the corresponding timestamp difference minus 18400 s.

The NetFlow time is supposed to synchronize with a NTP server and the PMA time is supposed to synchronize with a CDMA network [8]. However, as shown in Fig. 8, the absolute value of timestamp differences is unexpectedly large, and can be as high as 5 hours. After a detailed examination of monitoring data, it seems that PMA loses the CDMA time signal and uses the line CP signal instead. Even so, the traffic in both directions share a common characteristic: the range of timestamp difference is limited to a 12-millisecond scope. That is, although there is a large difference between the two timestamps, the variation of difference falls within a narrow range in a short period of PMA packet capture. This can be exploited to provide a fine-grained estimation of timestamp difference for each PMA trace with an error bounded to a number of milliseconds.

We also observe that the timestamp difference gradually increases over time with the eastbound traffic, while the timestamp difference fluctuates with westbound traffic. The fluctuation with westbound traffic may be due to the dynamics of queuing delay and the traffic load on the interface card.

To further investigate this pattern, in Fig. 9 we draw the diagram for the 8 PMA traces of one day in both directions. Each dot in the figure represents to a selected TCP SYN segment from one of the PMA trace. Notably, the range of the y-axis is 3 seconds (vs. 15 ms in Fig. 8) now. Since within each PMA trace the timestamp differences are within a narrow range of several milliseconds, the dots for the same PMA trace are clustered together as a single large point and hence not discernible in the figure. On the other hand, the variation of timestamp difference could be higher than 1 second across difference PMA traces. In Fig. 9(a) we can see that the time difference in the westbound direction jumps by 1 second every 3 traces. For the eastbound traffic shown by Fig. 9(b) the difference increases linearly with a certain rate (interestingly, a rate close to the increasing rate shown

in Fig. 8(b)) as time goes by. Regardless, the timestamp difference increases about 2 seconds per day, indicating loss of synchronization. This means that the timestamp difference obtained with one PMA trace cannot be reused for the other traces. Therefore, the fine-grained algorithm should be applied to each PMA trace to reduce the estimation error to milliseconds. We have tested our algorithm with monitoring data of the other days, and the conclusions are similar.

Thus, by applying the fine-grained matching algorithm to each PMA trace, our data-driven approach can reduce the estimation error of timestamp difference to a number of milliseconds, even when clocks lose synchronization and the absolute timestamp difference is large and slowly increases/decreases.

2.4 Algorithm discussion

Due to anonymized IP addresses, we can only use the pair of source/destination ports to identify flows and determine whether a flow record from NetFlow also appears in PMA. Obviously this may lead to "false-positives" with the coarse-grained algorithm, because different flows with different real IP addresses may have the same port pair. To be more specific, it may falsely increase the matching rates of NetFlow segments outside the scope of T. However, as we have demonstrated through real-world traffic data, it's robust enough for the purpose of coarse-grained timestamp synchronization for a highly-multiplexed network link.

One may also set the NetFlow segment size to be 10 minutes, instead of 1 minute. However, since PMA_t1 is not necessarily aligned with the boundary of NetFlow segments, a 10-minute PMA capturing interval [PMA_t1, PMA_t2] is very likely to overlap with two 10-minute NetFlow segments, each of which may have a relatively low matching rate. Hence, it will be difficult to identify them from a "noisy" background, since IP addresses are masked and only port information is used. Furthermore, with 10-minute NetFlow segments, the error bound of coarse-grained timestamp synchronization will be 10 minutes, instead of 1 minute.

In Sect. 2.2, when calculating the matching rate for each segment, we compare 1-minute of NetFlow segment against the 10-minute PMA data. An alternative approach is to first compare the PMA data with the first 10 NetFlow segments of the day, and then repeatedly remove the first NetFlow segment and compare the PMA data with the first 10 segments of the remaining NetFlow data. The PMA data are also divided into 10 1-minute segments, and in each step a NetFlow segment will only be compared with the corresponding 1-minute segment in PMA. However, again, since the boundaries of PMA segments are unlikely to be aligned with those of NetFlow segments, all the 9 internal segments of the peak period can end up with low matching rates, which makes it difficult to identify the peak matching period.

