

Evaluating The Performance of ANGEL Flow Meter

Thuy T.T. Nguyen

Centre for Advanced Internet Architectures, Technical Report 070228G

Swinburne University of Technology

Melbourne, Australia

tnguyen@swin.edu.au

Abstract—This technical report gives detail on the results obtained from evaluating the ANGEL Flow Meter (FM) performance. Evaluation metrics include packet loss, CPU and memory usage and the robust of the hash algorithm employed. It has been shown that the FM's suffered packet loss (more than 10%) when coping with approximately more than 30,000 packets per second and 30,000 flows per second. There has been a small number of collisions found for the hash generating mechanism when dealing with millions of flows coming from a small range of source IP addresses ($\sim 0.015\%$ per 1million unique flows - which is considered to be acceptable for the deployment of the hash algorithm in the FM architecture). We also tested the underlying redundancy system with the deployment of CARP protocol. Test results showed a handover time of ~ 3 seconds in case of master FM power failure.

I. INTRODUCTION

The ANGEL Flow Meter (FM) is one of the three main components of the ANGEL architecture. It is responsible for monitoring a copy of all network traffic at a monitor point in the network, filtering captured traffic into individual flows, and forwarding packet (timestamp, IP packet size) and flow (five-tuple flow identification: source and destination IP addresses, ports and protocol) information to the ANGEL Flow Classifier.

In order to minimise the volume of traffic delivered to the flow classifier, a 32-bit hash of the five-tuple flow information values is used as flow identification. The SuperFashHash algorithm [1] is employed in the current FM implementation.

As the output of the FM will be used as input for the flow classifier, it is required to meet a certain *Packet Loss Rate (PLR)* and precision in time-stamping measurement.

From February 2007 and July 2010 this report was a confidential deliverable to the Smart Internet Technologies CRC. With permission it has now been released to the wider community.

A PLR of 0% is desirable so that classifier's performance would not be affected.

Clock rate and time-stamping precision at the meter is also of great importance as it is used to compute the statistical properties of the traffic flow. However, this is dependent on the hardware used and is independent of the FM implementation. The FM does not modify the captured packet timestamps.

The requirement of maintaining a low PLR becomes a challenge when the meter has to capture the traffic on the fly. Depending on the location of ANGEL components at the ISP site and the packet rate at the traffic aggregation point, the flow meter must be able to handle high-rate live traffic capture with (tens or hundreds) thousands of concurrent flows. For practical deployment purpose, this needs to be done within the constraints of physical resources.

This is also important to evaluate the performance of the hashing algorithm, with regards to the collision rate of the flowID while dealing with large number of concurrent traffic flows, also the spreading of hash values generated throughout the whole available key space.

As the first monitor point of the ANGEL architectures, it is critical for the FM to response quickly in case of hardware failure. This necessitates redundancy meter(s) in case of the main meter's failure. However, we only expect this to happen very rarely, therefore, the swapping time is reasonably acceptable if it is less than 30 seconds.

These requirements give rise to our proposal that an ANGEL flow meter must precisely capture and process live traffic in the face of a number of constraints:

- The flow meter must be able to handle high traffic rate in terms of packets per second (PPS).

The reason for prioritising the packet rate is that the FM only reads and processes each packet's IP header, not the whole packet's payload, the packet rate in terms of PPS, therefore is more important

than the bit rate.

- The flow meter must be able to handle high number of concurrent traffic flows.
- The flow meter must maintain a low PLR and precise packet time-stamping.
- The flow meter must response quickly to hardware failure.
- The hashing algorithm must exhibit a low collision rate, with good distribution of hashed values (preferably uniformly distributed).

The time-stamping accuracy evaluation of the FM hardware have been evaluated and reported in [2].

In this report, we outline the results obtained from testing the performance of ANGEL Flow Meter (FM) in terms of PLR, CPU/memory usage vs. high traffic rate/high flow rate. It also evaluates the robustness of the hash algorithm FM deployed.

II. EQUIPMENT AND SETUP

The test setup for the first two performance tests (PLR, CPU and Memory usage vs. high traffic rate (Packets Per Second PPS) and high flow rate (number of concurrent hosts and flows)) is illustrated in Figure 1.

Major components of the test include the Traffic Generator, The FM and The Flow Classifier (FC), in which the FC is just functioning as a receiver for the FM's output.

The test setup is illustrated in Figure 1

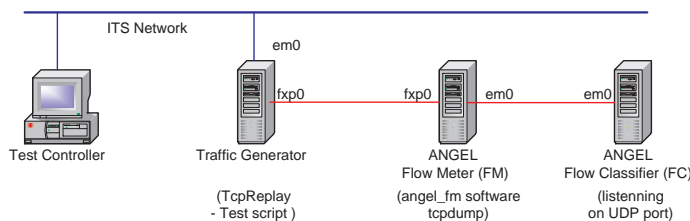


Fig. 1. Test Setup

Test components specifications and configurations can be found in Table I. To reduce the impact of limited default PCAP buffer space settings (default buffer size and maximum buffer size are 4KB and 512KB respectively) at the FM while capturing live traffic, the PCAP default and maximum buffer sizes have been reset to 512KB and 1MB respectively as suggested in [3].

A. High Data Rate Test

To test the performance of the FM while dealing with high data rate, we create a UDP flow with 200,000 packets, packet inter-arrival time (IAT) of 1msec (1000PPS)

TABLE I
TEST COMPONENT SPECIFICATIONS/CONFIGURATIONS

Test Component	Detail Specifications/ Configurations
Flow Meter	Intel Pentium 4 3.00GHz with Hyper-Threading, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1
PCAP	lipcap release version 0.9.5
PCAP Buffer Setting	sysctl net.bpf.bufsize=524288, sysctl net.bpf.maxbufsize=1048576
Angel_FM	Angel_FM as of 26 Oct. 2006
Traffic Generator	Intel Pentium 4 3.00GHz with Hyper-Threading, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1
TcpReplay	version 2.0
Flow Classifier	Intel Celeron 2.8GHz, 1GB (2 x 512MB) DDR2 533 RAM, Seagate ST380817AS 80GB SATA HDD, Asus P5LD2-VM motherboard, Running FreeBSD 6.1

and packet length of 64 bytes (included 4bytes CRC header) using the SmartBit [4]. The recorded tcpdump capture of the UDP flow is replayed at the Traffic Generator with different packet rates using the TcpReplay tool [5]. The traffic rates configured for the test are from 1000 to 50,000 PPS, with increasing step size of 5000 PPS for packet rates ≥ 5000 PPS.

