Some Observations of Internet Stream Lifetimes

CAIDA/WIDE, Los Angeles, 12 Mar 05

Nevil Brownlee

CAIDA and *The University of Auckland*

Overview

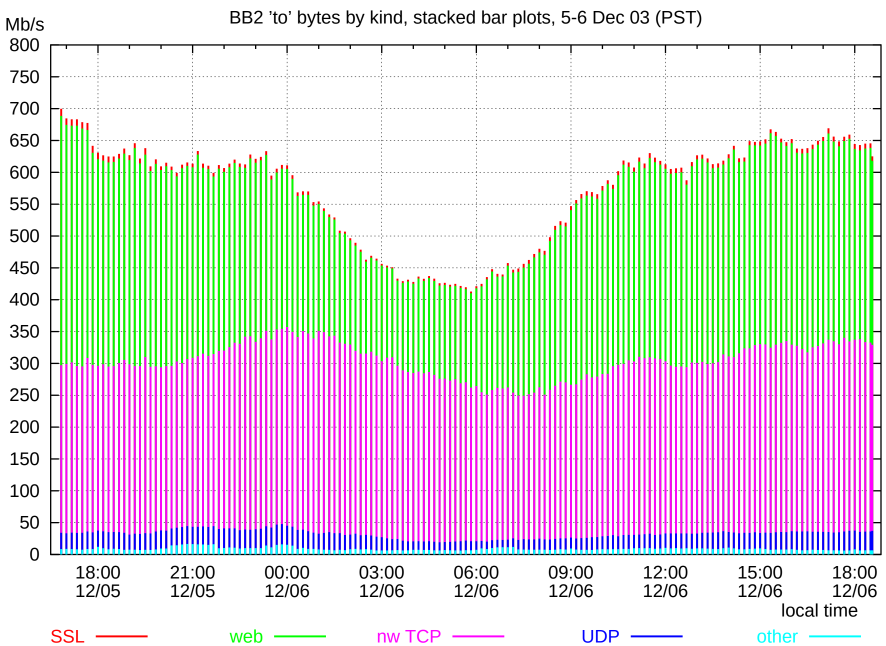
- Introduction, traffic flows
- Streams, stream density plots (packets and bytes)
- NeTraMet: implementation, performance
- Streams and packets at Auckland
- Usage metering, strategies to reduce meter overhead
- Effect of ignoring small streams
- Conclusion

Introduction

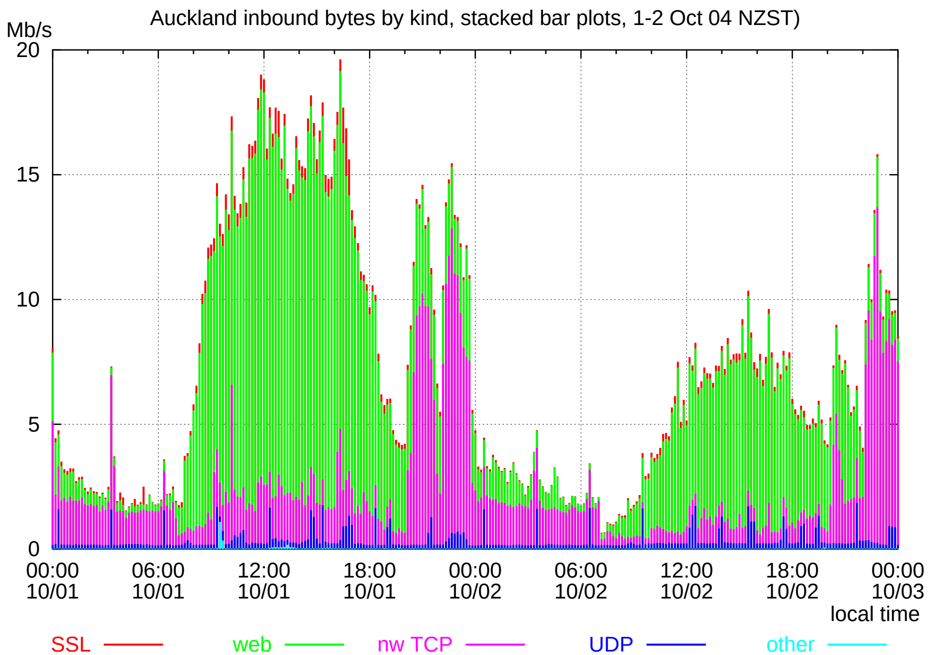
- A *traffic flow* is an abstraction representing the set of packets involved in some network activity
- There are two main classes of flows
  - CPB (unidirectional, 5-tuple, fixed timeout).
    Also known as *microflows*
  - RTFM (bidirectional, general, fixed timeout).
    User writes a *ruleset* to specify flows using values for a large set of *attributes*, and specifying direction
- Streams
  - are subsets of RTFM flows (bidirectional, 5-tuple, dynamic timeout)
  - more details later . . .

