KauNetEm

Deterministic Network Emulation in Linux

Per Hurtig, Johan Garcia

February 12, 2016

netdevconf 1.1, February 10-12, Seville, Spain

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Talk outline

1. Deterministic Network Emulation
2. Use Case Examples
3. KauNetEm System Design
4. Demo
5. Open Issues
6. The Way Forward / Closing Remarks

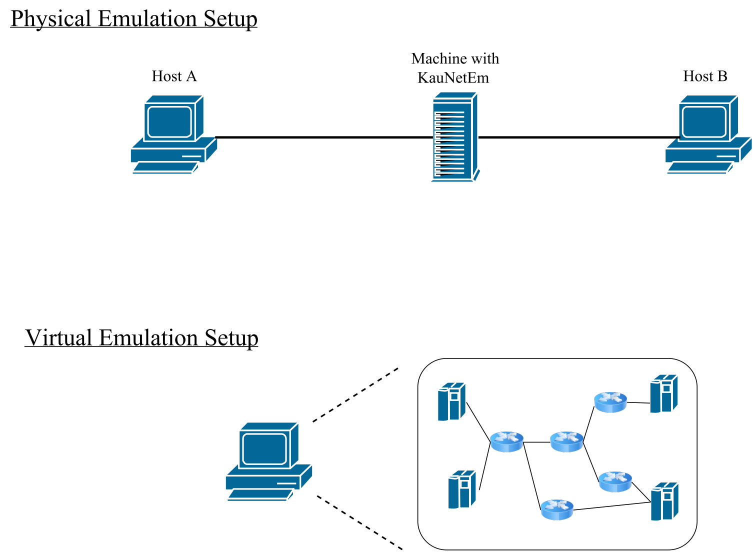
Deterministic Network Emulation

What?

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

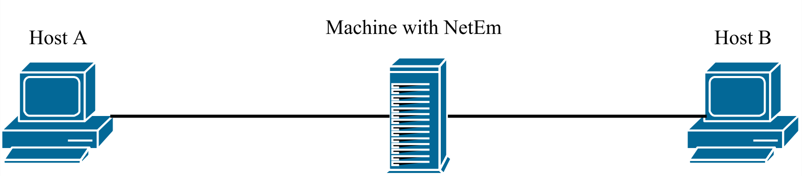
Deterministic network emulation allows the experimenter not only to generate various emulation effects such as packet loss, rate restrictions or delay, but to apply these emulation effects at **`precisely controlled places.`**

Emulation Setups



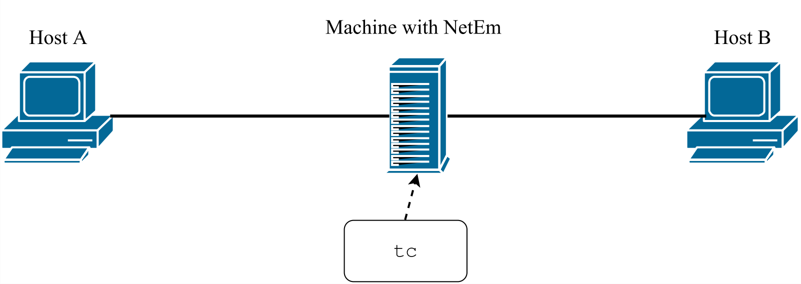
NetEm Workflow

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)



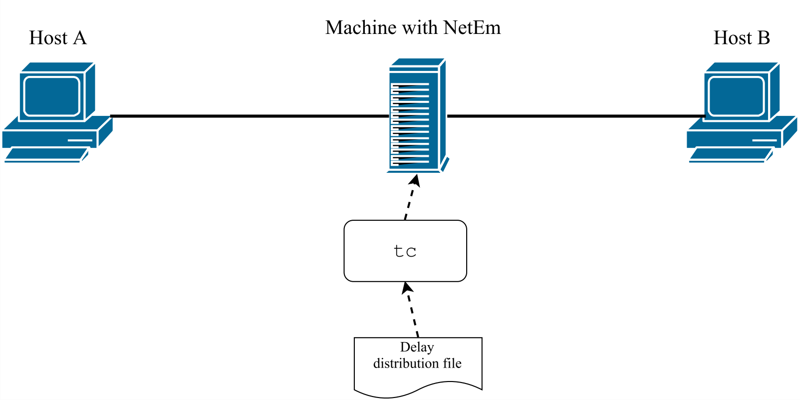
Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