Each test is repeated three times. During the data transfer, CPU and memory usages are recorded every one second for analysis.

B. High Flow Rate Test

To test the performance of the FM while dealing with high flow rate, we simulate a scenario of 1 to 50 concurrent IP hosts. Each host runs 50 UDP sessions sequentially with aggregate packet IAT of 0.5msec (64byte-packets with packet IAT of 0.5msec - total data rate of ~ 1 Mbps per host). Traffic from each host is generated using SmartBit and captured using tcpdump at the Traffic Generator. Each tcpdump file contains 50,000 packets for 50 UDP flows from each host.

To simulate 1 to 50 concurrent hosts, TcpRewrite (embedded in TcpReplay tool) is used to re-write the source IP addresses in each tcpdump file, and MergeCap [6] is used to join 1 to 50 tcpdump files respectively with concurrent timestamps.

Each test is repeated three times. During the data transfer, CPU and memory usages are recorded every

half a second for analysis.

III. TEST RESULTS AND ANALYSIS

A. FM performance with High Traffic Rate

1) *Packet Loss*: At the FM, there are three different steps in processing an incoming IP packet. They are packet sniffing (PCAP capture), packet header parsing (FM only processes IP packets and drops non-IP packets), and placing packet information in a queue for a UDP packet to be delivered to the FC. The output packet is referred to as a SIM (Summary of Information Message) packet in this report. To make the data communication more efficient, the FM doesn't send out information of each packet immediately after it finishes processing the packet but accumulates the packet information until they fill up a maximum segment size-UDP packet of 1500 byte long - With the current FM implementation, one full size UDP SIM packet can contain information of 97 incoming packets).

The processing steps are described in Figure 2

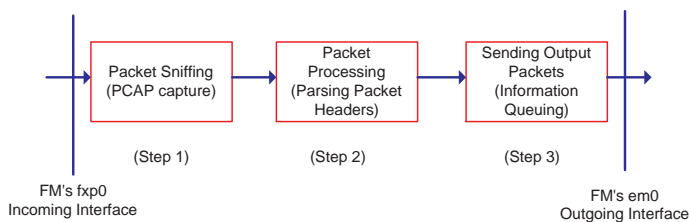


Fig. 2. Packet Processing Steps at the FM

The first evaluation metric is the amount of packet loss due to packet processing at the FM. This includes the packet loss by buffer overflow in the PCAP capturing in step 1, and possible packet loss due to the FM implementation in steps 2 and 3. (Packets that are dropped as not being IP packets in step 2 are not counted as packet loss in the evaluation).

In order to measure the number of packets lost due to PCAP buffer overflow in step 1, we inserted some lines of code into the FM program, so that upon receiving the user termination signal, the PCAP thread will display both the number of packets it received and dropped due to buffer overflow.

To measure the number of packets being parsed by the FM, tcpdump is run at the outgoing interface (em0 - toward the classifier). SIM packets are captured and analysed for the total numbers of IP packets processed by the FM. This number is compared to the number of packets received reported by the PCAP thread. This

indicates the total number of packet dropped in Steps 2 and 3.

While there is no exact solution to measure the number of non-IP packets being dropped in Step 2, we use an estimate approach. Tcpdump is run at the incoming interface (fxp0 - from the traffic generator) to capture all the user's traffic. The dump file is then filtered for the ratio of non-IP packets and total packets. This number serves as the upper bound for the packet loss rate due to non-IP packets in the above measurement.

We define a number of parameters as following. The *Number of packets to be processed* is the difference between the PCAP reported received packets and the PCAP reported dropped packets. The *Number of packets actually being processed* is the number of packets calculated by parsing the SIM packets captured at the outgoing interface of the FM.

Then, the total packets lost in Steps 2 and 3 is the ratio of *Number of packets actually being processed* to the *Number of packets to be processed*.

For the first few test trials, we found that even with low packet rates of 1000PPS to 25000PPS (no packets dropped in step 1), there was approximately 1% of packet loss in steps 2 and 3. Especially, there weren't any non-IP packets being captured in the incoming interface.

The reason for the test findings was found due to a small bug in the FM's implementation in step 3. While putting packet's information in the queue, 97 incoming IP packets' information would fill up one full-UDP SIM packet, however, the 98th IP packet's information was left out and not being exported by the FM. So the packet loss for this step is roughly 1 every 98 incoming IP packets ($\sim 1.02\%$) or 1 every outgoing SIM packet.

This section reports the test results after the bug was fixed.

Figure 3 shows the PLR due to PCAP buffer overflow in step 1 at the FM. It is plotted as a function of the incoming data rate (PPS).

As can be seen in the figure, the FM does not drop any packet for packet rates up to 25000PPS. For higher packet rates (from 30000PPS to 50000PPS), the packet loss rate increases significantly - to greater than 50% for packet rate of 50000PPS.

Figure 4 shows the total packet to-be processed and actually been-processed in steps 2 and 3 as discussed previously.