Traffic rate plots

- Count bytes in five flows for different kinds of traffic
- Match packets on protocol and port number:
  - *SSL* = TCP 443, *web* = TCP 80, *nw TCP* = other TCP ports
    UDP = all UDP, *other* = all other protocols



Ca Backbone, 6 Dec 03 (PST)

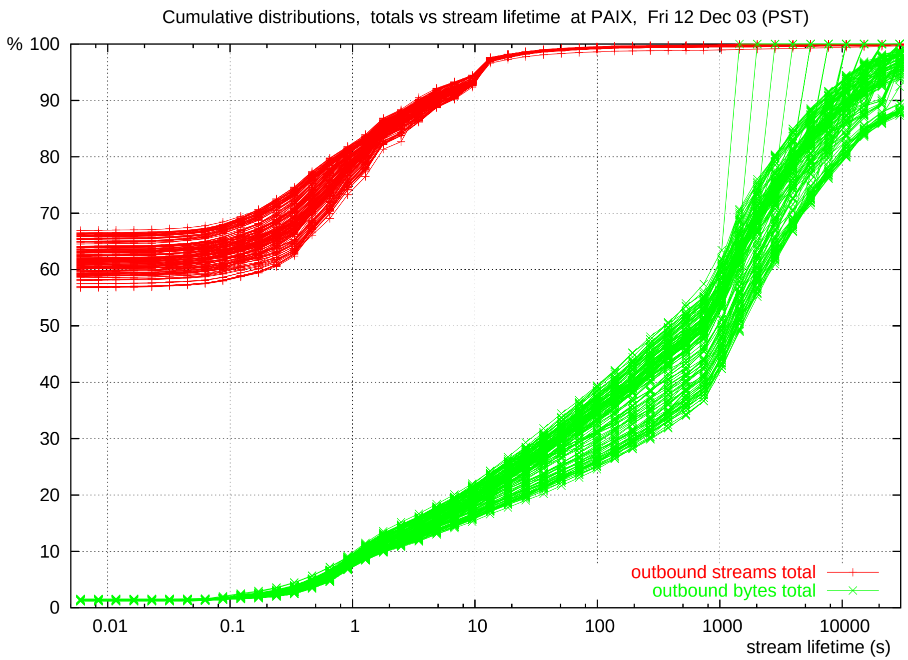


U Auckland, 1-2 Oct 04 (NZST)