3 Matching anonymized IP addresses

Since the IP addresses of NetFlow or PMA traces are anonymized, a critical issue of integrating flow-level and packet level traces is to find the correspondence between the anonymized IP addresses so that a NetFlow record can be exactly mapped to the corresponding flow in the PMA trace. This is achievable if, within one system, the IP address masking is a one-to-one mapping between real IP addresses and anonymized internal addresses, although the exact mapping might be different with PMA and Netflow. We have developed a simple algorithm based on port pairs and the coarse-grained timestamp synchronization (see Sect. 3.1 below) and a more sophisticated algorithm to improve the matching rate, based on the fine-grained timestamp synchronization (Sect. 3.2).

3.1 Simple IP address matching algorithm

For IP address matching, a PMA trace will only be compared with Netflow flows overlapping with the 11 NetFlow segments selected by the coarse-grained synchronization algorithm described in Sect. 2.2.

In Sect. 2.3, we have shown that if the source and destination port pair is unique among flows in both NetFlow and PMA, we can conclude that the two flows in different monitoring data are actually the same flow, and two IP address mappings (source IP mapping and destination IP mapping) can be derived. If the port pair is not unique, then the exact mapping is ambiguous and we cannot decide which IP address in NetFlow should be mapped to which IP address in PMA. However, we have the following observation:

Observation 3 For a non-unique port pair with matching ambiguity, if one of the masked IP address has been inferred (e.g. because it is either the sender or the receiver of another port pair which is unique), this information can be exploited to reduce the ambiguity: the flows with the inferred IP address as the source or destination IP can be placed into a separate set for comparison and matching.

As an example, there are m flows in NetFlow and n flows in PMA that have the same port pair, where $m > 1$ or $n > 1$. If an IP address `ip_pma` from PMA has already been mapped to an IP address `ip_NetFlow` from NetFlow (e.g. due to another port pair that is unique), then we can divide the flow set into three subsets:

(a) NetFlow flows with `ip_NetFlow` as source IP, and PMA flows with `ip_pma` as source IP
(b) NetFlow flows with `ip_NetFlow` as destination IP, and PMA flows with `ip_pma` as destination IP
(c) NetFlow flows without `ip_NetFlow`, and PMA flows without `ip_pma`

Table 4 Matched IP addresses in each iteration

Rounds	Number of pending NetFlow records	Number of matched IP addresses
0	13091	0
1	7755	2999
2	7477	3098
3	7460	3101
4	7460	3101

As can be seen, no PMA flow in one of the subset can be matched with a Netflow flow in a *different* subset. Therefore, matching can be performed within each subset. With the number of the flows to be compared reduced, potentially more mappings can be discovered, if any of these subset has only one flow from NetFlow and one flow from PMA.

Our IP matching algorithm thus scans the list of port pairs from NetFlow (since NetFlow usually has fewer port pairs than PMA) and applies the above mechanism. It may happen that IP address mapping discovered later on in a scan can be used to remove the ambiguity with a non-unique port pair early in the list. Therefore, our algorithm will run several rounds to match as many IP addresses as possible. After each round, the number of IP addresses matched may increase. The algorithm will terminate when the number of inferred IP addresses stops growing.

Table 4 gives the results with a PMA trace and 11 NetFlow segments. The round 0 shows the total number of flow records in NetFlow. The following rows show the number of NetFlow records that have at least one IP address unmatched, as well as the total number of matched IP, after each round. The 2nd and the 3rd rounds improves the matching rate by 3%. The algorithm converges at the end of the 3rd round, and only 43% of the NetFlow records can be matched. This matching rate is not sufficient, and hence we improve the matching rate further using the fine-grained synchronization.

3.2 Sophisticated IP address matching algorithm

The basic idea of improvement is as follows. With the fine-grained synchronization algorithm, we can derive the timestamp differences between PMA and NetFlow, accurate to several milliseconds. Our observation is:

Observation 4 Although it is likely that multiple flows with anonymized IP addresses use the same port pair, it is less likely that they send packets around the same time. This is especially true for short flows with a few packets, which is typical in web traffic.