As can be seen in the figure, it seems that there is no packet loss at Steps 2 and 3 of the process. Looking more closely, we found that there is still a small amount of packet loss, the majority being less than 100 packets

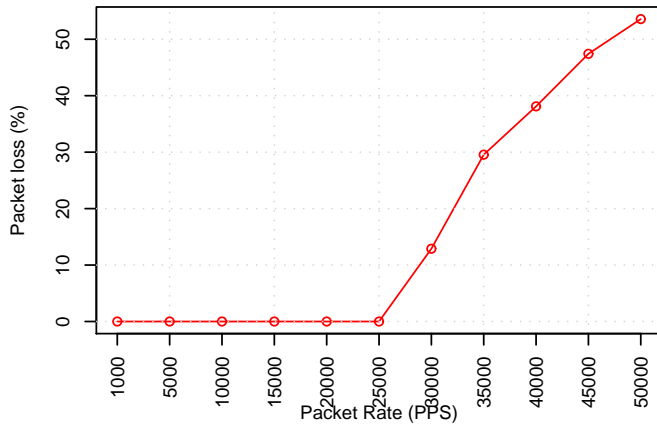


Fig. 3. Packet loss rate due to PCAP buffer overflow

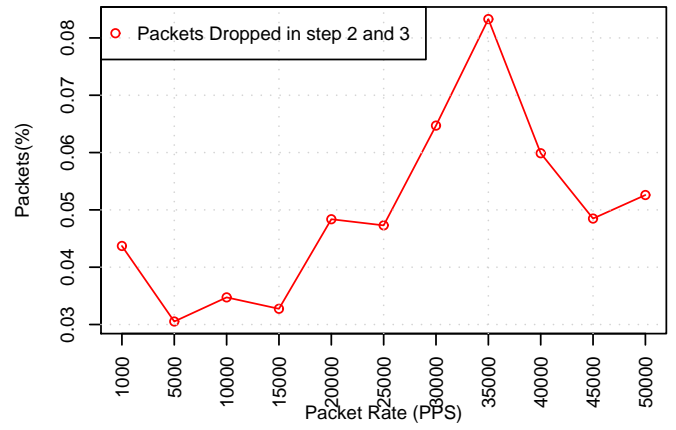


Fig. 5. Packet loss rate in Step 2 and 3 processing

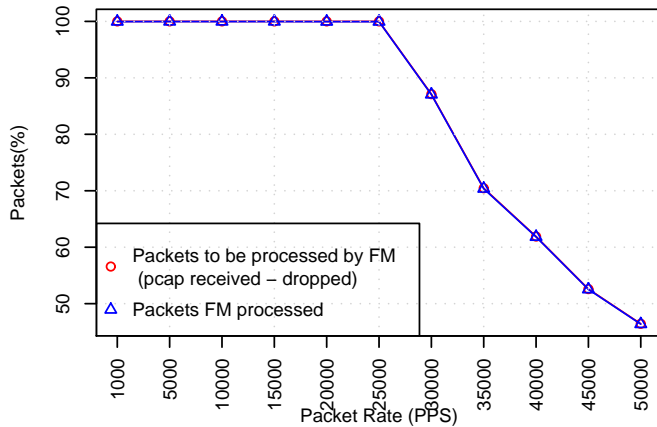


Fig. 4. Packets to-be/actually-being-processed in Step 2 and 3 processing

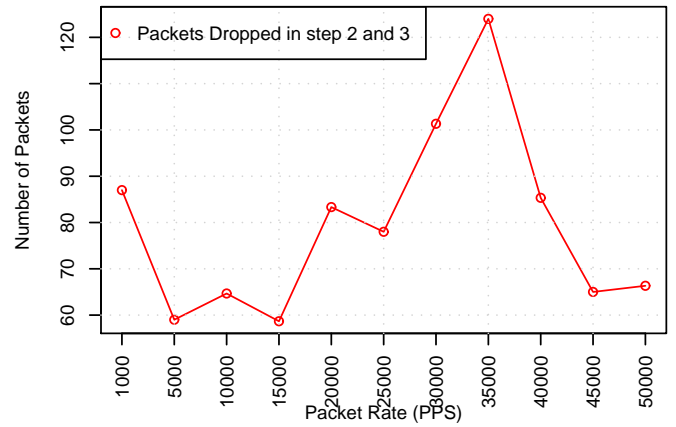


Fig. 6. Packet loss in Step 2 and 3 processing

(less than 0.08% out of total PCAP received packets). We speculate that this might be due to missing the last one or two un-full SIM packets when terminating the test. The results in percentage and absolute number of packets are shown in Figure 5 and 6.

This leads to the conclusion that with the hardware used, the FM can be used to capture and process traffic at rates of up to 25000PPS without losing any packets.

2) *CPU/Memory Usage*: The FM is designed as a multi-threaded application, including packet capture, packet processing, network and primary threads. The packet capture thread is responsible for capturing a packet, making a copy, and then queuing it for processing by another thread. The packet processing thread pulls packets captured by the capture thread out of the queue, and generating statistics to send to the FC. This information is then queued for the network thread to manage the delivery of SIM packets to the FC. The primary thread is responsible for periodically polling and updating the

database, as well as garbage collection. Details about the function and implementation of these threads are covered in [7].

This section reports on the CPU and memory usage of the ANGEL FM application when capturing and processing traffic at different data rates. By using the 'top' command with -H option every one-second during the test, CPU usage and Memory size are recorded for different thread states of the application for analysis.

Figures 7 and 8 show the memory usage of the FM capture and processing threads for different packet rates. Test results are illustrated using boxplot tool provided by R project [8]. The black line in the box indicates the median; the bottom and top of the box indicates the 25th and 75th percentile, respectively. The vertical lines drawn from the box are whiskers. The upper cap is drawn at the largest observation that is less than or equal to the 75th percentile + 1.5*IQR (interquartile range - which is essentially the length of the box). The lower cap is drawn at the smallest observation that is greater than or

equal to the 25th percentile - $1.5 \times \text{IQR}$. Any observations beyond the caps are drawn as individual points. These points indicate outliers [9].