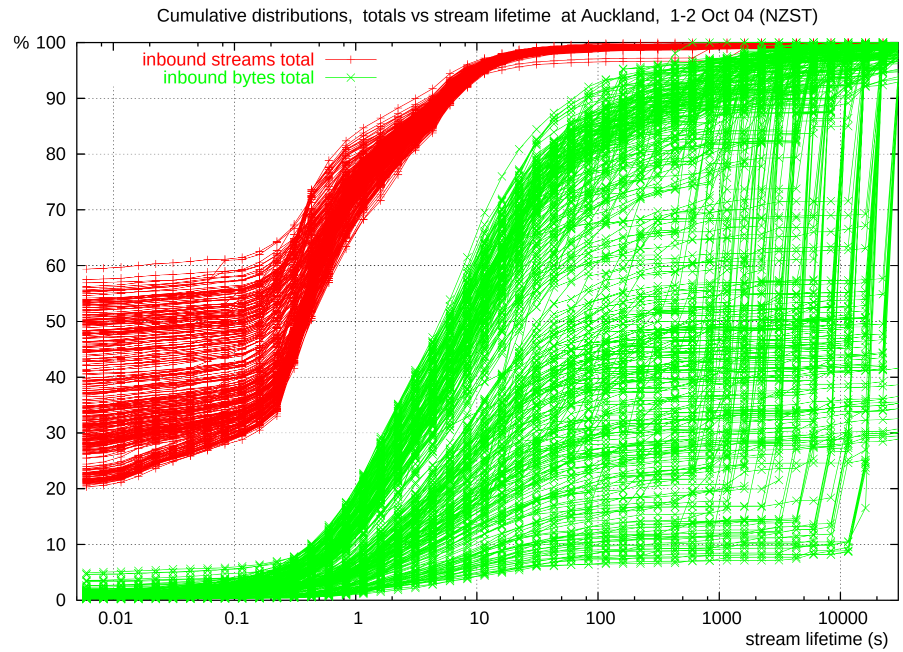
Streams – why are they *useful?*

- *Streams* allow NeTraMet to compute metrics for components of flows, e.g. RTTs and IATs
- NeTraMet can return distributions of those metrics as attributes for such flows
- For the stream-distribution attributes ..
  - lifetimes $<=$ 15m are counted directly
  - longer streams are treated as flows; we sum their data each interval to produce distributions with lifetimes up to 30,000s ($\approx$ 8h)
- The five different kinds are summed to produce 'total traffic' distributions at 10m intervals

Stream & byte density vs lifetime plots



Ca Backbone, 6 Dec 03 (PST)



U Auckland, 1-2 Oct 04 (NZST)

- At both sites, 95% of streams last $\leq$ 10s
- At U Auckland, up to 65% of the bytes are in streams $\leq$ 10s
- On the Ca backbone, only 20% of the bytes are in streams $\leq$ 10s, and about 60% of the bytes are in streams $\leq$ 1000s!