Given a NetFlow record, we know the timestamps of its first and the last sampled packets: `NF_First_UnixTime`, and `NF_Last_UnixTime`, respectively. When we search for the corresponding PMA flow, we will look at those PMA flows that not only have the same port pair, but also send packets around the PMA time corresponding to `NF_UnixTime` (either `NF_First_UnixTime` or `NF_Last_UnixTime`). To be conservative with the estimation accuracy, the corresponding PMA timestamp shall fall in the range of

$$[NF\_UnixTime + (ts\_diff\_min - d),$$
$$NF\_UnixTime + (ts\_diff\_max + d)]$$

Here we assume that the timestamp difference extracted from TCP SYN segments is in the range `[ts_diff_min, ts_diff_max]` (see Fig. 8), and `d=ts_diff_max-ts_diff_min`. `[ts_diff_min-d, ts_diff_max+d]` is a relaxed estimation of the range of timestamp differences.

This *additional* requirement is called "*timing constraint*". Notably, once the accurate timestamp difference is derived based on TCP SYN flows (Sect. 2.3), the timing constraint can be applied to all TCP/UDP flows, and even packets of protocols without port numbers, such as ICMP. Since ICMP record does not have port numbers, the ICMP packet size (available if the total number of packets with an ICMP record is 1) is used to reduce the search space. Figure 10 gives the overall algorithm. We've added a verification mechanism into the algorithm: If the same IP address in PMA is matched to different IP addresses in Netflow, or vice versa, the inconsistency will be reported.

In Table 5 we present the results of our algorithm with one PMA trace. The total number of PMA packets is 10,002,842, while total number of Netflow records is 6676. We define the matching rate as the percentage of Netflow flows whose PMA packets can be identified by inferring the correspondences of source and destination IP addresses. The overall matching rate is 93%, while the matching rate for TCP is 95%. Notably, without the timing constraint, the overall matching rate drops to 41.9%. The significant improvement with the timing constraint is due to many short flows in the traffic. In addition, some of the unmatched NetFlow records are due to the missing port pairs in PMA (i.e. PMA traces may miss some packets that pass through them, see discussions in Sect. 2.2), rather than the matching ambiguity.

We also tested our algorithm with monitoring data on different days with different traffic loads, and the results are similar. More importantly, no mapping inconsistency has been reported.

```
do {
    for (each flow record in NetFlow) {
        Based on Observation 3, reduce matching ambiguity by examining the inferred IP addresses.
        If (an ICMP record) {
            If (the number of packet is 1)
                If (only one ICMP packet in PMA satisfies the timing constraint and meanwhile has the
                same packet size) {
                    Establish a mapping between src IP addresses; Establish a mapping between dst
                    IP addresses;
                    Report a mapping inconsistency, if any;
                }
        } else
        If (a TCP/UDP flow) {
            If (only one flow in PMA has the same port pair and meanwhile satisfies the timing constraint){
                Establish a mapping between src IP addresses; Establish a mapping between dst IP
                addresses;
                Report a mapping inconsistency, if any;
            }
        }
    }
} while (the number of mapped IP addresses is still increasing);
```

Fig. 10 IP-address matching algorithm with fine-grained timestamp synchronization

Table 5 Experiment results: IP address inference

	Total number of netflow flows	Successfully matched flows
Entire NetFlow	6676	93% (without timing constraint: 41.9%)
TCP	5091	95%
UDP	1272	89%
ICMP	223	75%
Others	90	82%

Table 6 Comparison of total number of packets

	PMA	NetFlow	PMA/NetFlow ratio
Total Packets	4174476	39953	104.48
Westbound TCP packets	971564	9744	99.71
Eastbound TCP packets	946961	9545	99.21
Westbound UDP packets	231511	2334	99.19
Eastbound UDP packets	1996258	18065	110.50

4 Integrating flow-level and packet-level monitoring data: cross-validation

With the accurate timestamp synchronization and the IP address mapping table, we can integrate and compare the related information in both packet-level and flow-level monitoring data. We now demonstrate the results of cross-validation, as an example.