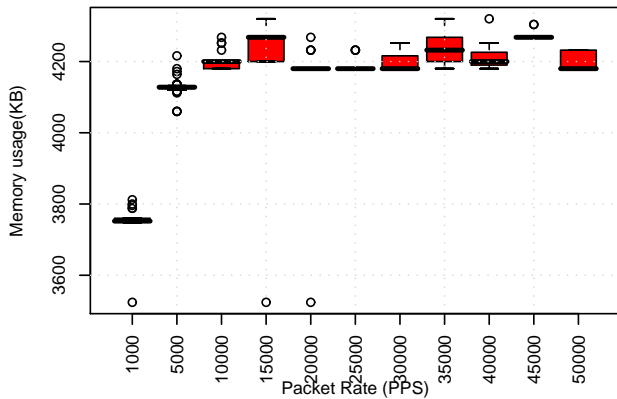


Fig. 7. Memory Usage for capturing thread

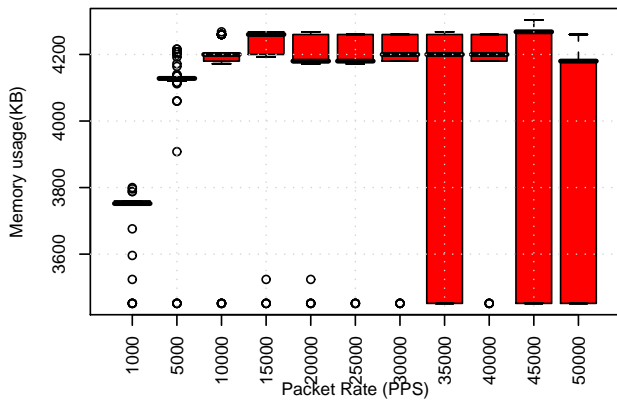


Fig. 8. Memory Usage for processing thread

As can be seen in Figures 7 and 8 and comparing to Figure 3, there is no strong correlation between memory usages and packet loss rate. This is expected, as the FM does not response to packet loss. The memory usage increased significantly when the packet rate increased from 1000PPS to 5000PPS. However, for traffic rates of higher than 5000PPS, it varied at less than 5MB.

Figures 9 and 10 show the CPU usage of the FM capturing and processing threads for different packet rates.

As can be seen in the figure, CPU usage for both packet capturing and processing threads increased slightly as the packet rate increased. However, up to 50,000PPS, the median CPU usage remained less than 20% and 10% respectively for each individual thread.

Another finding of note is that the maximum CPU usage was much greater than the median values in most

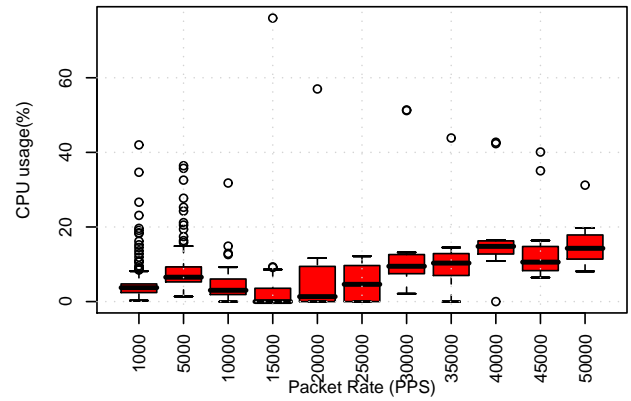


Fig. 9. CPU Usage for capturing thread

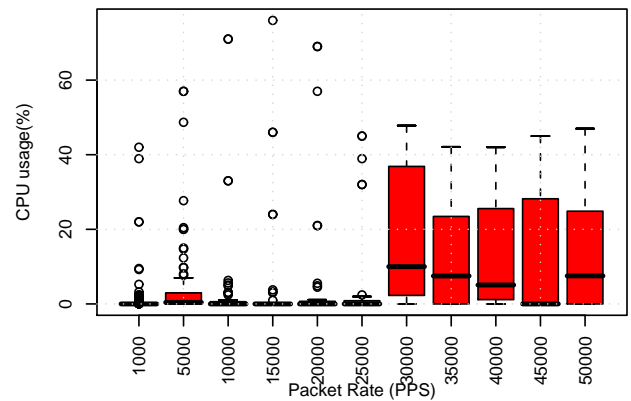


Fig. 10. CPU Usage for processing thread

cases. However, these were mostly single outlier value for these cases.

B. FM performance with High Flow Rate

In this test, we replayed the synthetic dump files created by MergeCap to simulate 1, 5, 10,... to 50 concurrent hosts. Each host sequentially generated 250 flows with a flow inter-arrival time of 0.5 millisecond. This results in 50 flows starting simultaneously at each host after 25msec. This made the merge files roughly equivalent to 250, 500, ... to 2500 flows started and run concurrently in 25msec, or proximately an average of 2000, 10000, 20000, ... to 100,000 flows per second.

1) *Packet Loss*: Figure 11 shows the packet loss rate due to PCAP buffer overflow in step 1 at the FM. It is plotted as a function of the number of concurrent hosts.

As can be seen in the figure, FM does not drop any packet for up to 10 concurrent hosts, with an equivalent flow rate of 20000 flows/sec. For higher flow rate (from 15 to 50 concurrent hosts), the PLR increases significantly - to more than 80% for 50 concurrent hosts, with an equivalent flow rate of 100,000 flows/sec.

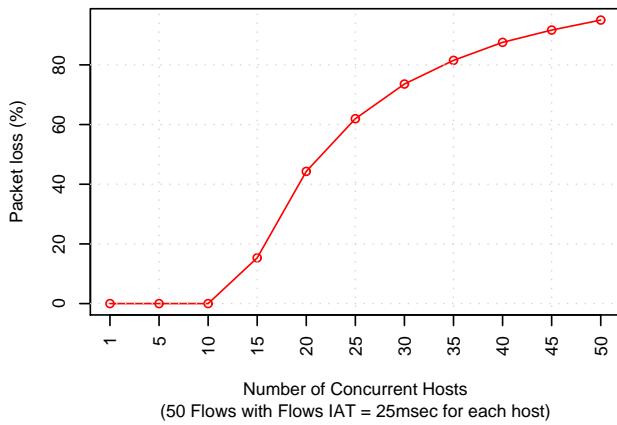


Fig. 11. Packet loss rate due to PCAP buffer overflow

Figure 12 shows the total packet to-be processed and actually been-processed in steps 2 and 3 as discussed in the previous section.

