Case: FUNET measurments

Markus Peuhkuri

2006-04-20

Lecture topics

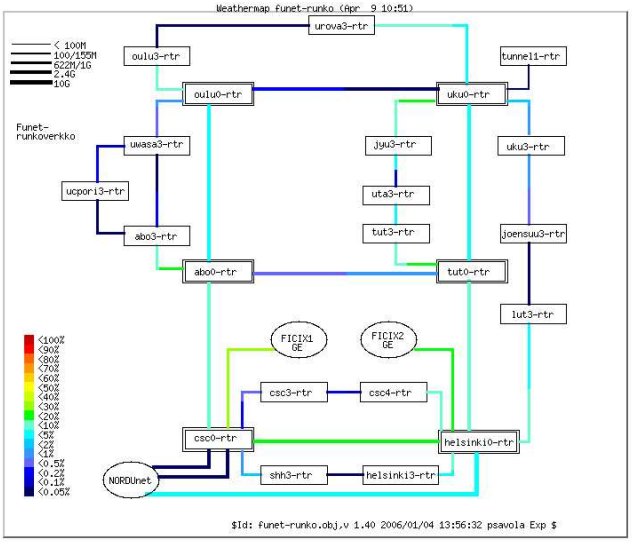
- Information about measurement location
- Information found

After this lecture you should

- Have some ideas how to do conduct measurements on high-performance links
- Be able to avoid some of problems that may arise

The measurement location

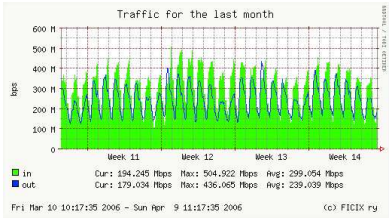
- FUNET core network
- OC-48 (2.4 Gbit/s) link between `csc0-rtr` and `helsinki0-rtr`
- Mixture of traffic



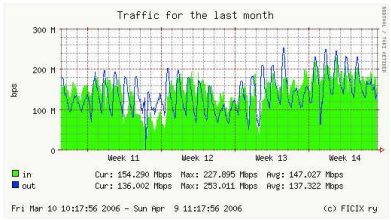
Traffic types

- Intra-Funet traffic
  - traffic between Helsinki University and CSC
  - in part East-West traffic
- University-ISP traffic: peering

  - FICIX 1



  - FICIX 2



- International traffic
  - Nordunet connections mostly via `helsinki0-rtr`
    ⇒ international traffic to/from Western Finland (including Espoo)

Three core problems

- Keeping with packets
  - standard or special hardware
- Data storage
  - efficiently store information
  - preserve interesting data
- Sensitive information
  - make data as insensitive as possible
  - preserve interesting data
- Select your trade-offs

Hardware requirements for dual OC-48 capture

- Things are pretty fast there
  - with average 500 B/packet over million packets per second. Note that in a DDoS situation the packet count may be tenfold.
  - capture on standard network cards just does not work even if there would be some OC-48 cards for PC
    * interrupt contention
    * standard PCI bus only slightly over 1 Gbit/s. A 64-bit, 133 MHz bus has still less than 10 Gbit/s
- Gigabit capture hard even with high-end systems
  - on-card queueing and interrupt throttling helps somewhat
- Special capture cards needed
  - the only problem is the price

Endace DAG capture cards

- Spin-off from the Waikato Applied Network Dynamics group
- We used DAG cards for OC-1 ATM (at time when 180 Mhz Pentium Pro was a fast processor)
- Time-critical work done on card
  - header capture
  - timestamping
  - packet transfer to memory (bus master)
- OS driver
  - card control
  - memory management
- Core memory allocated for circular buffer
  - reader process consumes data at its own rate
  - sufficient buffer provides safety from high load periods (256 MiB provides room for 3 million packets per card)
  - no interrupt problems: less than 100 interrupts per second

