SWINBURNE

CENTRE FOR ADVANCED INTERNET ARCHITECTURES

Experimental Experiences with Emulating Multihop Path using Dummynet

Thuy T.T. Nguyen

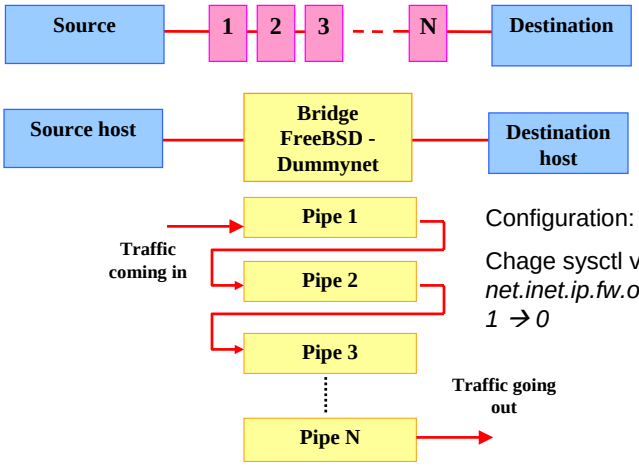
Introduction

- Motivations
- How to emulate multihop path using Dummynet
- Test setup & Results
- Analysis
- Conclusions

Motivations

- Investigate the network conditions (e.g. congestions, bottleneck bandwidth, multihop path, cross traffic) on flow traffic statistical characteristics
- The need of simulating multihop path with different hop configurations, testing with live traffic
- Emulating multihop path using Dummynet – FreeBSD – kernel-resident tool for traffic shaping with bandwidth, delay, loss … configurations ?

How to emulate multihop with dummynet



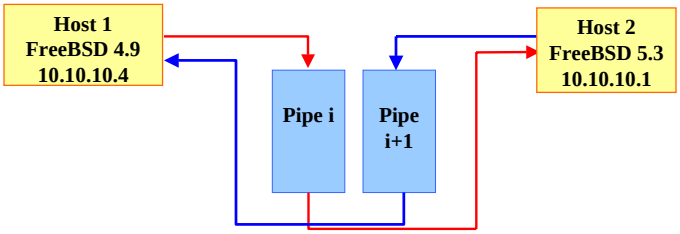
Configuration:

Chage sysctl variable *net.inet.ip.fw.one_pass: 1 → 0*

Test Setup and Results

- Emulate 1, 2, …, 10 hops between host1 and host2
- All pipes have same configurations (bw and delay)
- Ping traffic with different Ping packet sizes
- Clock precision at all machines HZ = 1000
- Path between host1 and host2 is full-duplex (different pipe for incoming and outgoing traffic)

Test Setup and Results



Hop *i* configurations:

ipfw pipe {i} config bw {1, 2}Mbps delay {0 10}ms

ipfw pipe {i + 1} config bw {1, 2}Mbps delay {0 10}ms

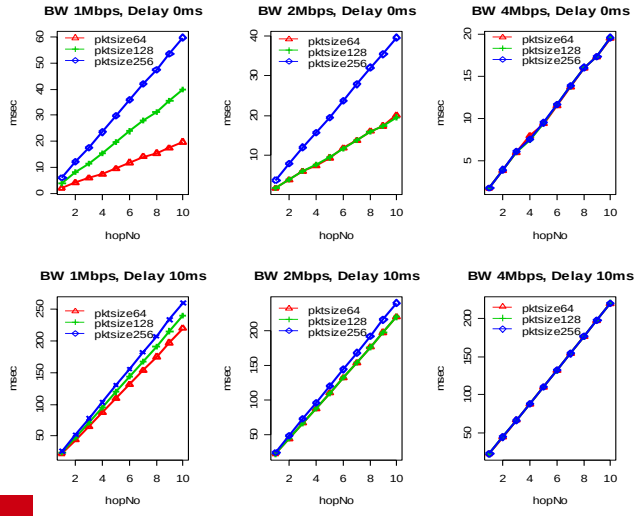
ipfw add {i} pipe {i} icmp from 10.10.10.4 to any