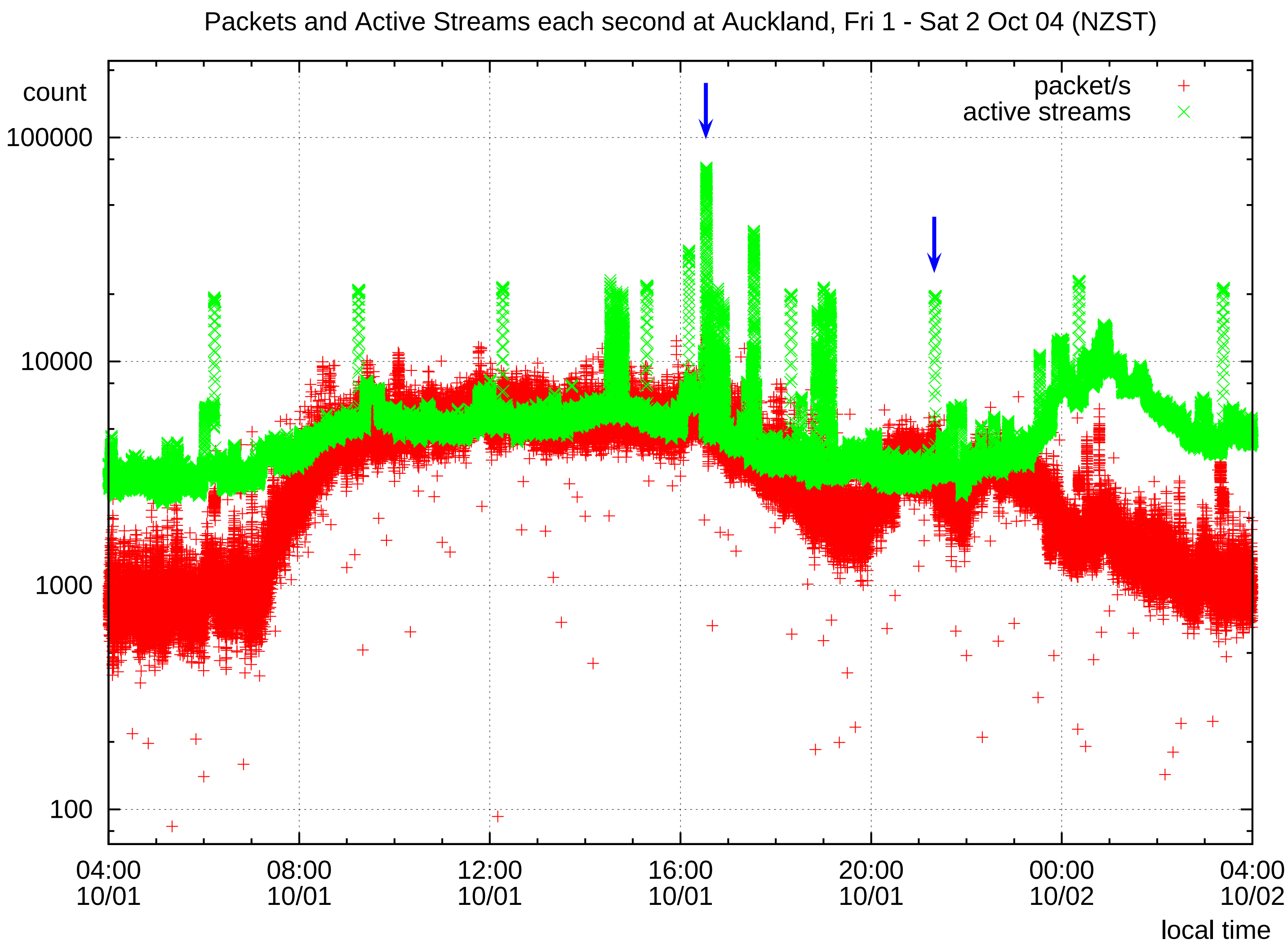
NeTraMet implementation details

- NeTraMet is an RTFM meter – user must write ruleset(s) that specify:
  - which flows to count
  - which end-point is the source
  - how much detail is to be reported
- Uses stream caching:
  - does flow matching for first packet of stream, saves flow number(s)
  - uses cached flow number(s) for later packets
  - *can't cache for rulesets that use non-5-tuple attributes*
  - usually gets $\approx$90% cache hit rate