Code example

```
while (continue_capture) {
  if (top - bottom < record_size) {
    top = wait_bytes (device, &bottom);
  }
  record = (record_t *) (membase + bottom);
  rlen = ntohs(record->rlen);
  bottom += rlen;
  payload = record->payload + link_layer_header_len;
  process_packet(payload);
}
```

Compression by flows

- Better compression rate if you utilise structure of data than if data is "just bits"
  - header compression [3, 2, 1]
- Data in flows (5-tuple)

  TCP sequence, ack numbers proceed, possibly same size. If one is transferring a large file, then TCP segments are mostly same size, e.g. MSS

  UDP possibly same size. For example VoIP packets are the same, codec-dependent, size.

  1. keep track of every active flow (large id space)
  2. compare to previous packet
  3. short codes for common cases
- Normal data compression used on top of that

What data not to include

- IP identification + fragment word
  - changes randomly
  - for most studies no-use
  - adds 32 or 0/24/40 bits for each packet
- Checksums
  - no use afterwards, just check if OK (if possible)
- Length and header length fields implicit
- TTL field and TOS/DS byte should be constant in a flow
  ⇒ record changes
- User data
  - sensitive
  - for many analysis, all of payload should be included to be useful

Removing sensitive information

- IP addresses sensitive, as they may identify a single user or household
  - application ids (TCP, UDP ports) may be sensitive
- Approaches for sanitising addresses
  - how good anonymity provided
  - is topology information preserved
  - do multi-location measurements have same identifiers
  - can measurements done on different times have same identifiers

Random replacements for IP addresses

- A straightforward method
- For each IP address seen
  - allocate a new id if previously unknown
  - remap IP addresses to new ids
- Provides a good anonymity
  - allocation made in temporal basis
  - if address `192.0.2.1` is mapped to `1.0.0.1`, then `192.0.2.2` may have any mapping, like `4.7.1.8`
- Drawbacks
  - topology is lost
  - different sites have different ids
  - to reuse mappings later, the mappings should be saved
    ⇒ highly sensitive database

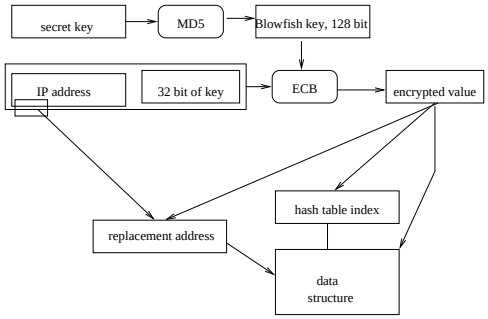
Prefix-preserving mappings

- Make upper part anonymous
  - removes organisational information
  - possible to identify organisations based on traffic volume
    ⇒ individuals could be identified
  - determining the right prefix length
  - good for some security related traces
- Make lower part anonymous
  - protects individual users
  - if only few IP addresses active on range, possible to identify users
  - makes possible to work on address aggregates

IP address desensitising

Initiate encryption (blowfish), then for each IP address

1. Encrypt
2. Check if in hash, if not then
   (a) insert into hash
   (b) write out record to stream
3. Replace real IP with 8 bits of clear and 24 bits from encrypted
   ⇒ codeIP



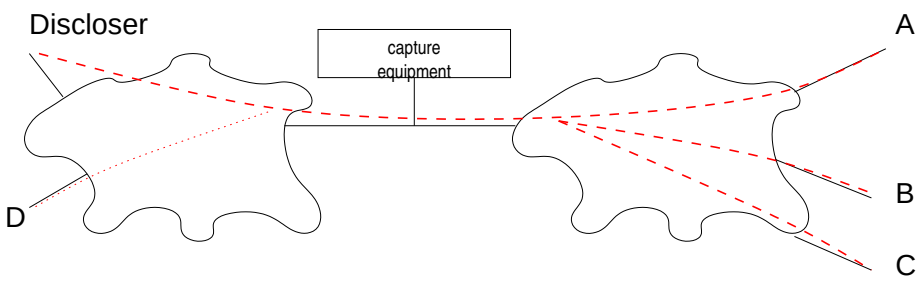
Using desensitised IP addresses

On decoding (off-line) each a time encryption record is found

1. Check if known mapping secret ⇒ anonIP, if not then
   (a) pick random unused IP from that network
   (b) store (secret,anonIP) to a (persistent) database
   (c) maintain hash table of codeIP ⇒ (secret,anonIP) mapping