ipfw add {i + 1} pipe { i + 1} icmp from any to 10.10.10.4

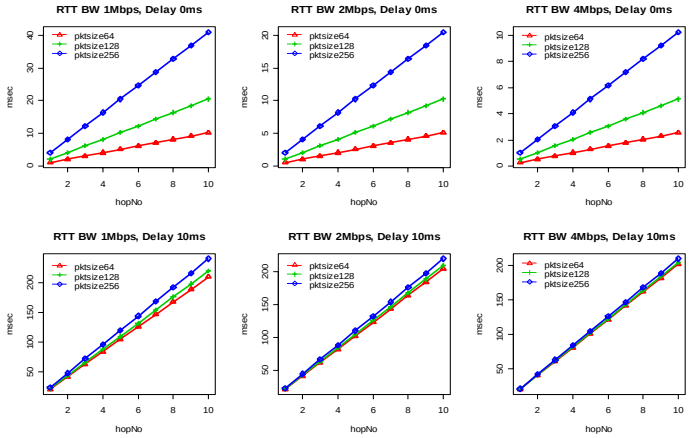
Test Setup and Results

- Ping from Host 1 to Host 2
- Ping Interval 1sec
- Ping Packet Size 64, 128, 256 bytes
- Configured at bridge: BW 1, 2, 4 Mbps, Delay 0, 10ms

Test Setup and Results



Test Setup and Results

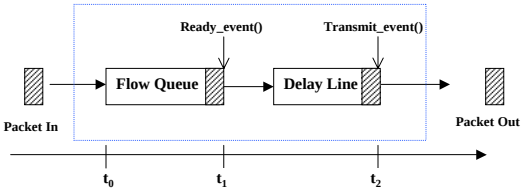


Tp Ping(64, 128, 256, BW 1Mbps) = 0.512, 1.02, 2.05 (msec)

Tp Ping(64, 128, 256, BW 2Mbps) = 0.256, 0.512, 1.02 (msec)

Analysis

❑Dummynet internal timing and scheduling implementation for one hop:



t_0: Time packet arrives and is placed into queue

t_1 : Time packet is pulled out of the queue by the scheduler to put into delay line

= Time packet entering the delay line

t_2 : Time packet is transmitted out of pipe

Analysis

Pipe configurations:

bw = bandwidth (Bits/sec)

p_delay = delay (msec)

p_len = packet length in bytes

Let t_{last_sched} = time the first packet arrives, or the previous scheduler event fired (simulator virtual time, unit clock ticks)

t_0 = the packet current (arrival) time (simulator virtual time, unit clock ticks)

Analysis

Number of bytes allowed to be extracted at t_1 is calculated based on the cumulative bandwidth from the last scheduled event. Dummynet calculation [3]:

$$q\_numBytes = (t_0 - t_{last\_sched}) \times bw$$

Then, the number of clock ticks that the packet needs to wait before being put into the delay line is:

$$q\_numTicks = \frac{p\_len \times 8 \times HZ - q\_numBytes - 1 + bw}{bw} \quad (1)$$

Analysis

The number of clock ticks that the packet needs to wait in the delay line is:

$$dl\_numTicks = \frac{p\_delay \times HZ}{1000}$$

So the total clock ticks from packet coming-in to coming-out is:

$$pipe\_numTicks = \frac{p\_len \times 8 \times HZ - q\_numBytes - 1 + bw}{bw} + \frac{p\_delay \times HZ}{1000}$$

Analysis

- Ambiguity about question (1)
- Possible explanation:

$$bw = bits/\sec \rightarrow BW = \frac{bw}{8 \times HZ}(bytes/tick)$$

No of bytes allowed to transmit at t0:

$$(t_0 - t_{last\_sched}) \times BW$$

The rest of the packet need to be sent after:

$$\frac{pkt\_len - (t_0 - t_{last\_sched}) \times BW}{BW}(ticks)$$

$$= \frac{pkt\_len - (t_0 - t_{last\_sched}) \times \frac{bw}{8 \times HZ}}{\frac{bw}{8 \times HZ}}$$