We studied the total number of packets going through the OC-3 link (see Fig. 1) within a 10-minute PMA capturing time, based on the results of our timestamp synchronization algorithm. We compared the estimated numbers based on Netflow and PMA, respectively. Shown in Table 6, mostly the numbers of packets from PMA is around 100 times the NetFlow numbers, reflecting that fact that the NetFlow sampling interval is set to 100 packets. However, somewhat surprisingly, some of the PMA numbers are much higher than 100 times the corresponding NetFlow numbers, especially for the eastbound UDP traffic.

To investigate the reason behind this, we then examined the PMA/NetFlow packet ratios at flow-level, thanks to our IP matching algorithm described in Sect. 3. Note that a sampling interval of 100 packets on the entire traffic does not necessarily mean that each flow's PMA/NetFlow packet ratio will be close to 100, since the ratio with each flow depends on the flow size [3]. However, it helps us to narrow down the scope and pinpoint the root cause. We identified those UDP flows that have abnormally high PMA/Netflow ratios and have sent large number of packets that can significantly affect the ratio of total traffic. We found that all those UDP flows are actually multicast traffic that came in

Table 7 Comparison of total number of packets (multicast traffic incorporated)

PMA traces	Total PMA packets	Total NetFlow packets	PMA/netflow ratio
1	5313018	53232	99.81
2	7992868	79434	100.62
3	8881071	89190	99.57
4	9609340	95639	100.48
5	8165544	81299	100.44
6	4174479	41863	99.72

from one port and went out to multiple ports. However, the output interfaces of these multicast flows are not correctly marked in Netflow records. Hence, with this cross-validation between flow-level and packet-level records, we discovered that multicast traffic is not correctly incorporated in Netflow Version 5, which can be verified by the Netflow manual.

Table 7 illustrates the results with 6 PMA traces, after the multicast traffic has been included into the NetFlow packet counts. The PMA/NetFlow ratio is very close to 100, which is the configured sampling interval of NetFlow. We've also examined the estimations of total number of bytes, based on PMA and Netflow respectively. The results are similar.

We further study the accuracy of NetFlow total packet estimation in shorter time intervals. For example, if we divide the 10-minute capturing time into 10 intervals of size 1-minute, what will be the average accuracy of NetFlow estimation in each interval? We use the PMA data as the "base-line" and examine how much NetFlow-derived estimation deviate from the baseline. We are interested in how small this time interval can be, given accuracy requirements on the NetFlow-derived packets estimation (we hope this interval is as small as possible so that we can understand the "real-time" status of the network). The estimation error rate is defined as

$$(\texttt{Packet\_Number\_in\_PMA} - \texttt{Packet\_Number\_in\_NetFlow} * 100)/\texttt{Packet\_Number\_in\_PMA},$$

where `Packet_Number_in_PMA` is assumed to be accurate, due to the 1:1 sampling ratio. The NetFlow estimation error will increase with a smaller time interval, to which the error of timestamp synchronization will be more significant. In addition, we assume that the flow packets sampled by NetFlow are uniformly distributed within `[NF_First_UnixTime, NF_Last_UnixTime]`, which may lead to non-negligible error when the time granularity is small. As shown in Table 8, when the time interval decreases, the error rate will increase, and in general the interval should be longer than 30 seconds, if the error rate is required to be less than 5 percent.

Table 8 Estimation accuracy of NetFlow on total number of packets

Duration	Max error rate % (westbound)	Max error rate % (eastbound)
10 s	9.1591	7.1555
30 s	4.8804	5.1238
60 s	4.3041	4.2317
120 s	2.4096	0.7985

5 Related work

The periodical sampling of NetFlow can be improved for advanced traffic analysis. Estan et al. [4, 5] improves NetFlow based on the observation that a small number of "heavy hitters"' accounts for a large share of traffic. It introduces a scheme that concentrates only on large flows. Zhang et al. [17] focuses on online identification of 1-D and 2-D hierarchical heavy hitters. Estan et al. [6] proposes an adaptive NetFlow, which dynamically adapts the sampling rate to achieve robustness without sacrificing accuracy. Kumar et al. [7] presents a novel data streaming algorithm providing much more accurate estimates of flow distribution. These works focus on how to improve NetFlow. However, a NetFlow optimized for one purpose (e.g. identifying top flows) may not be sufficient for other analyses.

