KauNet Triggers

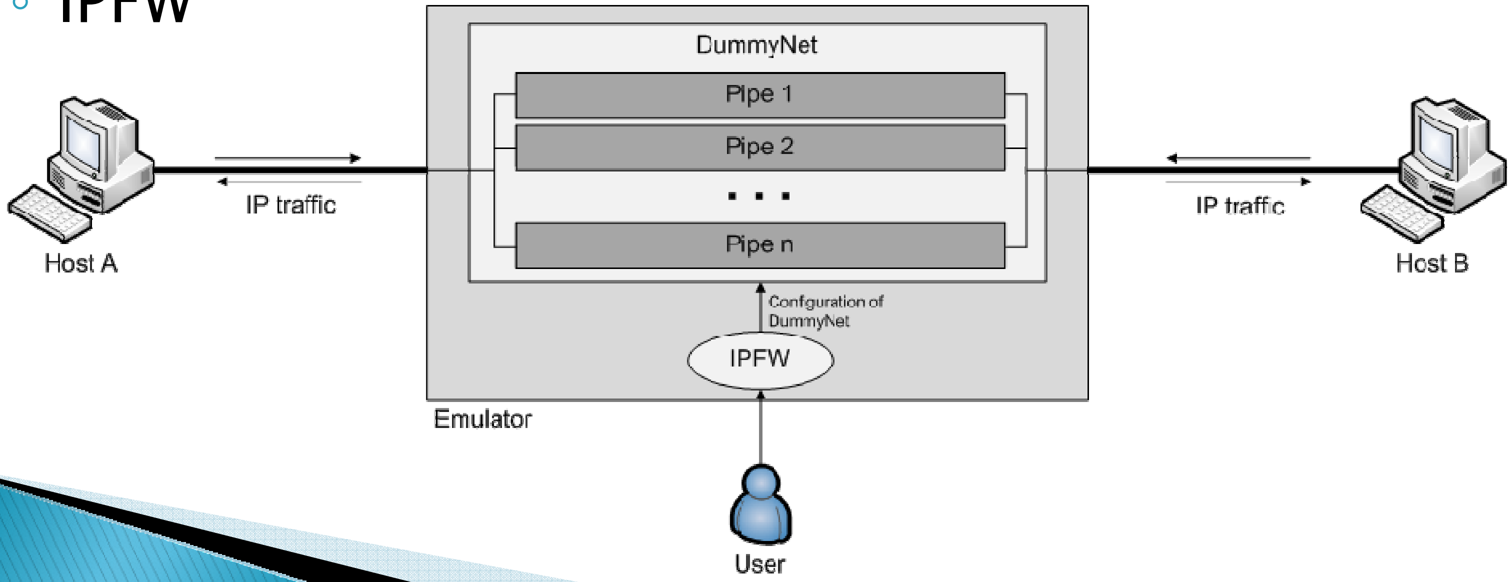
Tomas Hall
Andreas Midestad

Contents

- Background
- Problem description
- Design considerations
- Results
- Design
- Implementation
- Summary