$$= \frac{pkt\_len \times 8 \times HZ}{bw} - (t_0 - t_{last\_sched})$$

$$= \frac{pkt\_len \times 8 \times HZ - (t_0 - t_{last\_sched}) \times bw}{bw}$$

$$= \frac{pkt\_len \times 8 \times HZ - q\_numBytes}{bw}(ticks)$$

Analysis

So total number of ticks that packet needs to wait in the flow queue is :

$$\frac{pkt\_len \times 8 \times HZ - q\_numBytes}{bw} + 1(ticks)$$

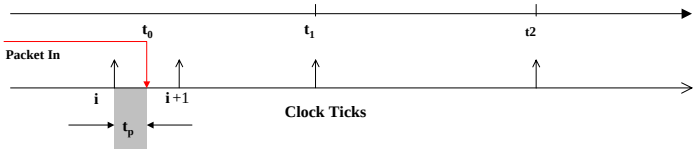
To avoid waiting for an extra tick in the case when bytes to send ***o*** *bytes allowed to be sent, adding a rounding factor. Therefore:*

$$q\_numTicks = \frac{p\_len \times 8 \times HZ - q\_numBytes + bw - 1}{bw}$$

Total number of ticks that packet needs to wait in the pipe:

$$pipe\_numTicks = \frac{p\_len \times 8 \times HZ - q\_numBytes + bw - 1}{bw} + \frac{p\_delay \times HZ}{1000}$$

Analysis



One-way delay for 1 hop (OWD):

$$OWD = floor(t_p + \frac{pipe\_numTicks}{HZ}) - t_p(\text{sec})$$

One-way delay for N hop (OWD_N):

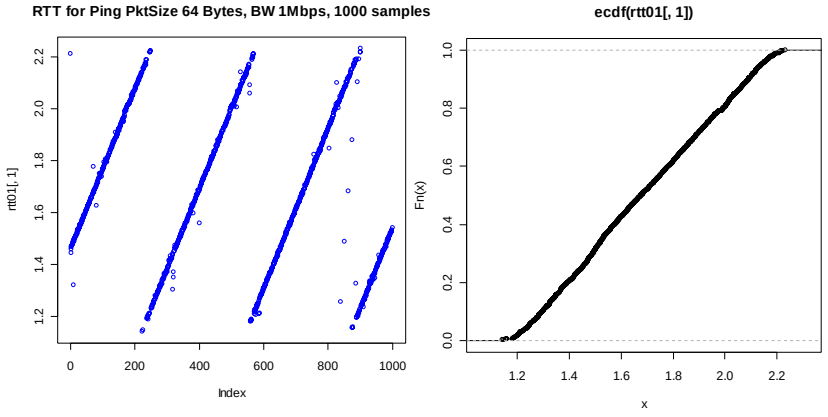
$$OWD_N = (floor(t_p + \frac{pipe\_numTicks}{HZ}) - t_p(\text{sec})) + (N-1) \times \frac{pipe\_numTicks}{HZ}(\text{sec})$$

Analysis

Question:

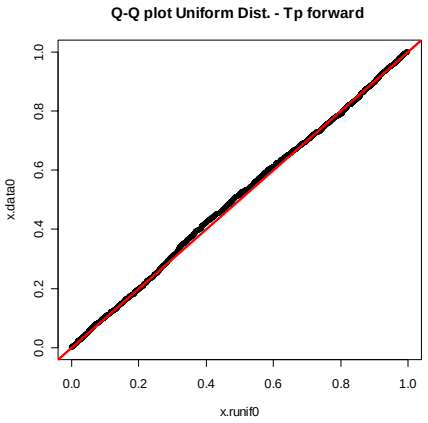
What factors needed to take into account while experimenting with dummynet?