Sommer and Feldmann [15], Duffield, Lund and Thorup [2], Duffield and Lund [1], Duffield, Lund and Thorup [3], Mori et al. [11] perform analysis based periodically packet-sampled flow-level records. Duffield, Lund and Thorup [2] discusses how to infer the traffic properties from the packet-sampled flow statistics. Duffield, Lund and Thorup [3] provides methods to infer the absolute frequencies of flow lengths in the unsampled stream. The TCP SYN flag is utilized to estimate original TCP flows. Duffield and Lund [1] samples NetFlow records to further reduce the processing overhead. Mori et al. [11] investigates how to identify elephant flows based on Bayes' theorem. Our work complements these works by integrating packet-level traces and performing cross-validation. Furthermore, the system characteristics (rather than the sampling characteristics only) of network monitoring devices are captured. We've also considered anonymized IP addresses.

Traffic analysis based-on packet-level traces is also an important direction. McGregor et al. [9] utilizes the timestamps in packet-level traces and the packet inter-arrival time (IAT) to classify applications.

Synchronizing clocks or timestamps is a well-studied area. Veitch, Babu and Pasztor [16] studies synchronizing the software clock with the standard time. Paxson [13] and Moon, Skelly and Towsley [10] remove clock skews in delay measurements. The goal of our timestamp synchronization algorithm is to integrate packet-level and flow-level records.

Our approach is data-drive, rather than based on models of clocks or transit delays.

Finally, many research works have been done in data-centric information processing, such as Rupp et al. [14], and Estan, Savage and Varghese [6].

6 Conclusion and future work

In this paper, we investigated the integration of flow-level records, exemplified by Cisco NetFlow, and packet-level traces, exemplified by NLANR PMA. To integrate heterogeneous monitoring data, we first synchronize their timestamps, and then match their masked IP addresses. Our key observation is that although the IP addresses are masked, some other header fields can be exploited to match different types of monitoring data. In our future work, we will apply the data mining techniques on both flow- and packet-level information, possibly using the PMA data as training sets to set up a Neural Network learning model and then using the learned model and NetFlow data to estimate the real-time status of the network.

References

1. Duffield, N., & Lund, C. (2003). Predicting resource and estimation accuracy in an IP flow measurement collection intrastate. In *ACM internet measurement conference*, October 2003.
2. Duffield, N. G., Lund, C., & Thorup, M. (2002). Properties and prediction of flow statistics from sampled packet streams. In *ACM internet measurement workshop*, November 2002.
3. Duffield, N., Lund, C., & Thorup, M. (2003). Estimating flow distributions from sampled flow statistics. In *ACM SIGCOMM*, August 2003.
4. Estan, C., Keys, K., Moore, D., & Varghese, G. (2002). Building a better NetFlow. In *ACM SIGCOMM*, August 2002.
5. Estan, C., Keys, K., Moore, D., & Varghese, G. (2002). New directions in traffic measurement and accounting. In *ACM SIGCOMM*, August 2002.
6. Estan, C., Savage, S., & Varghese, G. (2003). Automatically inferring patterns of resource consumption in network traffic. In *ACM SIGCOMM*, August 2003.
7. Kumar, A., Sung, M., Xu, J., & Wang, J. (2004). Data streaming algorithms for efficient and accurate estimation of flow distribution. In *ACM SIGMETRICS*, June 2004.
8. Micheel, J., Donnelly, S., & Graham, I. (2001). Precision timestamping of network packets. In *ACM internet measurement workshop*, November 2001.
9. McGregor, A., Hall, M., Lorier, P., & Brunskill, J. (2004). Flow clustering using machine learning techniques. In *Passive and active measurement workshop*, April 2004.
10. Moon, S. B., Skelly, P., & Towsley, D. (1999). Estimation and removal of clock skew from network delay measurement. In *IEEE INFOCOM*, March 1999.
11. Mori, T., Uchida, M., & Kasahara, R., et al. (2004). Identifying elephant flows through periodically sampled packets. In *ACM internet measurement conference*, October 2004.
12. Cisco NetFlow. http://www.cisco.com/warp/public/732/Tech/nmp/NetFlow/.
13. Paxson, V. (1998). On calibrating measurements of packet transit times. In *ACM SIGMETRICS*, June 1998.
14. Rupp, A., Dreger, H., Fedlmann, A., & Sommer, R. (2004). Packet trace manipulation framework for test labs. In *ACM internet measurement conference*, October 2004.
15. Sommer, R., & Feldmann, A. (2002). NetFlow: information loss or win. In *Internet measurement workshop*, November 2002.
16. Veitch, D., Babu, S., & Pasztor, A. (2004). Robust synchronization of software clocks across the internet. In *ACM internet measurement conference*, October 2004.
17. Zhang, Y., Singh, S., Sen, S., Duffield, N., & Lund, C. (2004). Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *Internet measurement conference*, October 2004.