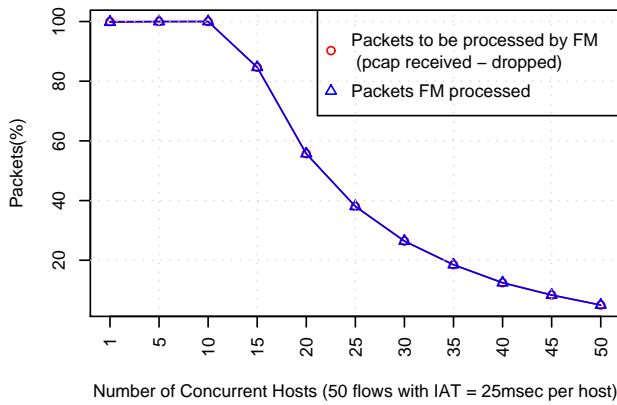


Fig. 12. Packets to-be/actually-being-processed in Step 2 and 3 processing

As can be seen in the figure, it seems that there is no packet loss in Steps 2 and 3 of the process. Looking more closely, we find that there is still a small packet loss, a majority less than 100 packets. This might due to the missing of the last one or two un-full SIM packets when terminating the test. The results are shown in 13.

Comparing results in Figure 11 and Figure 3, with the same packet rate, increasing number of flows in the traffic aggregate results in a slightly increase in PLR. Upto 20,000 flows/sec (10 concurrent hosts) PLR = 0%. With flow rate greater than 30,000 flows/sec, the FM suffers from packet loss. With the same packet rate, the increase in the number of flows in the traffic aggregate causes slightly increase in PLR. For example, with 30,000 flows/sec (15 concurrent hosts), the PLR is 15.33%; compared to the packet rate of 30,000 PPS

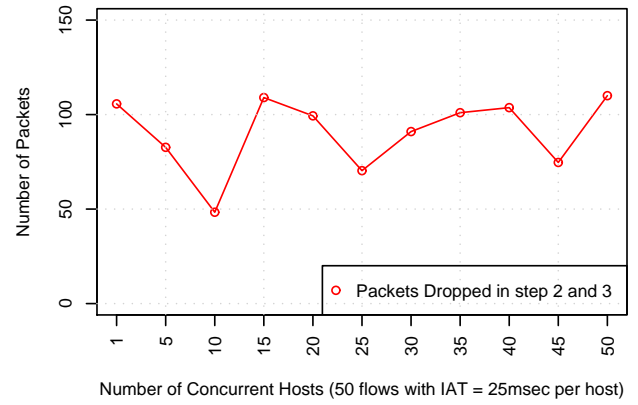


Fig. 13. Packets loss in Step 2 and 3 processing

from a single flow, the PLR is 12.89%. PLR for 40,000 flows/sec (20 concurrent hosts) is 44.30% while PLR for 40,000 PPS of a single flow is 38.11%. Similar results are seen for higher flow and packet rates. This shows that increased number of distinct flows in a traffic aggregate increases the PLR for the FM, even the packet rate is the same.

2) *CPU/Memory Usage*: Figure 14 and 15 show the memory usage of the FM capturing and processing threads for different packet rates.

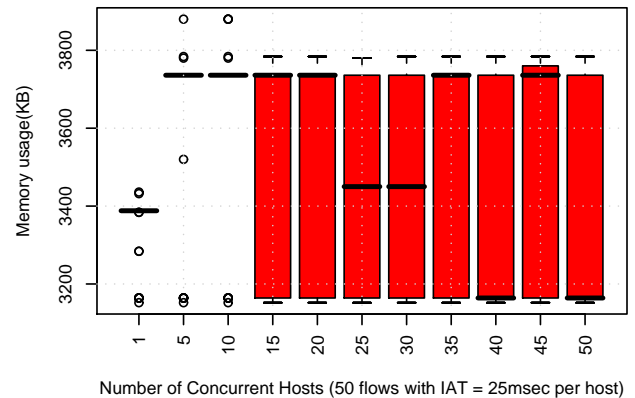


Fig. 14. Memory Usage for capturing thread

As can be seen in Figures 14 and 15 and comparing to Figure 3, there was no strong correlation between memory usages and packet loss rate. This is expected, as the FM does not response to packet loss. The memory usage increased when the number of concurrent hosts increased from 1 to 5 hosts (2000 to 10000 flows per second), and remained stable up to 20 concurrent hosts (roughly 40000 flows per second). However, for traffic rates higher than 25 concurrent hosts, it fluctuated at less than 3.8MB (noted that the packet loss rate for these cases were of greater than 50%).

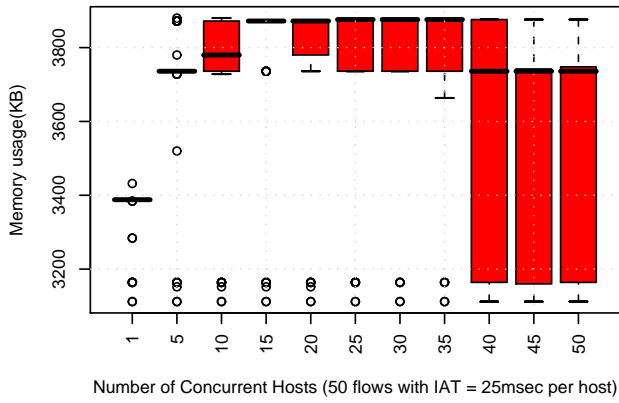


Fig. 15. Memory Usage for processing thread

Figure 16 and 17 show the CPU usage of the FM capturing and processing threads for different number of concurrent hosts.