NetEm Workflow



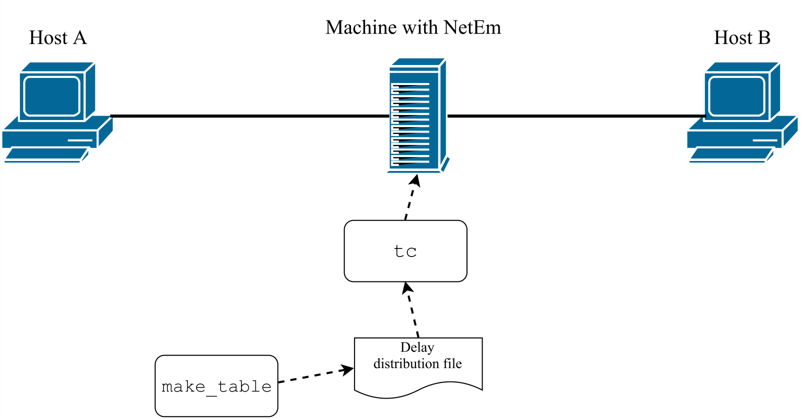
Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

NetEm Workflow

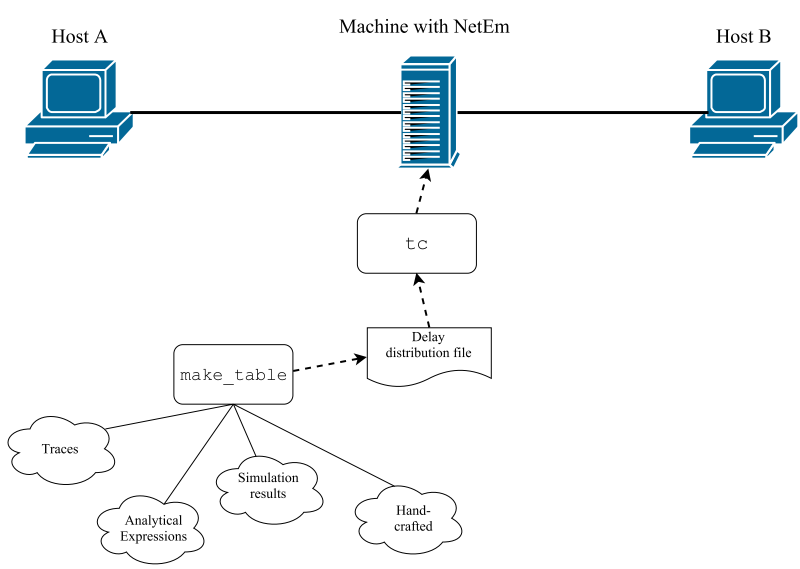


Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

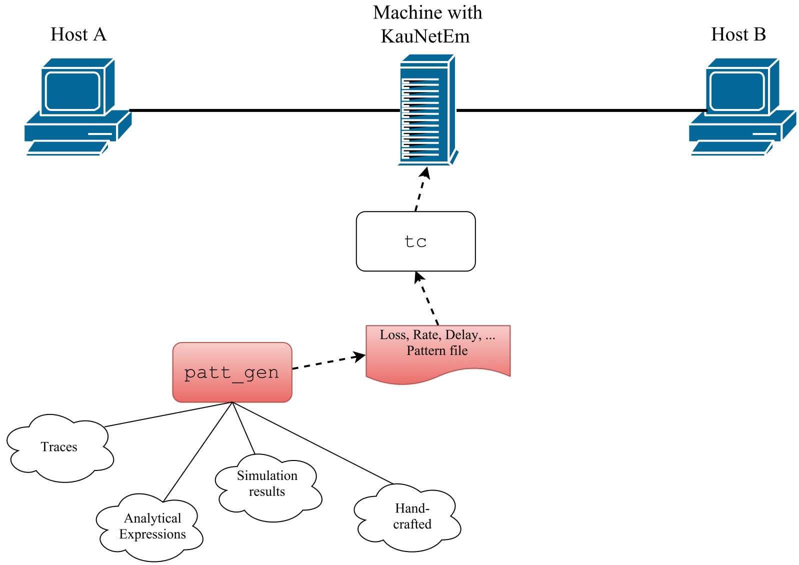
NetEm Workflow



NetEm Workflow

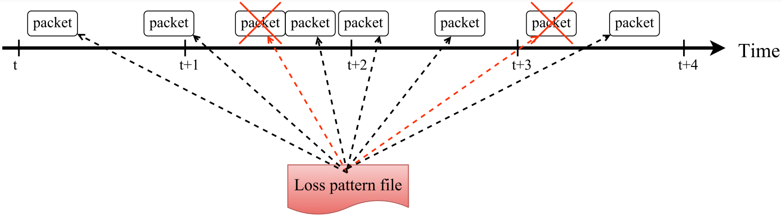


KauNetEm Workflow

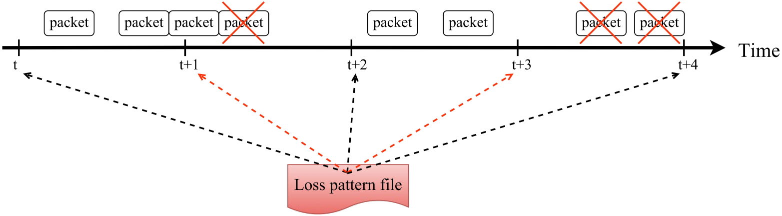


Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Data vs. Time driven



Data driven mode



Time driven mode

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Emulation Effects

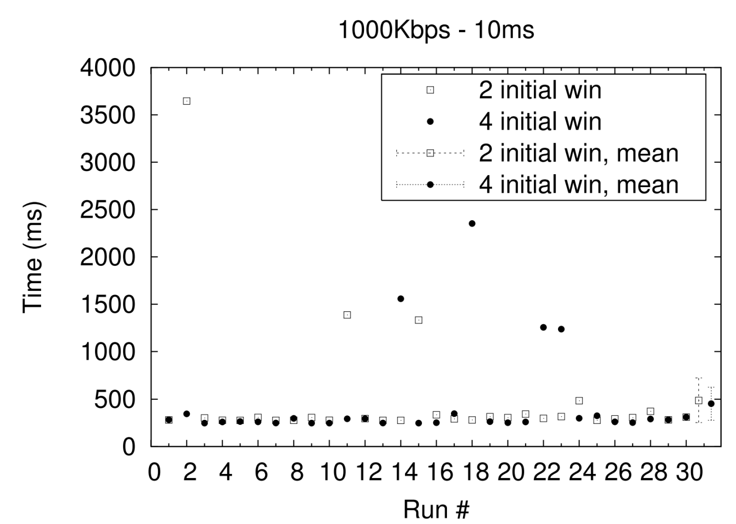
Emulation effect	Data-driven	Time-driven
Packet loss	X	X
Delay	X	X
Rate	X	X
Bit error	X	-
Duplication	X	X
Reordering	X	-
Trigger	-X-	-?-

Use Case Examples

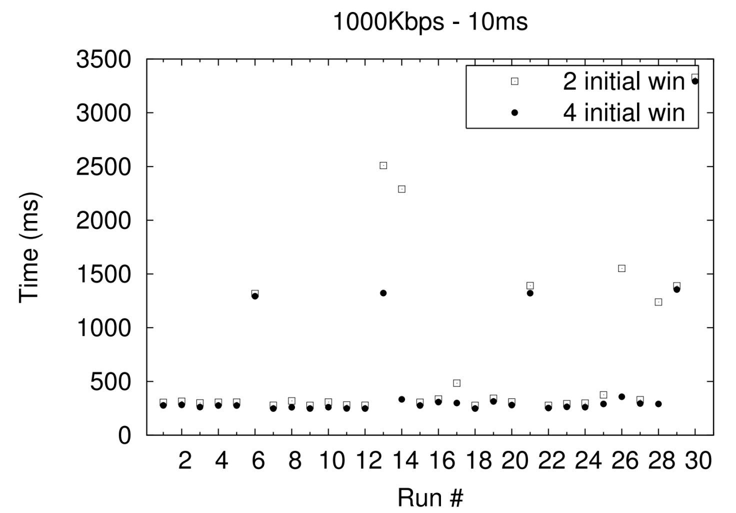
Why?