Chi Zhang is a Senior Kernel Engineer at Juniper Networks. He was an Assistant Professor of Computer Science at Florida International University from 2003 to 2006. He received the B.E. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 1996, and the Ph.D. degree in Computer Science from Northeastern University, Boston, MA, in 2003.
Dr. Zhang's research interests lie in the areas of network protocols, congestion control, mobile computing and QoS. He has published 28 papers and issued 1 US patent. He served on a number of program committees of networking conferences, and is on the editorial board of the Journal of Internet Engineering. Dr. Zhang received the runner-up award in the 7th IEEE Symposium on Computers and Communications (ISCC 2002), and is a member of the Phi Kappa Phi Honor Society.

Bin Liu is a software engineer at Microsoft. He received his B.S.E.E from Xi'an Jiao Tong University in 1997. From 2004 to 2006, he was a gradate student at Florida International University.

Xun Su received the B.S.E.E from the University of Electronic Science and Technology of China in 1992, the M.S.E.E from Southeast University in 1995 and the Ph.D. in Electrical Engineering from the University of Texas at Austin in 2002 with a focus on dynamic network routing algorithms.
He was a research engineer with the High Energy Physics department at California Institute of Technology. His research interests include network protocol design, wireless networking, and network measurement. He is now a senior software engineer with Fulcrum Microsystems Inc., a 10Gbps Ethernet chip company in Calabasas, California.

Heidi Alvarez received her Ph.D. from Rotterdam School of Management, Erasmus University, Netherlands. She is currently the Director of the Center for Internet Augmented Research and Assessment (CIARA) at Florida International University (FIU). Heidi is the PI for the Global CyberBridges CI-TEAM three year implementation project (www.cyberbridges.net). She has served as Co-PI for the AMPATH International Exchange Point since April, 2000 and as Co-PI for the Inter-regional Grid Enabled Center for High Energy Physics Research and Educational Outreach at FIU (CHEPREO) since 2003, as well as the Western Hemisphere Research and Education Network (WHREN)—Links Interconnecting Latin America (LILA) International Research Network Connections program since 2005. Dr. Alvarez also participates on the NSF UltraLight and PlaNets projects lead by Caltech.

Julio Ibarra is the Executive Director of the Center for Internet Augmented Research and Assessment (CIARA)—a State of Florida Type-II research center at Florida International University. Operating from an annual budget of $4M from contracts and grants awards, Ibarra heads the CIARA center, which is focused on contributing to the pace and the quality of research at FIU through the application of advanced cyberinfrastructure. He is responsible for the strategic planning and development of advanced research networking services for the University, the regional GigaPOP and the AMPATH International Exchange Point for Research and Education networks to enable US e-Science initiatives in South and Central America, Mexico and the Caribbean. He is the Principal Investigator of the Western Hemisphere Research and Education Network (WHREN)—Links Interconnecting Latin America (LILA), NSF-OCI International Research Networks Connection award, and the AmericasPATH (AMPATH) project. He is Co-PI of the Global CyberBridges CI-TEAM three year implementation project (www.cyberbridges.net) which began in October, 2006. He is PI and Co-PI on several other NSF grants to broaden participation in science and engineering research and education through the application of cyberinfrastructure, and participates on the NSF UltraLight and PlaNets projects lead by Caltech.