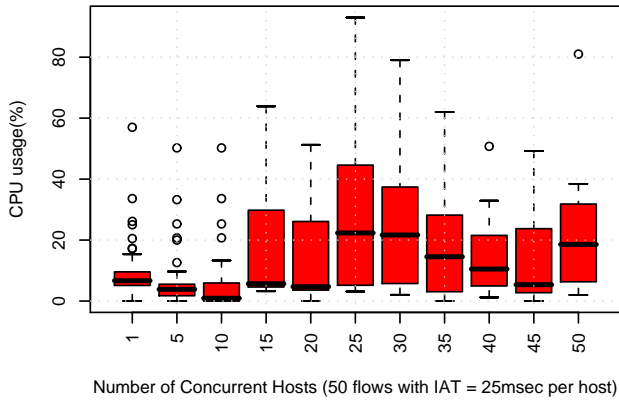


Fig. 16. CPU Usage for capturing thread

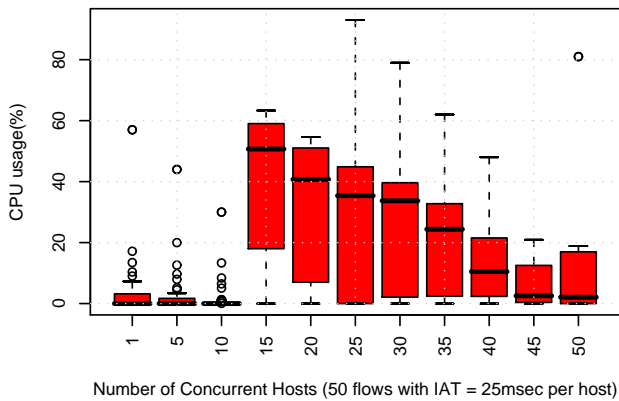


Fig. 17. CPU Usage for processing thread

As can be seen in the figure, the average CPU usage for the packet capturing thread varied when for different number of concurrent hosts, however, up to 50

concurrent ones, CPU usage remained less than 20%. Differently, the median CPU usage for the processing thread was very low (0%) for 1 to 10 concurrent hosts, increased significantly to approximately 50% for 15 concurrent hosts, and gradually reduced with higher number of concurrent hosts. This behaviour can be explained by the packet loss rate. There was no packet loss for 1 to 10 concurrent hosts, while there was more than 10% packet loss for 15 concurrent hosts. For more than 15 hosts running concurrently (equivalent flow rate of greater than 30000 flows per second), the packet loss rate increased dramatically. As most of the packets were dropped by PCAP buffer overflow, this reduced the number of packets that the FM needed to process.

C. Evaluation of The SuperFastHash algorithm

This section reports on the evaluation of the SuperFastHash algorithm employed in the FM. It is evaluated in terms of collision rate, hash generation speed and the distribution of hash values.

Packets coming from an ISP normally have a limited range of source IP addresses. This might increase the chance of flow hash collisions. In this test, we evaluate the Hash algorithm using the following procedure.

We wrote a small C++ program to automatically generate from 250,000 to 1 million unique flows, which have the source IP addresses correlated with 8, 16, and 24 bit netmasks (e.g. network with 16 bit netmask has src IP addresses in the range 100.100.0.1 to 100.100.255.254, network with 24 bit netmask has src IP addresses in the range 100.100.0.1 to 100.100.0.254). The destination IP address is totally random, while the source and destination ports are specifically configured, so that each flow is unique. During the test, IPv4 addresses were used, however, their values were stored in bigger buffers (16-byte source and destination IP addresses) as for the implementation scalability to support IPv6 addresses. With the current implementation, the FlowID is constructed so that the smaller IP address will be stored first. FlowID is 37-byte buffer, that consists of 16-byte IP address (smaller one), 2-byte corresponding port, 16-byte IP address (greater), 2-byte corresponding port and 1-byte protocol. The source port is incremented for each flow block, and wrapped around when it reaches the maximum value. The destination port is kept the same for each flow block and increased by one for the next block.

The collision rate is measured by the ratio of unique hash numbers generated out of 250K to a million input flows. The occurrence of each hash value is recorded for

analysis. Hash values are also recorded for post analysis of histogram distribution. A timer is inserted to calculate the total time taken to generate the hashes. (The hash generating time includes the time taken to compare and re-order IP addresses). An average speed is recorded for analysis.

Figure 18 shows the mean collision rates of the three repeated tests for each subnet (/8, /16, /24).

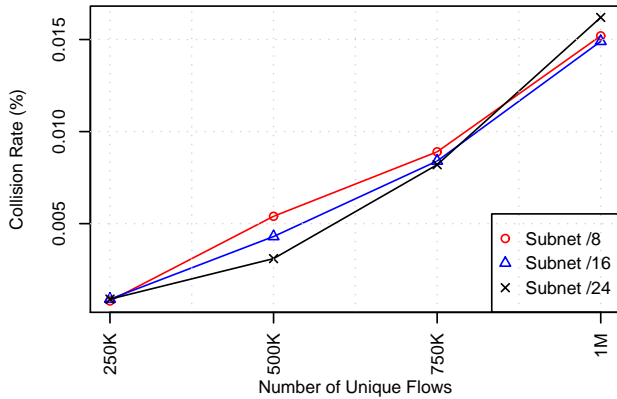


Fig. 18. Average hash collision rate for different netmask values

As can be seen in the figure, though increasing as the number of input flows increases, the collision rate reaches an average of $\sim 0.015\%$ (out of 1 million values) for different IP subnet.

The frequency occurrence of each hash value for each network IP address pool is shown in Figure 19. As can be seen in the figure, less than 0.015% of hash values have 2 collided occurrences for the whole population.

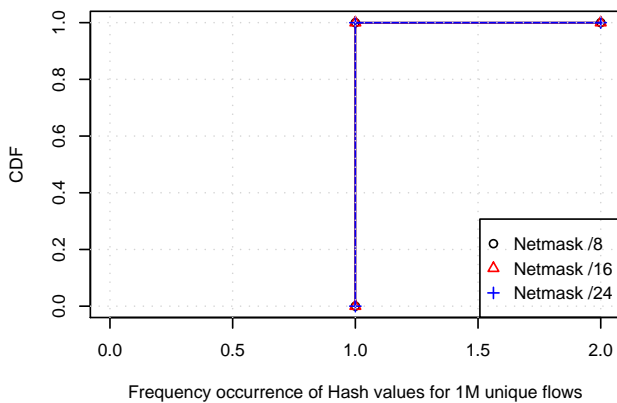


Fig. 19. Average hash collision rate for different netmask values

Figure 20 shows the histogram distribution of hash values for network IP netmask of 16 (100.x.x.x). The 1 million hash values have been put into 100 bins in the range of all possible values. Results show quite an

even distribution, despite a slightly higher frequency of the first bin's density. Similar results have been seen for other networks with IP addresses with netmasks of 8 and 24 bits.