- Transport protocol implementation debugging / validation
- Functional evaluation of new Transport layer mechanisms
- Transport or Application layer performance evaluations
- "Control what you can, and randomize the rest"

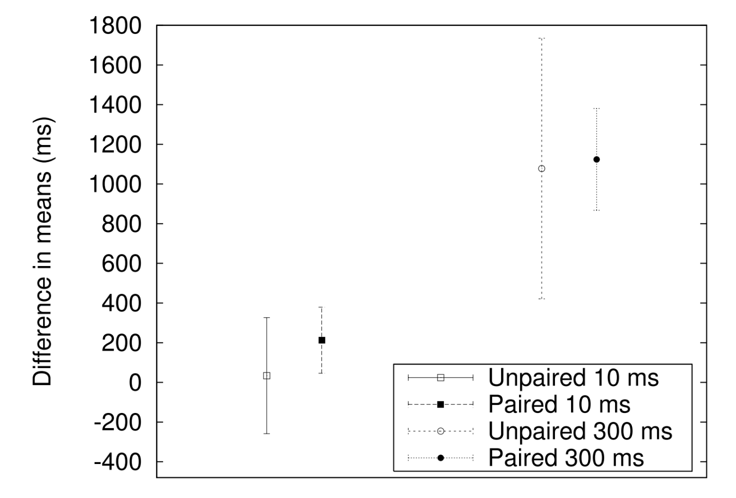
Random losses



Randomly generated loss pattern

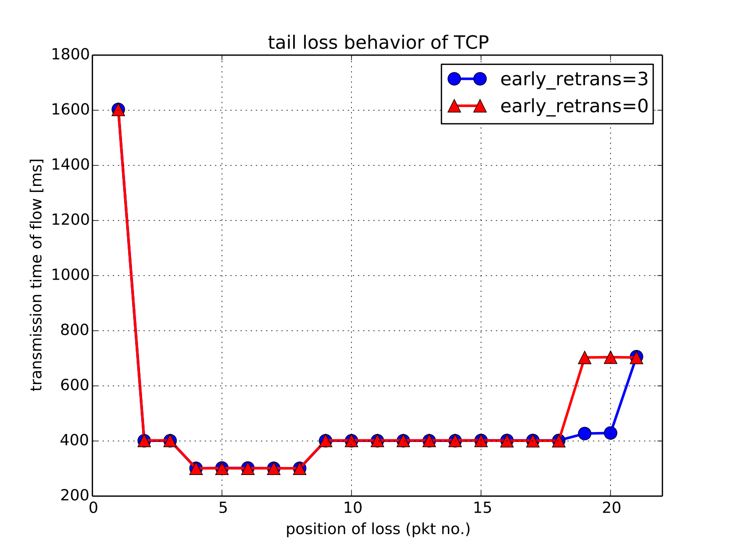


Increased statistical strength



Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

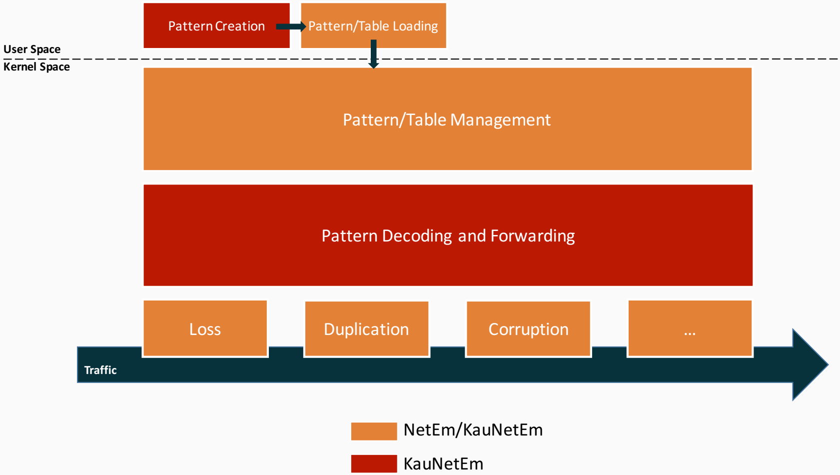
Functional evaluation of Transport layer mechanisms



KauNetEm System Design

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

How?