Replace codeIP with anonIP in headers

Possible disclosure



By sending packets by using a temporal process or other mechanism making it possible to later identify packets from anonymised trace, a discloser can learn it's own IP address and IP addresses of A, B, and C. In security terms this is a covert channel used to disclosure sensitive information.

Capture machine

- Dual Xenon 2.4 GHz
- 2 GB of memory
- 120 GB system disk
- 4*160 GB IDE disks for data
- 2*1000BaseT NIC for trace transfers although only one connected for 100 Mbit/s
- Endace DAG 4.23 OC48 capture cards
- Linux 2.4.20
- Performance:

  Disk I/O write 77 MB/s = 617 Mbit/s
  (currently as 2 RAID-0 stripes)

  Compression / Anonymization initial tests:

  single-thread 2.5 Gbit/s (disk-disk)
  double-thread 7 Gbit/s (estimated based on CPU usage)

  Compression ratio about 12 bytes / packet, 1:40 reduction for wire speed
- Unfortunately, no conditional full capture

Data collected so far

- On average, 10 Mbit/s of compressed trace
  $\Rightarrow \approx 1$ TiB/week
- System properly running since May 2004
- No single packet lost! (according to card diagnostics) However, not every packet is analysed because of problems in analysis.
- Traces stored: 7.5 TiB
  - 4 complete weeks
  - 71 complete days, 24 partial
  - 2^{-28} s resolution (3.7 ns)
- Stateless statistics calculated for most of data

Storage and analysis

- Daily volume
  - 23rd December: 55 GiB
  - 26th September: 124 GiB
- Weekly volume: 600 – 750+ GiB
  $\Rightarrow$ to do a week-long analysis more than 1 TiB disk capacity needed
- Capture machine keeps disks 90 % full
- Problems with CSC tape archive
  - maintenance periods

  - results holes if data ages
    ⇒ need for temporary storage as buffer

- Analysis needs 1 GiB memory to start with
  - needed to buffer IP address DB
  - more for stateful analysis
  - basic flow analysis about real-time

Stateless statistics calculated for all data

- IP protocol (TCP, UDP, . . .) counts for every second
  - packet length histogram (40, 64, 128, 256, 512, 1024)
- OSPF packet timestamps and lengths
- For every 10-minute file
  - TCP, UDP port counts and packet length histogram
  - TTL histogram by IP protocol
  - TCP retransmissions/reorders in 32k byte window

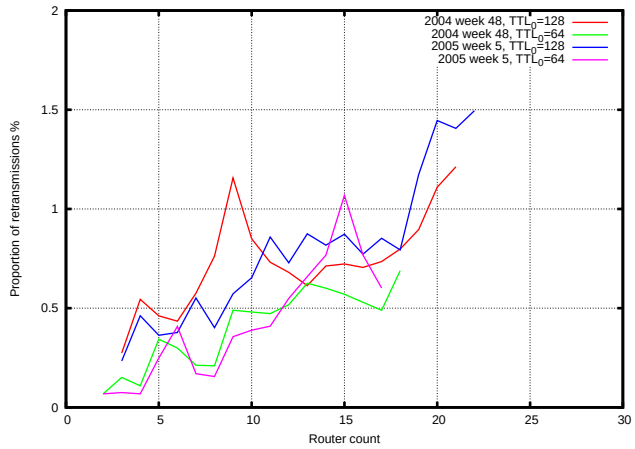
Analysis made based on data

- Protocol distribution
- Evaluation of TCP retransmission compared to TCP ports
- Traffic matrix analysis
- Routing protocol analysis
- Packet IAT analysis (by VTT)
- Routing table lookup performance (by KTH)