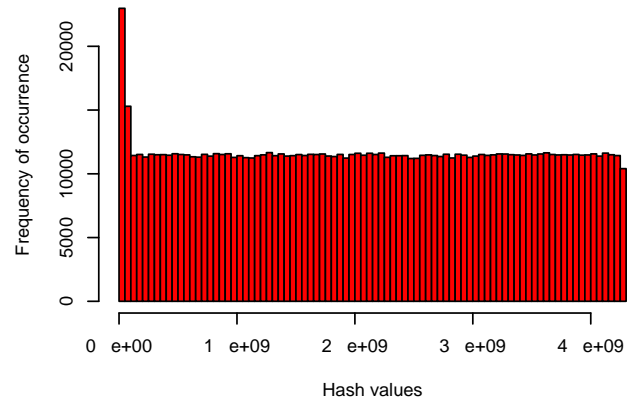


Fig. 20. Average hash collision rate for different netmask values

For all tests, the average time taken to generate each block of 250K hash values (including comparing and re-ordering IP addresses) was 0.097s, which is equivalent to ~ 2.5 million hashes per second.

The hash algorithm's performance in terms of collision, speed, and spreading values are considered to be good enough for deployment in the FM architecture.

IV. PERFORMANCE WITH POWER FAILOVER

A redundancy test for the Flow Meter in case of the main meter's failure is carried out separately. To achieve system redundancy, CARP (Common Address Redundancy Protocol [10]) - a tool which has been created and maintained by the OpenBSP project - is used. It is a free alternative to the VRRP (Virtual Router Redundancy Protocol) and the HSRP (Hot Standby Router Protocol). The tool enables multiple FM boxes to share a single, virtual network interface between them, so that if any machine fails, another can respond instead. The FM software is coded to keep polling the machine for master state. The polling interval is 1 second.

The test setup is illustrated in Figure 21 as below.

We have two identical hardware configuration FM boxes, one acting as the master and the other as backup FM. CARP is installed in both of the boxes. At configurable intervals (1 second by default - set by the *advbase* parameter), the master advertises its operation on IP protocol number 112. If the master goes offline (the backup CARP host doesn't see an advertisement from the master for 3 consecutive advertisement intervals), the backup system in the CARP group begins to advertise.

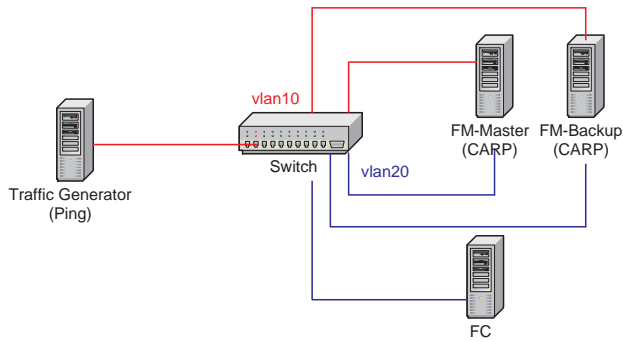


Fig. 21. Redundancy Test Setup

The host that is able to advertise most frequently becomes the new master. The CARP advertisement skew (*advskew*) parameter is used to skew the advertisement interval of a less preferred host in becoming master. In this test, the master FM box is set with default *advskew* value of 0, and the backup FM box is set with *advskew* = 100. The total advertisement interval is calculated as $\text{advbase} + (\text{advskew} / 255)$.

Detail CARP configurations for the FM master and backup boxes are described in Appendix A.

In this test, the traffic generator continuously pings the FC with ping interval of 5msec. To simulate the power failure of the master, we reboot the master FM during the test. The master's power failure would cause a number of ICMP packets to be lost while the backup FM detects the failover and assumes the packet capture task. Tcpdump is run at the Traffic Generator to capture ICMP packets. The number of Echo Reply packet losses and hence the maximum packet inter-arrival time of two consecutive Echo Reply messages is used to estimate the time taken for the system to recover.

We ran five repeated trials for the handover time estimation.

Figure 22 shows the sequence of ICMP Echo Reply packets during the 5 repeated trials. As can be seen in the figure, there is one small gap in the packet sequence for all the tests, which lies in the packet sequence window of [10700-11300].

Figure 23 shows the results in the sequence of ICMP packets during the 5 repeated trials with the sequence window of [10500-11500]. It shows that during the tests, there were an average of 512 Echo Reply packets lost during the power failover.

Figure 24 shows the Echo Reply packet inter-arrival times during the test vs. packet sequence number. As can be seen in the figure, the maximum packet IATs of approximately 2.5 to 3 seconds (average of 2.88 seconds)

