KauNetEm

Deterministic Network Emulation in Linux

Per Hurtig, Johan Garcia
February 12, 2016

netdevconf 1.1, February 10-12, 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
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

Physical Emulation Setup

Virtual Emulation Setup
NetEm Workflow

Host A  Machine with NetEm  Host B
NetEm Workflow

Host A  Machine with NetEm  Host B

`tc`
NetEm Workflow

Host A

Machine with NetEm

Host B

tc

Delay distribution file
NetEm Workflow

Host A  Machine with NetEm  Host B

```
tc

delay distribution file

make_table
```
Data vs. Time driven

Data driven mode

Time driven mode
## Emulation Effects

<table>
<thead>
<tr>
<th>Emulation effect</th>
<th>Data-driven</th>
<th>Time-driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet loss</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Delay</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Rate</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bit error</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Duplication</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Reordering</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Trigger</td>
<td>-X-</td>
<td>-?-</td>
</tr>
</tbody>
</table>
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

![Graph showing time vs run number for different initial win scenarios at 1000Kbps and 10ms latency.]

- 2 initial win
- 4 initial win
- 2 initial win, mean
- 4 initial win, mean
Randomly generated loss pattern

![Graph showing the effect of randomly generated loss pattern on packet delivery time. The graph compares two scenarios: 2 initial window size and 4 initial window size. The x-axis represents the run number, ranging from 2 to 30, while the y-axis represents time in milliseconds, ranging from 0 to 3500. The graph indicates that a larger initial window size generally leads to a quicker recovery from packet loss.](image-url)
Increased statistical strength
Functional evaluation of Transport layer mechanisms
KauNetEm System Design
How?

Pattern Creation → Pattern/Table Loading → Pattern/Table Management → Pattern Decoding and Forwarding → Loss → Duplication → Corruption → ...

NetEm/KauNetEm

KauNetEm
Data representation - Float data

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>11 bit mantissa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15 bit run length value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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} \text{ [bps, \(\mu s\)] (i.e max at \sim 2Pbps, 2 \cdot 10^9 s)}
Data representation - Float data

11+4 Float representation error

- Percent
- 11+4 Float representation error

### Data representation - Packet data

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
Data representation - Integer data

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15 bit integer value

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**: ~1400 lines of c code
  - Value encoding/decoding routines
  - Pattern creation handling
  - I/O
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:
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
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?
Why extend Netem?

Is using NetEm the best way to achieve deterministic 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
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