NeTraMet performance

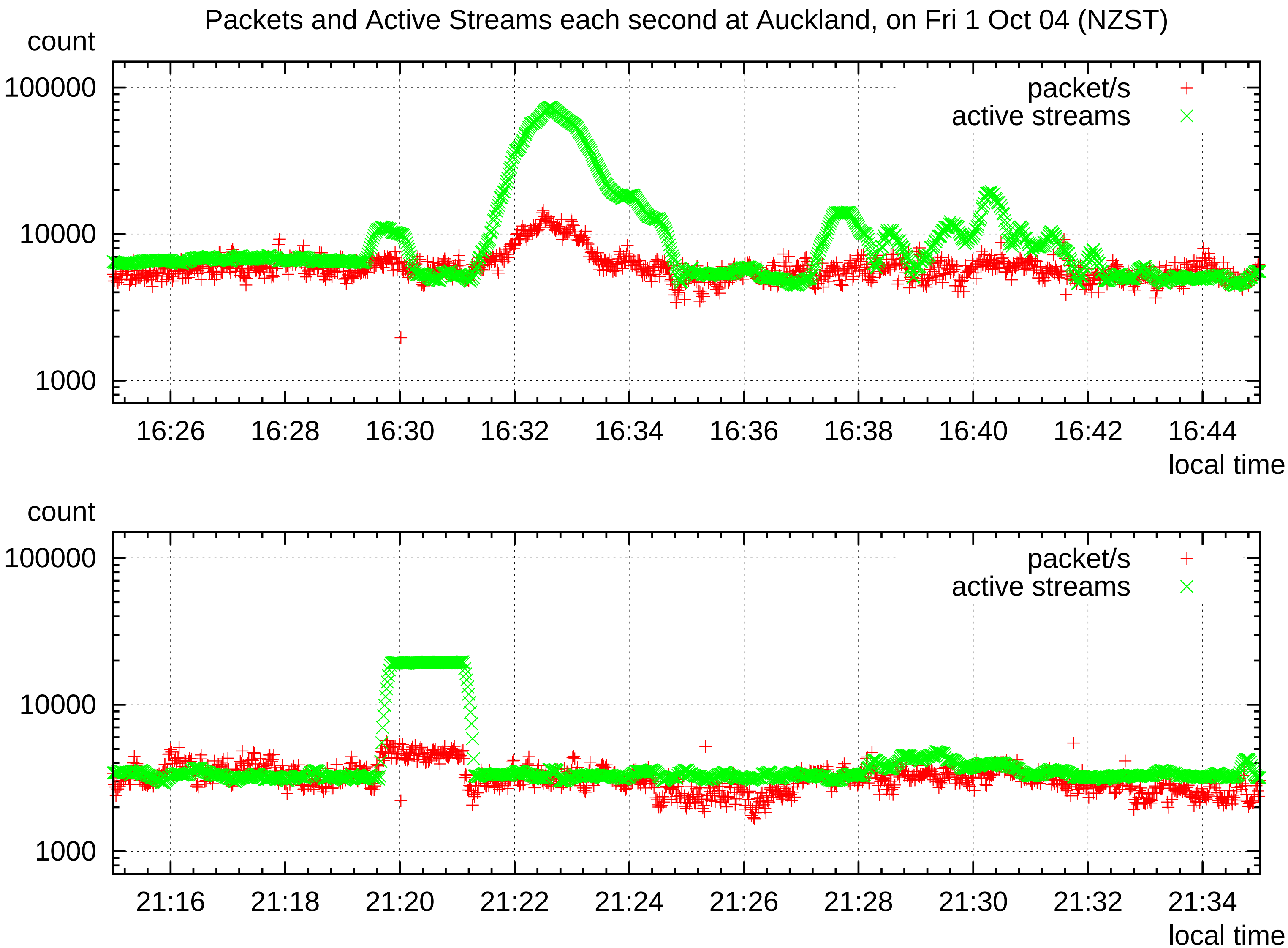
- 1 Gb/s testbed, 1-processor meter, 1 DAG card
  - 1500B frames, 1000Mb/s traffic
    NeTraMet sees 164 kp/s, reports 996.6 Mb/s
  - 128B frames, 130 Mb/s of traffic
    NeTraMet sees 219 kp/s, reports 123.2 Mb/s
- Higher frames rate cause meter to ignore packets if they're sustained for more than a second or two
- OC48 backbone, 2-processor meter, 2 DAG cards
  - 600 Mb/s traffic
    NeTraMet sees 215 kp/s, no lost packets
- Tests performed in 2003 and 2004. Working on further speed improvements

Streams vs time at Auckland



- ⋆ Stream numbers follow the packet rate
- ⋆ High spikes about every 3 hours
- ⋆ Peak around midnight, 2 Oct, was not part of the diurnal pattern – it didn't recur