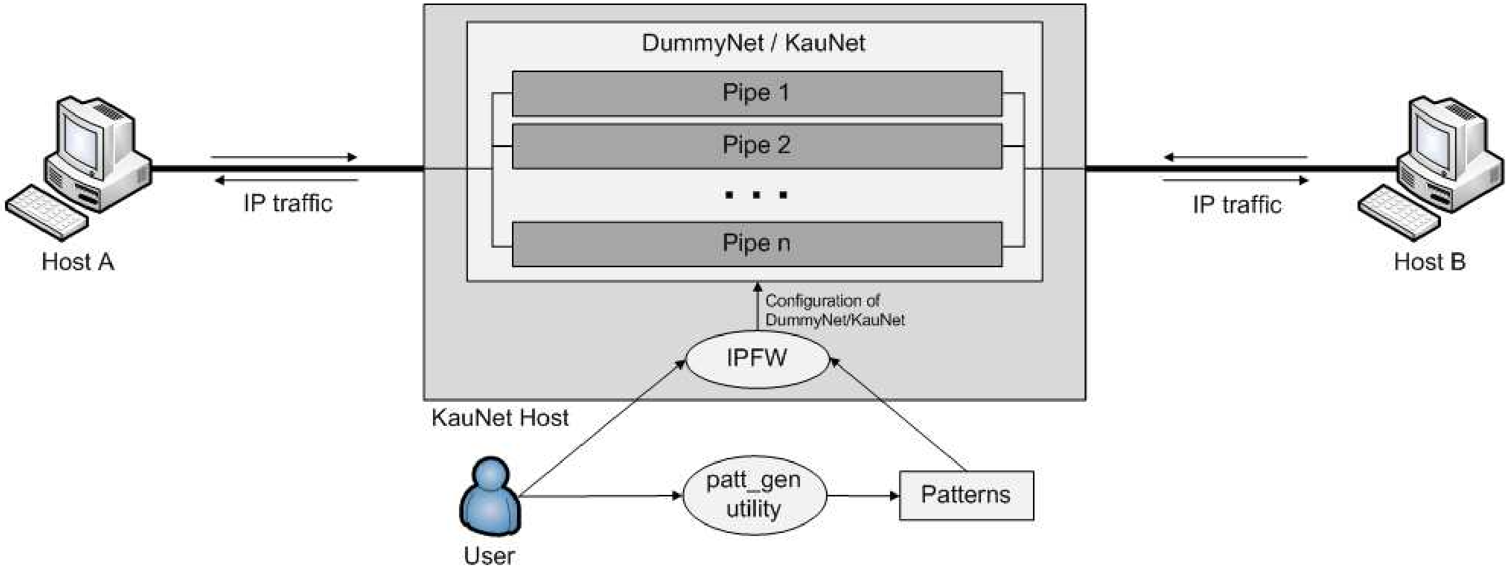
Background

- Evaluation of computer network systems
  - Theoretical, live testing, simulation, emulation
- Dummynet
  - Pipes – Simulated link & traffic filtering
  - Probabilistic emulation effects
  - IPFW



Background

- KauNet
  - Extends Dummynet
  - Deterministic, pattern based emulation
  - Pattern generation utility



Background

KauNet - Patterns

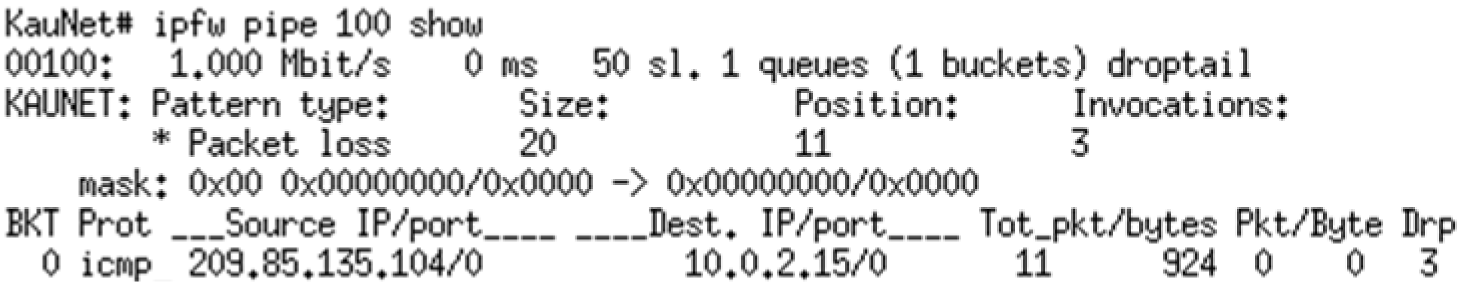
- Reproducibility
  - Reuse of patterns

- Modes of operation
  - Time-driven
  - Data-driven

- Types of patterns
  - Bandwidth change
  - Packet loss
  - Delay change
  - Bit-error

Problem description

- On-demand statistics



```
KauNet# ipfw pipe 100 show
00100:   1.000 Mbit/s    0 ms   50 sl. 1 queues (1 buckets) droptail
KAUNET: Pattern type:       Size:           Position:       Invocations:
        * Packet loss       20              11              3
    mask: 0x00 0x00000000/0x0000 -> 0x00000000/0x0000
BKT Prot ___Source IP/port____ ____Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp
  0 icmp  209.85.135.104/0           10.0.2.15/0        11      924  0    0   3
```

Problem description

- Desired functionality
  - Send *event information* (trigger value) to *subscribers* (trigger passing)
  - Pattern based event passing (trigger patterns)
- Examples
  - Real-time emulation updates/statistics
  - Cross-layer optimization
  - Link properties estimation

Design considerations

- Pattern synchronization (example)