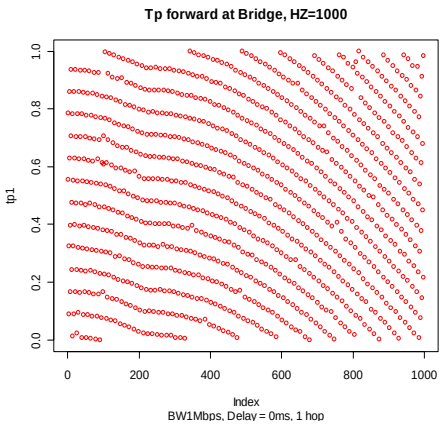
Analysis - More empirical results



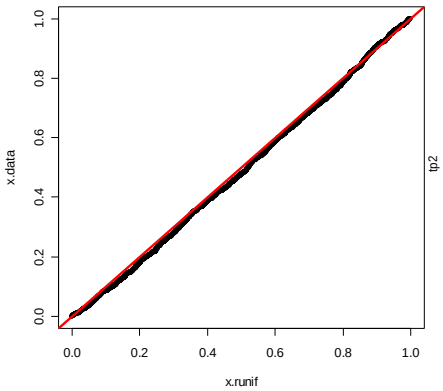
Analysis – More empirical results

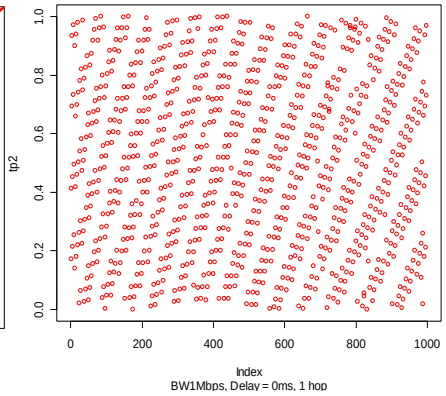
t_p factor : Taking the microseconds part from pkt timestamp





Analysis – More empirical results





Analysis – More empirical results

a = transmission time of pkt at 100Mbps(❍s)

✆t = packet interarrival time (ping interval)

Ks and Kb = clock skewing factor at source and bridge

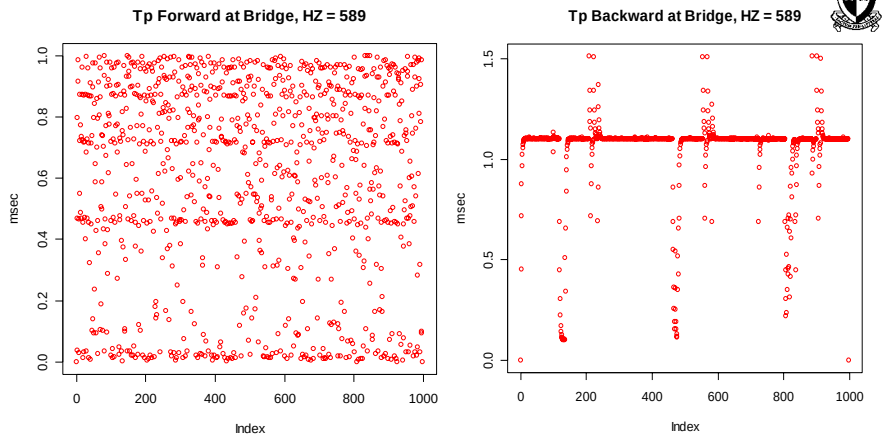
Since the test environment is near-perfect synchronous condition, HZ at Bridge, S, and D are equal = 1000, and ✆t = 1000000❍sec

Source	Bridge	
t01	t11 = t01 + a	→ tp1
t02	t12 = t01 + ✆t + a + Ks + Kb	
	→ tp2 = tp1+ Kb + Ks	
t03	t13 = t01+2✆t + a + 2(Ks+ Kb)	
	→ tp3 = tp1+ 2(Ks + Kb)	

And so on….tpN = tp1 + (N-1)(Ks+Kb)