Details of Auckland stream spikes

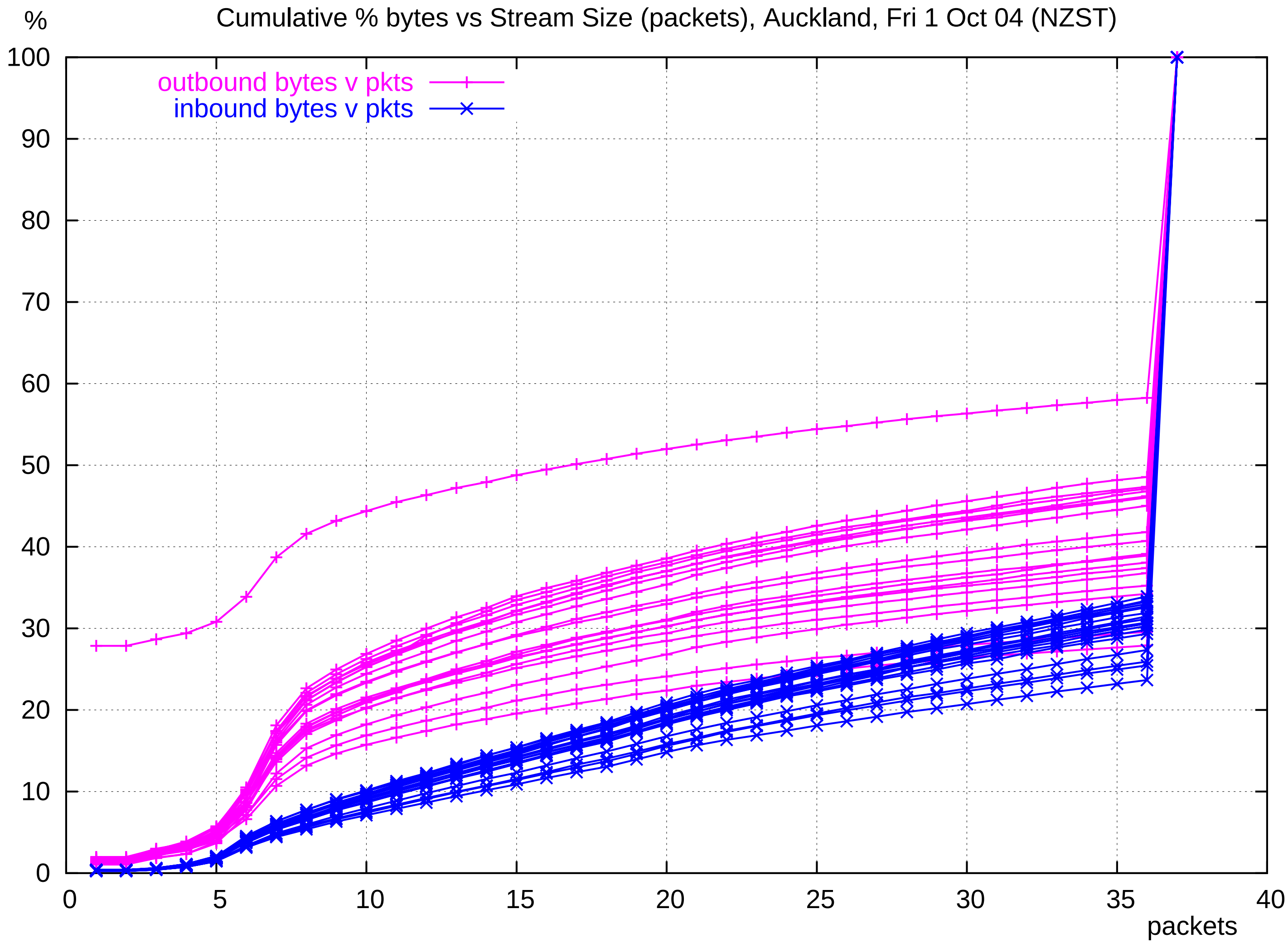


⋆ *Note that increase in streams ≫ increase in packet rate!*

Usage metering at Auckland

- High peaks in stream numbers load the meter, especially if many of them map to new flows
- Such peaks load the meter reader (data collection system) too
- We want to understand the peaks so that we can summarise them as special kinds of flow
- To start with, what is the effect of ignoring streams $\leq K$ packets in size?
- What % of bytes are ignored for various K values?

Auckland *byte density* vs *stream packets*



- ⋆ Three hours of data, 10-minute intervals
- ⋆ Seems safe to ignore streams ≤ 6 packets
- ⋆ But one interval looks *different !?*