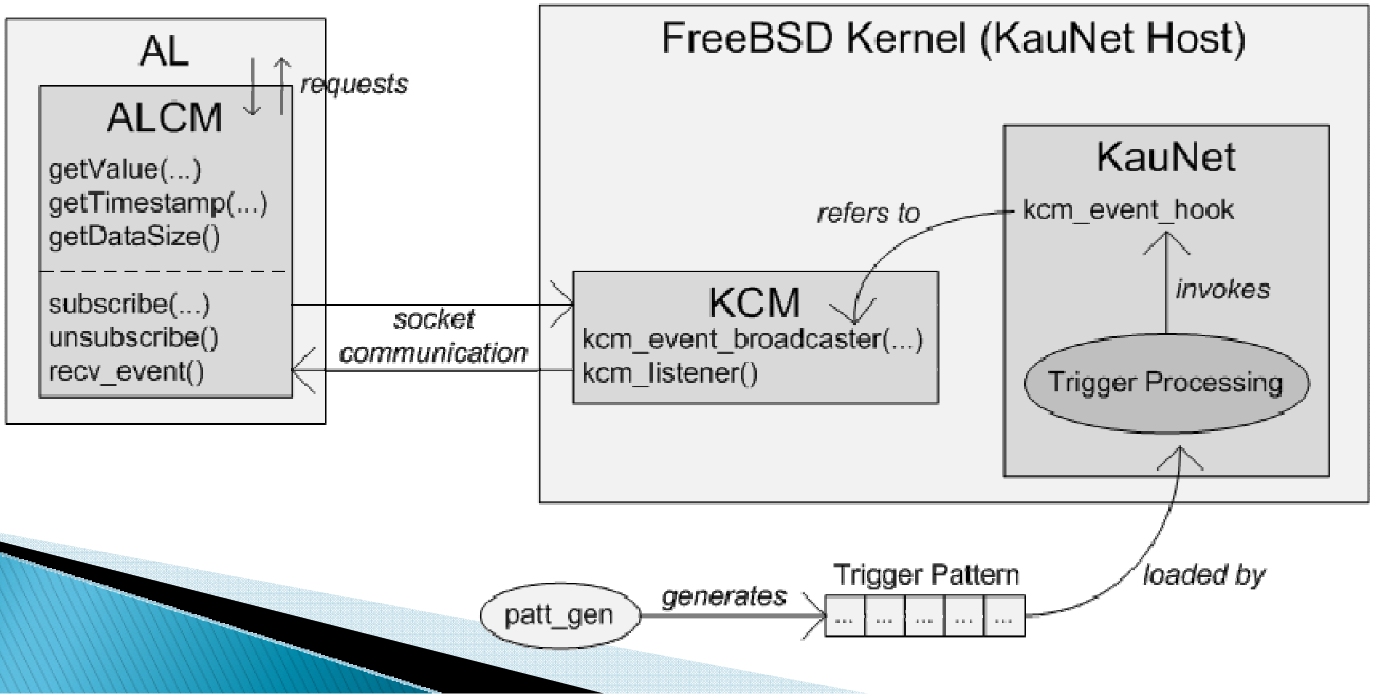
Packet	1	2	3	4	5	6	7	8	9	10
Packet loss pattern	1	0	0	0	0	1	0	0	1	0
Trigger pattern	42	0	0	0	0	29	0	0	37	0

  - patt_gen –pkt –pos loss.pat data 10 1,6,9
  - patt_gen –trig –pos trg.pat data 10 1,42,6,29,9,37

- Send 4 bytes of data reliably at one event per millisecond (minimum).

Results

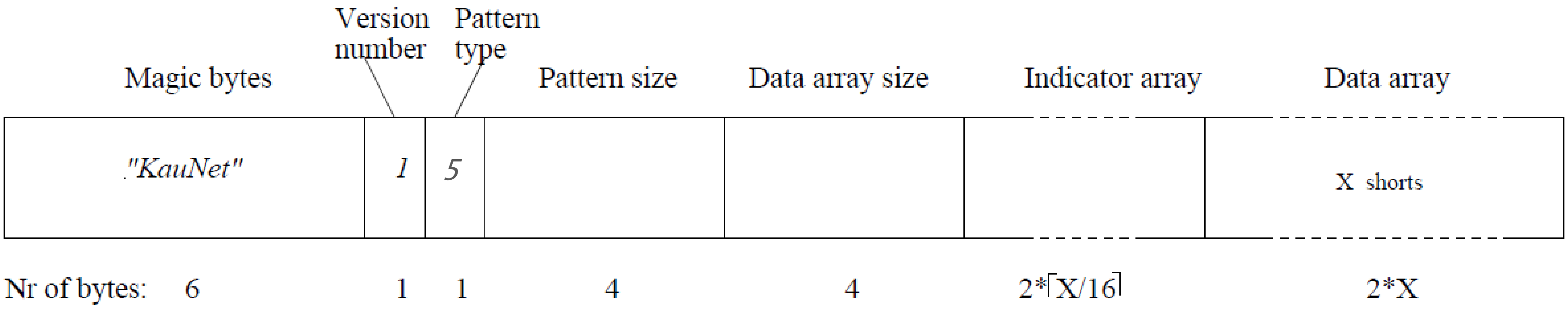
- Trigger pattern
- Trigger passing
  - KauNet communication module
  - Adaptation layer
  - Adaptation layer communication module