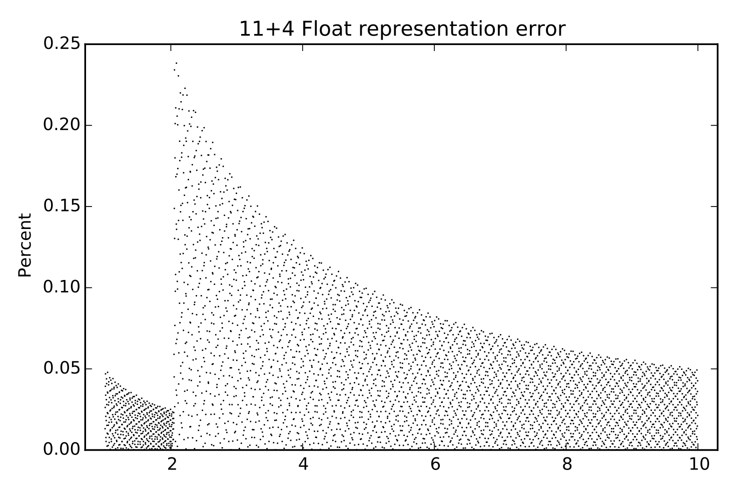
Data representation - Float data

15	14	13	12	11	10	9	8	7	6	5	4	3	3	2	1	0
F	11 bit mantissa												4 bit exponent			
F	15 bit run length value															

F: Flag to indicate if run-length value or float value is encoded

- Used for Bandwidth and Delay patterns
- 0 - $2.047 \cdot 10^{0-15}$ [bps, μs] (i.e max at $\sim$ 2Pbps, $2 \cdot 10^9$s)

Data representation - Float data



Data representation - Packet data

15	14	13	12	11	10	9	8	7	6	5	4	3	3	2	1	0
F	15 bit run length value															

F: Indicates if packet at current position should be dropped / duplicated

- Used for Paket loss and Packet duplication patterns
- 0.1% packet loss rate at 100Mbps ⇒ 24 minutes / 16KByte pattern

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Data representation - Integer data

15	14	13	12	11	10	9	8	7	6	5	4	3	3	2	1	0
F	15 bit integer value															
F	15 bit run length value															

F: Flag to indicate if run-length value or float value is encoded

- Used for Reordering, Bit errors, and future Trigger patterns
- 14 bits to encode bit to flip / reordering distance / trigger value

User Space Code Extensions

- `tc/q_netem.c:` 75 lines of code added
  - Command line parsing
  - Pattern loading/transfer
- `patt_gen:` $\sim$ 1400 lines of c code
  - Value encoding/decoding routines
  - Pattern creation handling
  - I/O

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Pattern generation

```
patt_gen -pkt|-ber|-del|-bw|-reo|-dup|-trig
         -s <size> [-o <outfilename>]
         <position-values>|-f infilename
```

<position-values>: A comma separated list of positions and values, or loss/duplication positions.

<size>: Specifies the length of the generated pattern. The unit is packets or milliseconds.

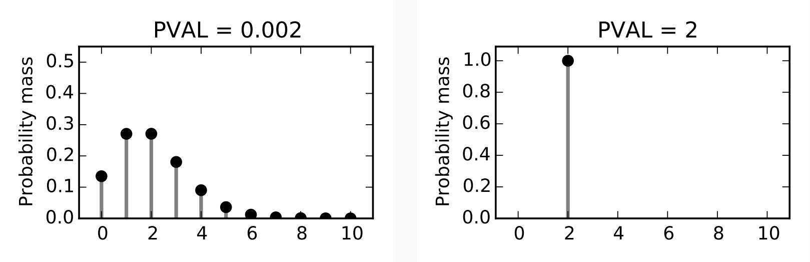
For packet loss patterns:

```
patt_gen -pkt -rand -s size [-r random_seed] <PVAL>

patt_gen -pkt -ge -s size [-r random_seed]
       <good_rate> <bad_rate> <good_tran_prob> <bad_tran_prob>

patt_gen -pkt -int -s size <interval-list>|-f <infilename>
```

Packet loss probability vs Actual loss rate

There is a difference between a 0.2% packet loss probability and an actual 0.2% achieved loss rate.

Example: 0.2% loss for a 1000 packet flow:



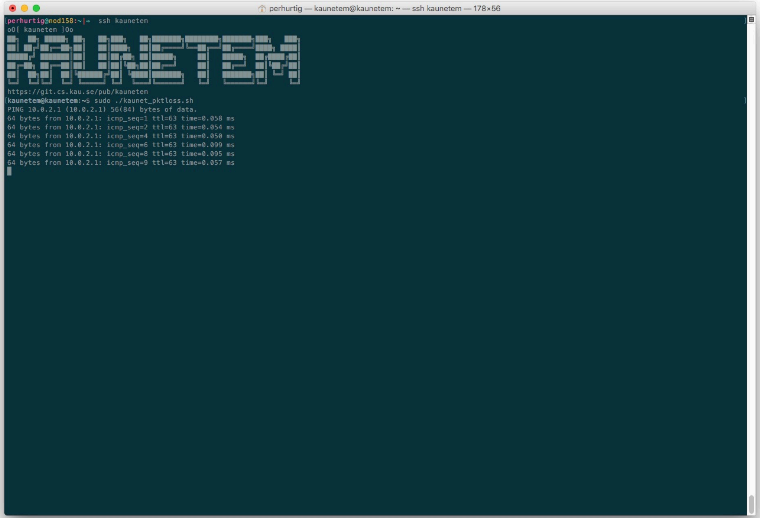
Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Kernel Space Code Extensions