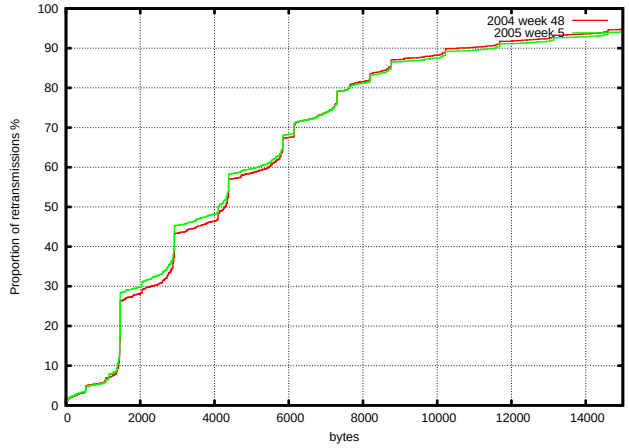
Findings: protocols used: week 2004/48 and 2005/5

- TCP protocols
  - top 10 ports use 25 – 30 % of bandwidth
  - http (16 – 20 %), nntp (20 –30 %)
  - p2p traffic halved
  - email traffic 0.5 – 0.7 %
- UDP protocols
  - game traffic
  - dns
  - malicious traffic

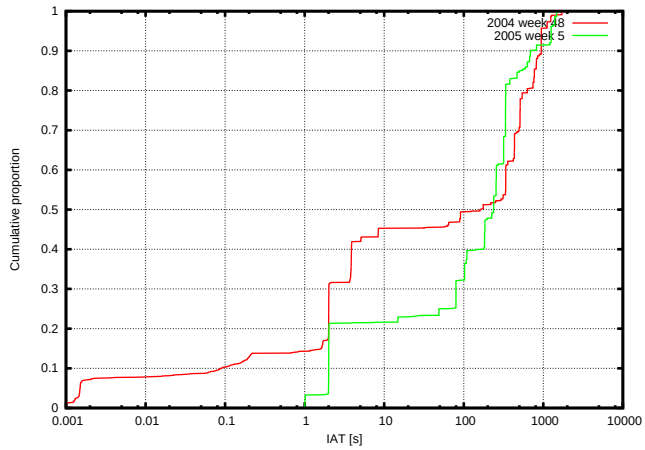
Findings: TCP retransmissions and hop count



Findings: TCP retransmission distance



Findings: OSPF non-Hello IAT



Lessons learned

- Data capturing easy with proper devices
  - must shovel some €€€
  - even low-cost disk subsystem can keep with speed
- Well-balanced backend system

  - enough storage buffer
  - reliable storage system
- Flow compression efficient, about 5:1 ratio. Flow compressionm itself about 3:1 ratio and gzip compression 3:2 ratio on top of that.
- Analysis should be put more effort

Conclusion

- Core network measurements provide insight to real traffic
  - traffic processes
  - traffic directionality
- Important resource for evaluating models

References

[1] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. Request for Comments RFC 3095, Internet Engineering Task Force, July 2001. (Internet Proposed Standard) (Updated by RFC3759). URL:`http://www.ietf.org/rfc/rfc3095.txt`.

[2] M. Degermark, B. Nordgren, and S. Pink. IP Header Compression. Request for Comments RFC 2507, Internet Engineering Task Force, February 1999. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc2507.txt`.

[3] V. Jacobson. Compressing TCP/IP headers for low-speed serial links. Request for Comments RFC 1144, Internet Engineering Task Force, February 1990. (Internet Proposed Standard). URL:`http://www.ietf.org/rfc/rfc1144.txt`.