Design

Trigger pattern

Trigger pattern structure:



	Magic bytes	Version number	Pattern type	Pattern size	Data array size	Indicator array	Data array
	"KauNet"	*1*	*5*				X shorts
Nr of bytes:	6	1	1	4	4	2*⌈X/16⌉	2*X

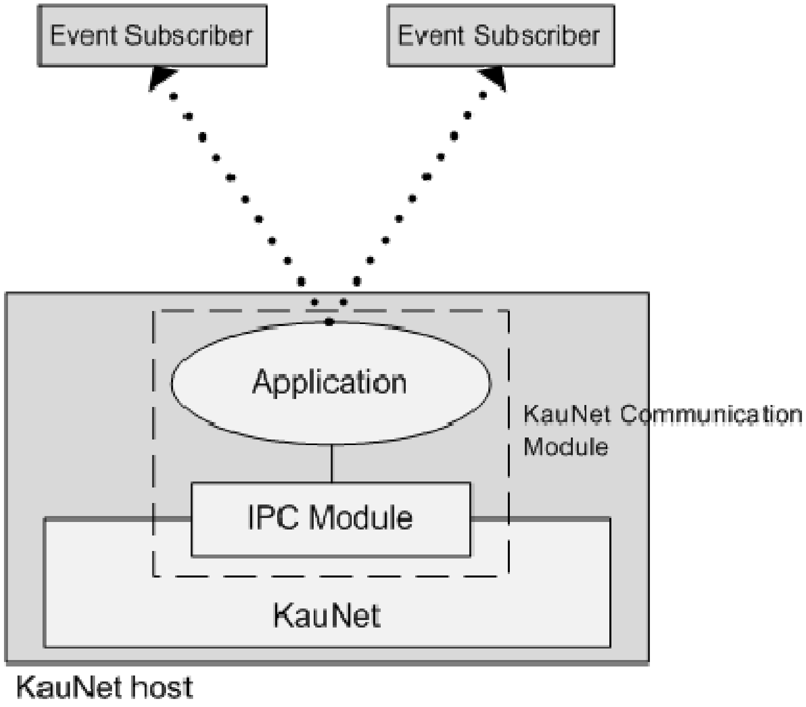
Design

Trigger passing

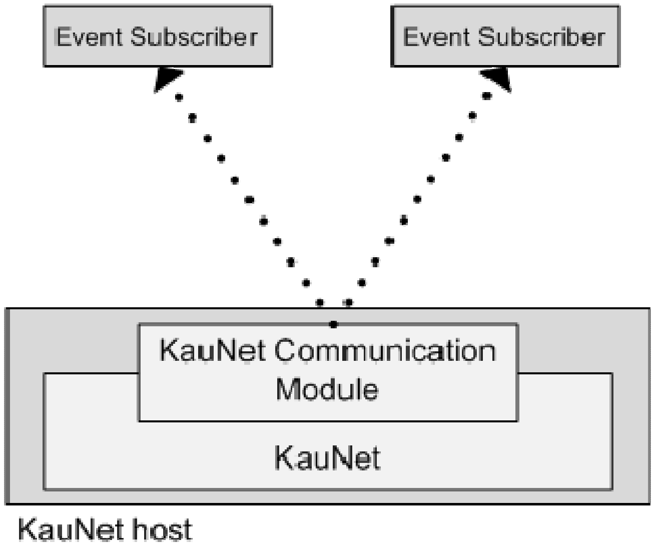
- KauNet Communication Module (KCM)
  - Receives events from KauNet
  - Forwards events to subscribers
  - Handles subscribers
  - Kernel module – KauNet plugin
    - Simplifies implementation
  - Modular design