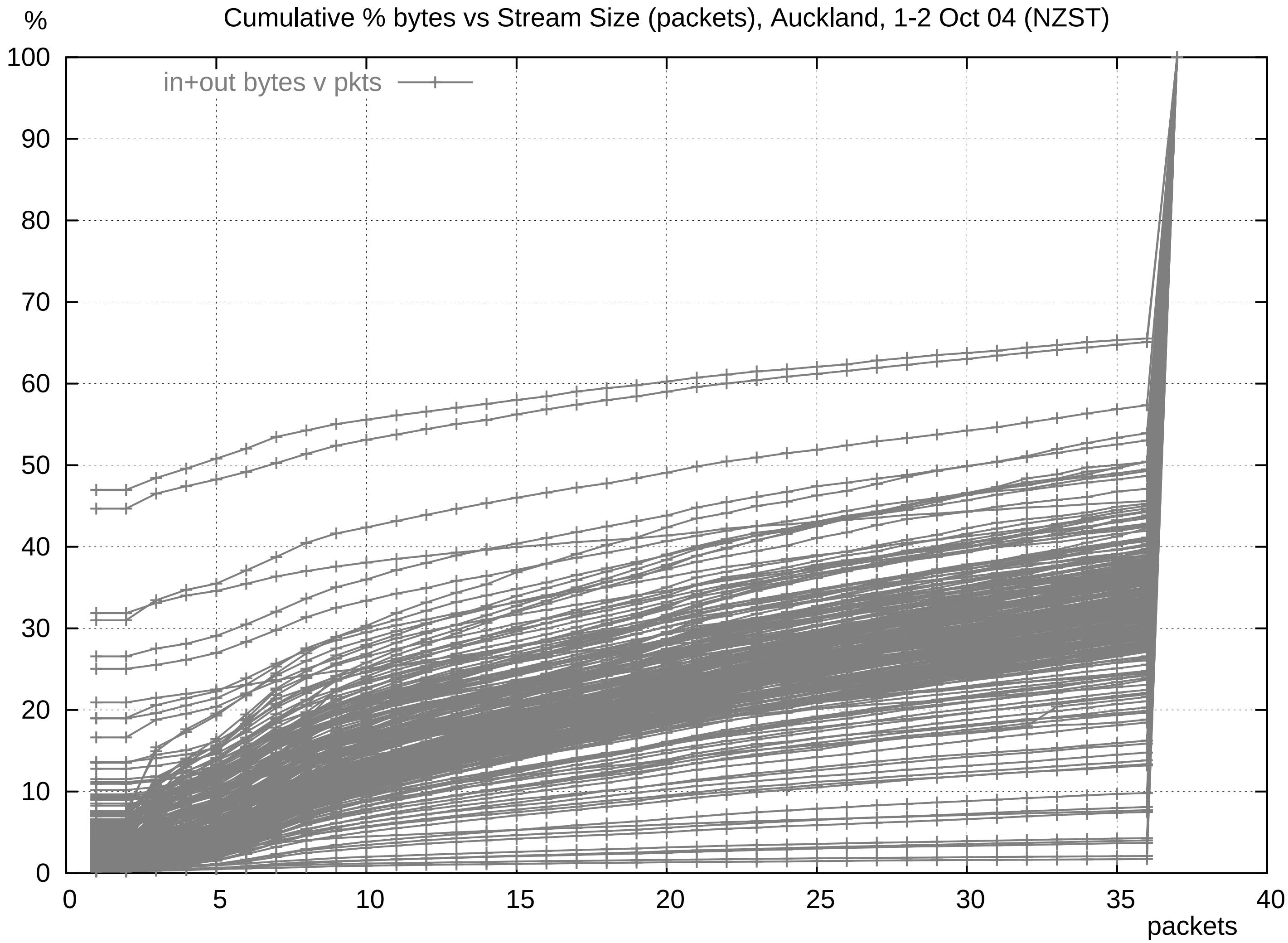
Intervals with high small-stream %

Inbound rate	UDP	non-web	web	SSL	other
2110	0.15	2.91	8.85	0.51	0.03
2120	*1.66*	2.23	10.15	0.52	0.04
2130	0.21	1.37	9.86	0.50	1.09

Outbound rate	UDP	nonweb	web	SSL	other
2110	0.10	1.47	3.31	0.73	0.03
2120	0.10	*0.92*	3.34	0.850	0.03
2130	0.10	3.71	3.54	0.859	0.07