- `pkt_sched.h:` 12 lines (2 variables, 10 defines)
- `sch_netem.c:` ~400 lines
  - Management and traversal of patterns
    - pattern forwarding
    - value decoding,
    - hrtimer management
  - Pattern effect invocation

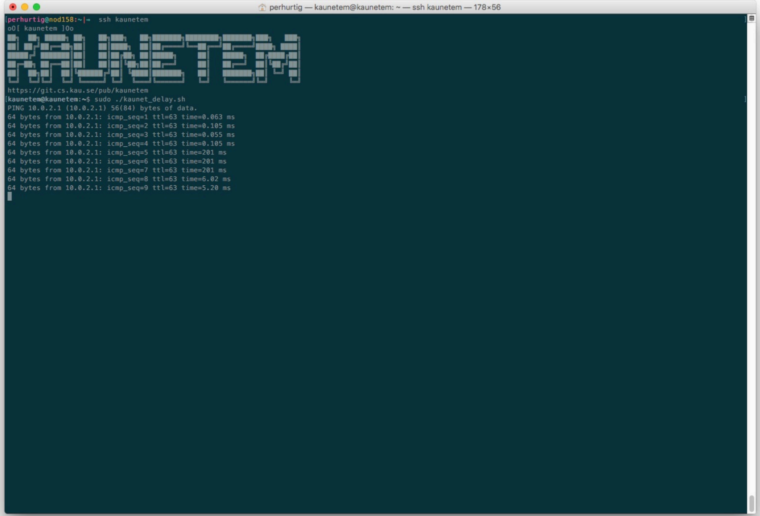
Demo

Packet Loss


perhurtig — kaunetem@kaunetem: ~ — ssh kaunetem — 178×56
perhurtig@nod158:~|→ ssh kaunetem
oO[kaunetem]Oo
KAUNETEM
https://git.cs.kau.se/pub/kaunetem
kaunetem@kaunetem:~$ sudo ./kaunet_pktloss.sh
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=63 time=0.058 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=63 time=0.054 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=63 time=0.050 ms
64 bytes from 10.0.2.1: icmp_seq=6 ttl=63 time=0.099 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=63 time=0.095 ms
64 bytes from 10.0.2.1: icmp_seq=9 ttl=63 time=0.057 ms

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Duplication



Open Issues

Open design considerations

- What should happen at the end of a pattern?
  - Just end pattern?
  - Wraparound?
  - Append new pattern?
- Behavior for packet being sent at low throughput when time-driven increase of throughput happens?
  - Send remaining bits at original low rate?
  - Send remaining bits at new higher rate?
- Need for additional patterns?
  - Trigger patterns will be added
  - Any more?
- Fold patt_gen code into tc?
- Split delay patterns to delay_reorder and delay_fifo?

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Why extend Netem?

Is using NetEm the best way to achive deteministic emulation?

+ Not a lot of added kernel code

+ Much of emulation effect infrastructure is present

- NetEm qdisc cannot be nested

- Not possible to easily have multiple concurrent patterns

Is an approach based on filter actions a possible alternative or possible complement?

Should we create a separate qdisc instead?

The Way Forward / Closing Remarks

The way forward

We will continue to:

- add code functionality
- provide documentation / examples.

Possible integrations:

- CORE: Common Open Research Emulator
- LNST: Linux Network Stack Testing tool

Room for contributions:

- Bug reports
- Code patches
- Feature requests

Proceedings of NetDev 1.1: The Technical Conference on Linux Networking (February 10th-12th 2016. Seville, Spain)

Closing remarks

We believe deterministic emulation has an important role to fill for networking researchers and protocol implementers.

KauNetEm is an ongoing work aiming to provide deterministic emulation in Linux in an easy to use and well-documented way.

Questions?

git.cs.kau.se/pub/kaunetem

per.hurtig@kau.se

johan.garcia@kau.se