Design

Trigger passing – KauNet Communication Module



Local IPC



Network sockets

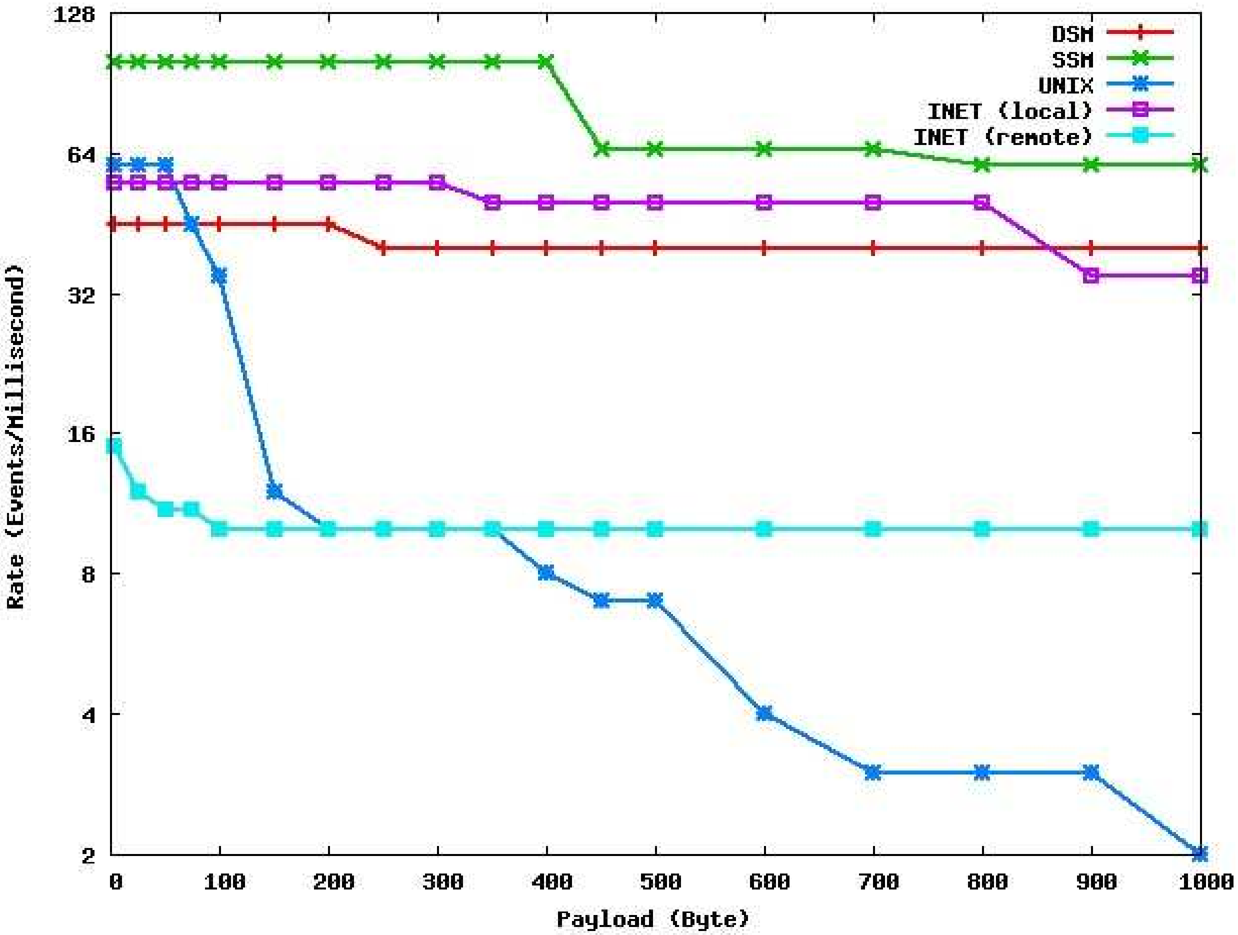
Design

Trigger passing evaluation – IPC mechanisms

- Signals (sigqueue)
    - Standard
        - Signal + 4 bytes data
    - With shared memory (dynamic, static)
        - Signal + 4 bytes key or offset
- Sockets
    - UNIX domain sockets
    - Network sockets (UDP)