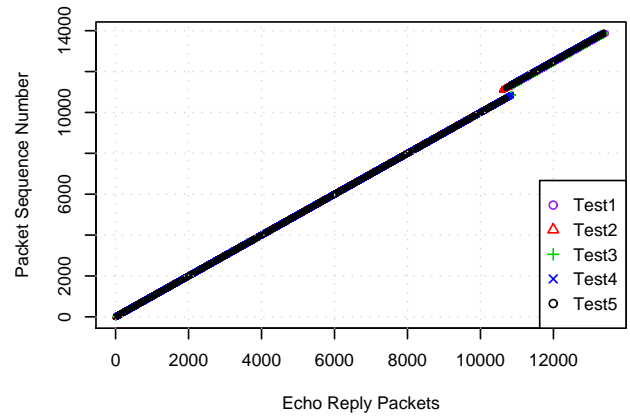


Fig. 22. Ping Packet loss during the power failover

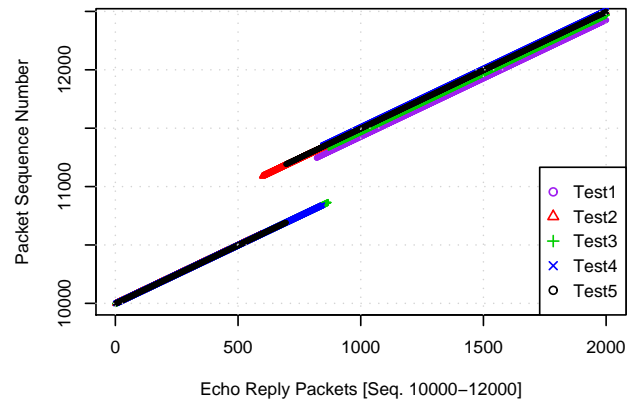


Fig. 23. Ping Packet loss during the power failover - zoomed-in

were caused by the interruption of the master FM power failure.

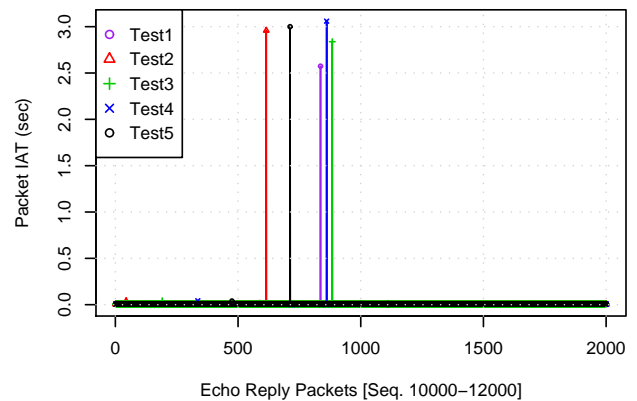


Fig. 24. Echo Reply Packet IAT (as captured at the Traffic Generator)

V. CONCLUSION

The performance of the FM in terms of packet loss, CPU and memory usages when coping with high data

rate and high number of concurrent flows has been tested and reported. There was significant packet loss (greater than 10%) when the incoming packet rate reaches 30,000 PPS or the incoming flow rate reaches 30,000 concurrent flows per second. With higher traffic/flow rate in real-world operational networks, a cluster of FM boxes might be required for better traffic capture in a high rate data network.

The evaluation of the hash algorithm employed by the FM was reported. A real-world deployment scenario was constructed and tested, with a small range of IP packets' source IP addresses. It has been found that with the test scenario, there was approximately 0.015% of hash collisions. This is considered to be acceptable for the operation of the ANGEL architecture.

The redundancy test results showed a handover time of ~ 3 seconds in case of master FM power failure for CARP architecture. Including the small amount of time (less than 2 seconds) for the FM application to poll the interface and start up, it would make the total of much less than 30 seconds for the system to recover from the failover. That is considered to be acceptable for the operation of the ANGEL architecture.

ACKNOWLEDGMENT

This work was supported from 2005 to early 2007 by the Smart Internet Technology Cooperative Research Centre, <http://www.smartinternet.com.au>.

I would also like to thank Jason But for his great help with debugging the FM and valuable discussions on testing approach, Lawrence Stewart for his FM's CARP coding and helpful discussion on testbed setup, Nigel Williams for his support with the FM's output traffic parser. I am also grateful for Grenville Armitage and Jason But's feedbacks on improving the report.

APPENDIX A

At both Master and Backup FM boxes, re-compile kernel with the following line:

```
device carp
CARP configuration:
```

At the Master

```
ifconfig carp0 create
ifconfig carp0 vhid 1 pass angel 1.1.10.1/24
ifconfig carp1 create
ifconfig carp1 vhid 2 pass angel 1.1.20.1/24
```

```
sysctl net.inet.carp.preempt=1
```

At the Backup

```
ifconfig carp0 create
ifconfig carp0 vhid 1 advskew 100 pass angel 1.1.10.1/24
ifconfig carp1 create
ifconfig carp1 vhid 2 advskew 100 pass angel 1.1.20.1/24
```

```
sysctl net.inet.carp.preempt=1
```

REFERENCES

- [1] P. Hsieh. (as of October 18th 2006) Hash functions. [Online]. Available: <http://www.azillionmonkeys.com/qed/hash.html>
- [2] T. Nguyen, "Evaluating Timestamping Accuracy for ASUS P5LD2-VM Motherboard with Intel NICs," CAIA, Tech. Rep. 070228F, February 2006, <http://caia.swin.edu.au/reports/050204A/caiaonly/CAIA-TR-070228F.pdf>.
- [3] J. But and J. Bussiere, "Improving NetSniff Capture Performance on FreeBSD by Increasing the PCAP Capture Buffer Size," CAIA Technical Report, Tech. Rep. 051027AA, October 2005.
- [4] *Industry Standard Network Performance Analysis System SmartBits 2000*, Spirent Communications, <http://www.spirentcom.com/documents/4077.pdf>, as of January 2007.
- [5] (as of September 15th 2006) Tcpreplay. [Online]. Available: <http://sourceforge.net/projects/tcpreplay/>
- [6] (as of September 18th 2006) Mergecap. [Online]. Available: <http://www.ethereal.com/docs/man-pages/mergecap.1.html>
- [7] J. But, "ANGEL Flow Meter - Software Architecture Design Document," CAIA, Tech. Rep. 070228C, January 2007, <http://caia.swin.edu.au/reports/050204A/caiaonly/CAIA-TR-070228C.pdf>.
- [8] (as of October 10th 2006) The R project for statistical computing. [Online]. Available: <http://www.r-project.org/>
- [9] (as of October 10th 2006) Getting started with R. [Online]. Available: <http://www.people.carleton.edu/~lchihara/Splus/R-Start.pdf>
- [10] OpenBSD. (as of October 25th 2006) The Common Address Redundancy Protocol (CARP). [Online]. Available: <http://www.openbsd.org/faq/faq6.html#CARP>