→ that's why the curve in previous Tp graphs

Analysis

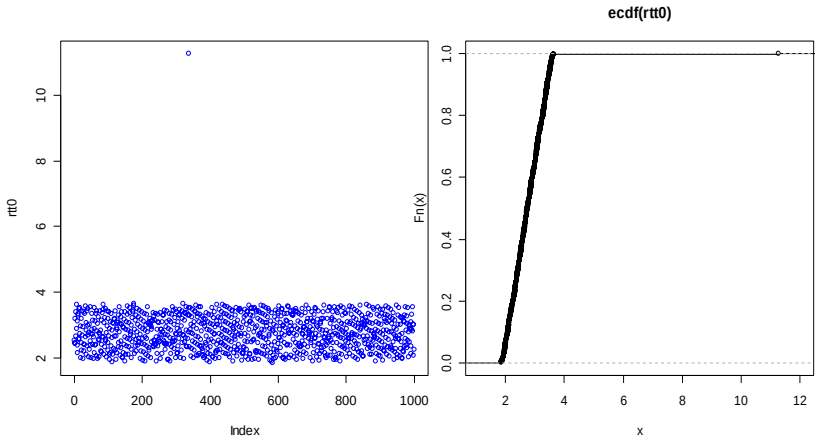


HZ at Source = HZ at Destination = 1000

tp factor : assume tp[1] = 0, i ¤ 2: $t_p[i] = \mathrm{mod}((t[i]-t[i-1])+t_p[i-1]*1000)/(1000000/589)(m\,\mathrm{sec})$

*note: tp shown in the graph is calculated as relative to tp[1]

Analysis



Analysis

Question:

What factors needed to take into account while experimenting with dummynet?

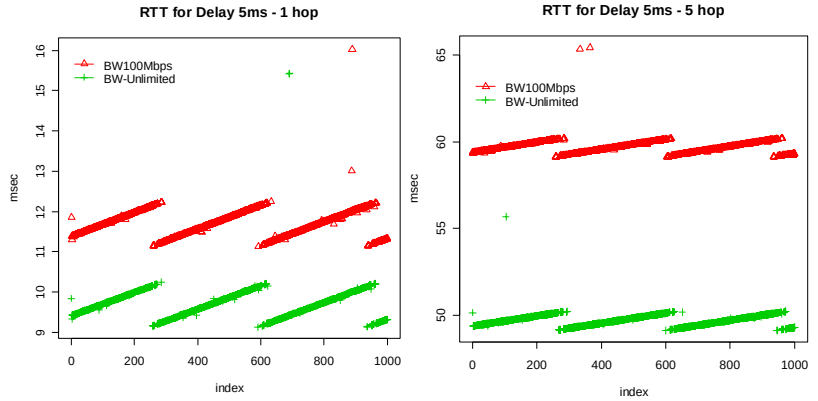
Answer:

Dummynet's results might be affected by the real system, in which the components are usually asynchronous, e.g:

- Clock granularity differences between machines
- Clock skewness within a single machine and between different machines
- Kernel Load
- Interfering traffic …

Some implications

- Configure Delay with BW simultaneously



$$pipe\_numTicks = \frac{p\_len \times 8 \times HZ - numBits - 1 + BW}{BW} + \frac{p\_delay \times HZ}{1000}$$

Some Implications

- Configure Delay only (without BW)
  - The higher the delay, the more precise the dummynet [1]

$$OWD = floor(\frac{p\_delay \times HZ}{1000} + t_p) - t_p$$

$$Error = \frac{1 - t_p}{p\_delay} \times 100\% \qquad \textit{when HZ =1000}$$

- Real Delay is not influenced by the packet rate [1]
  - Only true if the clock granularity of the sending source is ¤ the processing time of the packet & the clock granularity at the bridge

→Otherwise, would be affected by packet rate and packet sizes caused by queuing delay at the delay line

Conclusions

- Dummynet can be used for multihop path emulation, however, need to be aware of the effects of testbed factors, e.g. clock granularity, skewness of machine's clock, synchronisation of simulation time and system time, load at server, source and bridge, precision of interface's timestamp …
- More factors to investigate: cross traffic, pkt loss

References

[1] W.A. Vanhonacker, " Evaluation of the FreeBSD dummynet network performance simulation tool on a Pentium 4-based Ethernet Bridge", CAIA Technical Report 031202A, December 2003

[2] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols", ACM SIGCOMM Computer communications Review, Volume 27, Issue 1, January 1997

[3] Dummynet source code FreeBSD 4.9

Acknowledgement

Great thanks to Grenville and Lawrence!

THANK YOU!