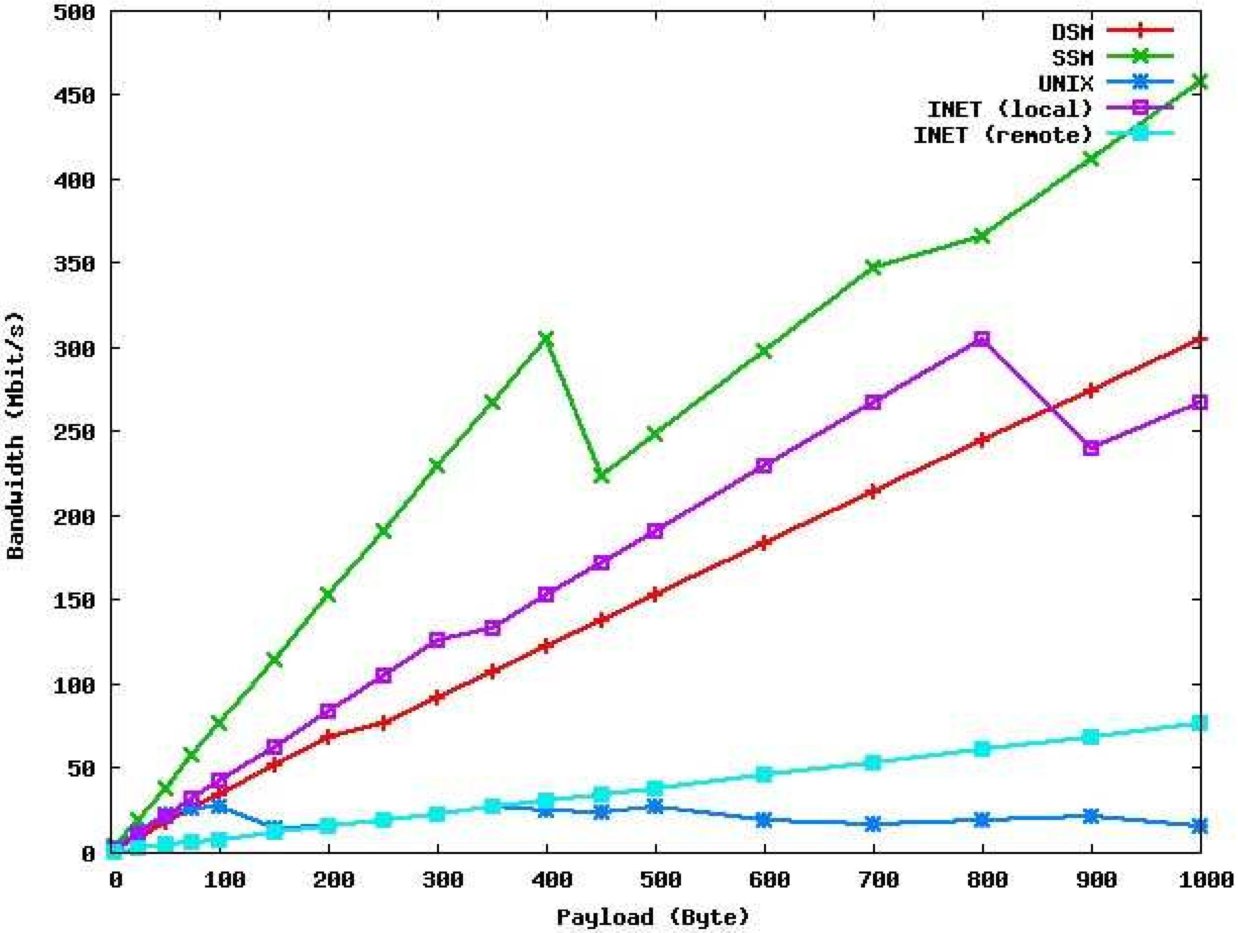
Design

Trigger passing evaluation – Results



Design

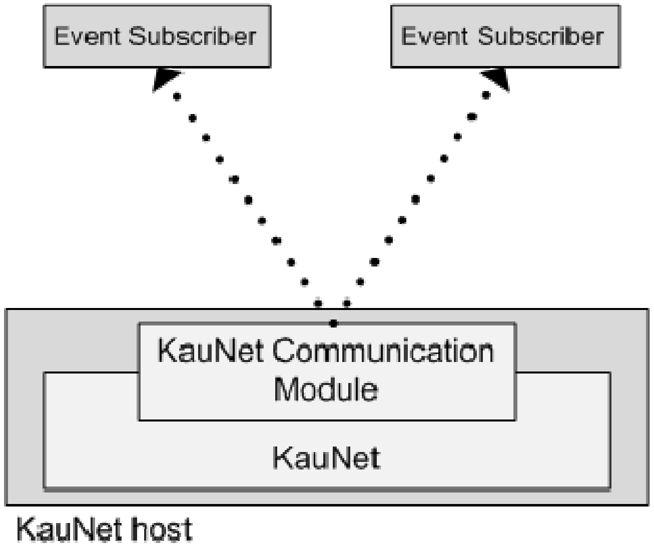
Trigger passing evaluation – Results



Design

Trigger passing evaluation – Summary

- Network sockets
  - Sufficient data rate and payload
  - Simplifies the design / implementation



Design

Trigger passing – Adaptation Layer

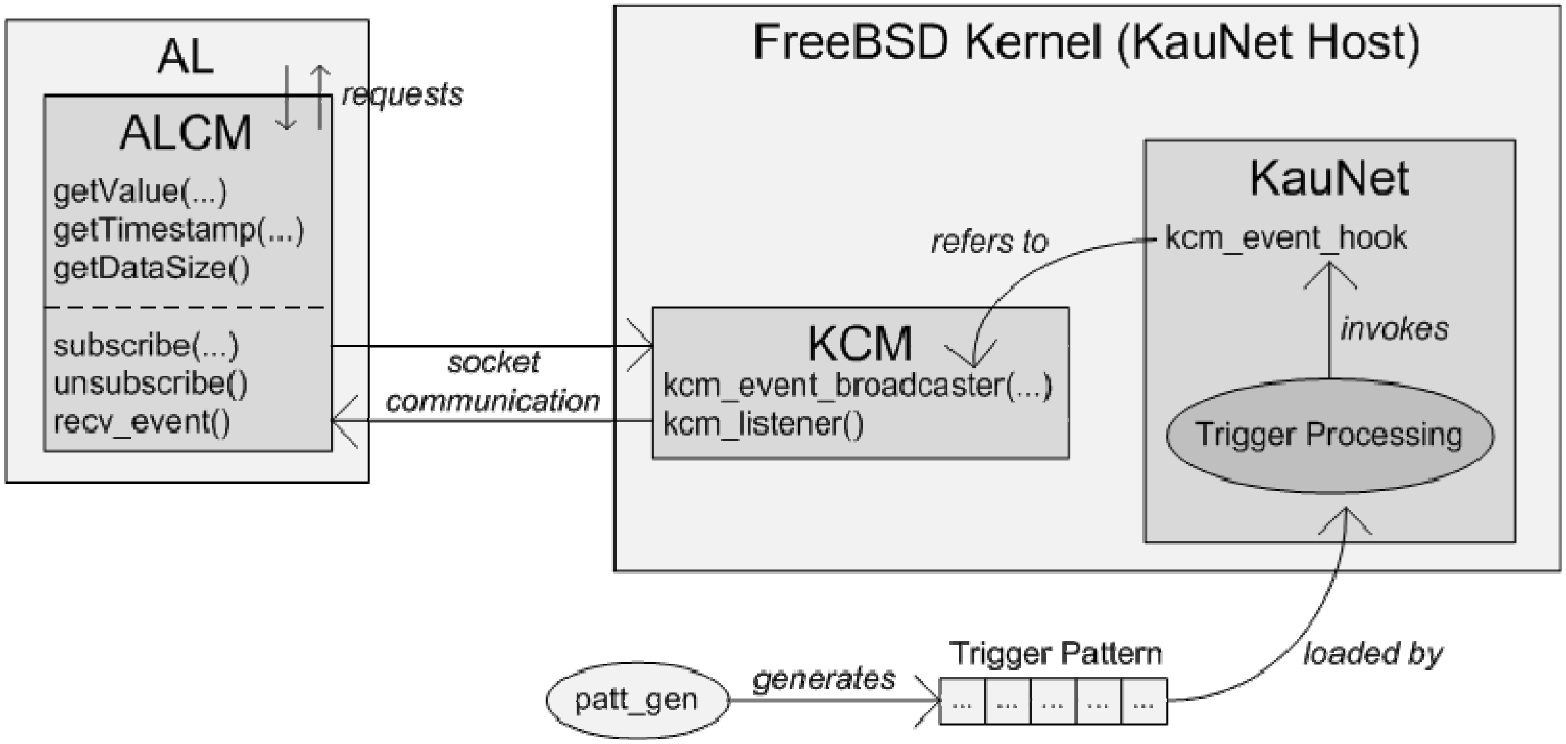
- Adaptation Layer
  - Arbitrary application which interprets received events
  - Experiment specific implementation
  - Uses the adaptation layer communication module to interact with KauNet

Design

Trigger passing – Adaptation Layer Communication Module

- Adaptation Layer Communication Module
  - C-library (API)
  - Simplifies adapation layer
  - Backward compatibility
- Functionality
  - Communicates with KauNet
  - Parses received events
  - No semantic awareness of events

Implementation



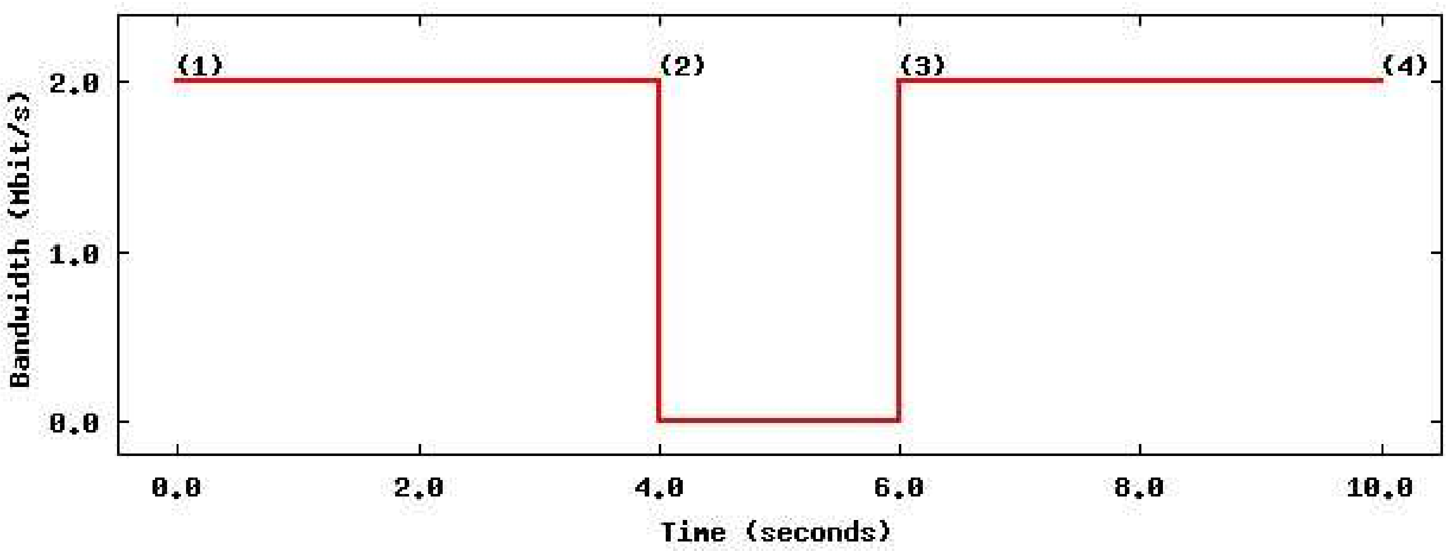
Summary

- Trigger pattern *generates* events
- Trigger passing *distributes* events
- Subscriber *interprets* events

- Enables:
  - Define when to generate events and what to send
  - Implement adaptation layer to interpret events

Demonstration

- Bandwidth change pattern
  - No bandwidth between the seconds 4 and 6
- Trigger pattern
  - Triggers at the seconds 0, 4, 6 and 10



Demonstration

Ping results

```
…
64 bytes from 10.0.2.1: icmp_seq=6 ttl=64 time=0.319 ms
64 bytes from 10.0.2.1: icmp_seq=7 ttl=64 time=0.322 ms
64 bytes from 10.0.2.1: icmp_seq=8 ttl=64 time=0.325 ms
64 bytes from 10.0.2.1: icmp_seq=13 ttl=64 time=0.257 ms
64 bytes from 10.0.2.1: icmp_seq=14 ttl=64 time=0.333 ms
64 bytes from 10.0.2.1: icmp_seq=15 ttl=64 time=0.334 ms
…


21 packets transmitted, 17 packets received, 19.0% packet loss
```

Demonstration

Adaptation layer output

```
Received value 1 at time 1260871692:574073 (12 bytes)
Received value 2 at time 1260871696:581029 (12 bytes)
Received value 3 at time 1260871698:585016 (12 bytes)
Received value 4 at time 1260871702:592739 (12 bytes)
```

Packet (#)	1	2	3	4	5	6	7	8	9	10	11
Time (s)	0		1		2		3		4		5
Trigger	1								2		

Packet (#)	12	13	14	15	16	17	18	19	20	21
Time (s)		6		7		8		9		10
Trigger		3								4