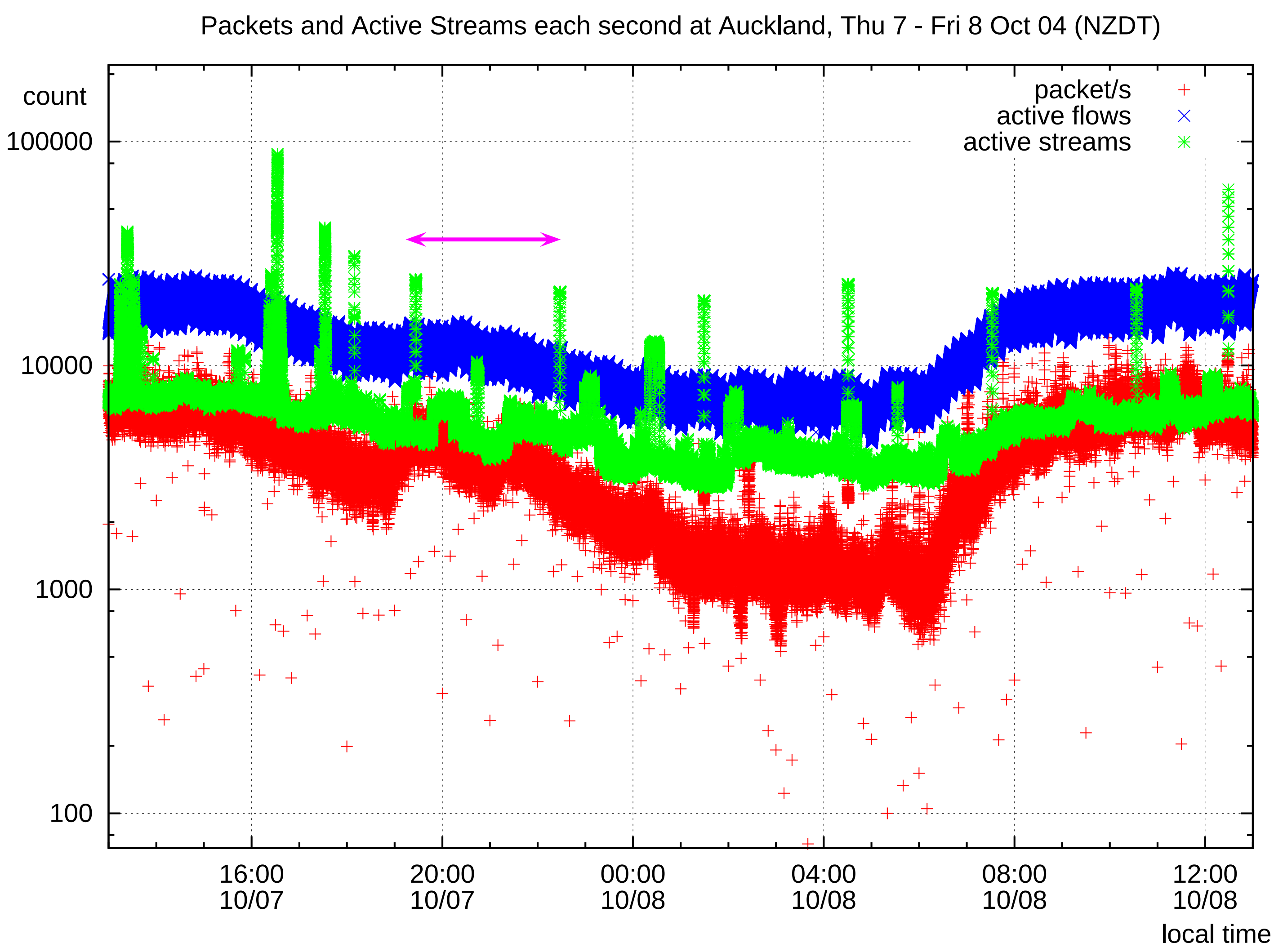
- Tables show Mb/s rate for each traffi c kind
- Seldom saw low outbound non-web TCP, often saw high inbound UDP
- High inbound UDP rate
  - most small streams don't generate a response
  - those that do dominate outbound traffi c

Auckland *in+out* byte density



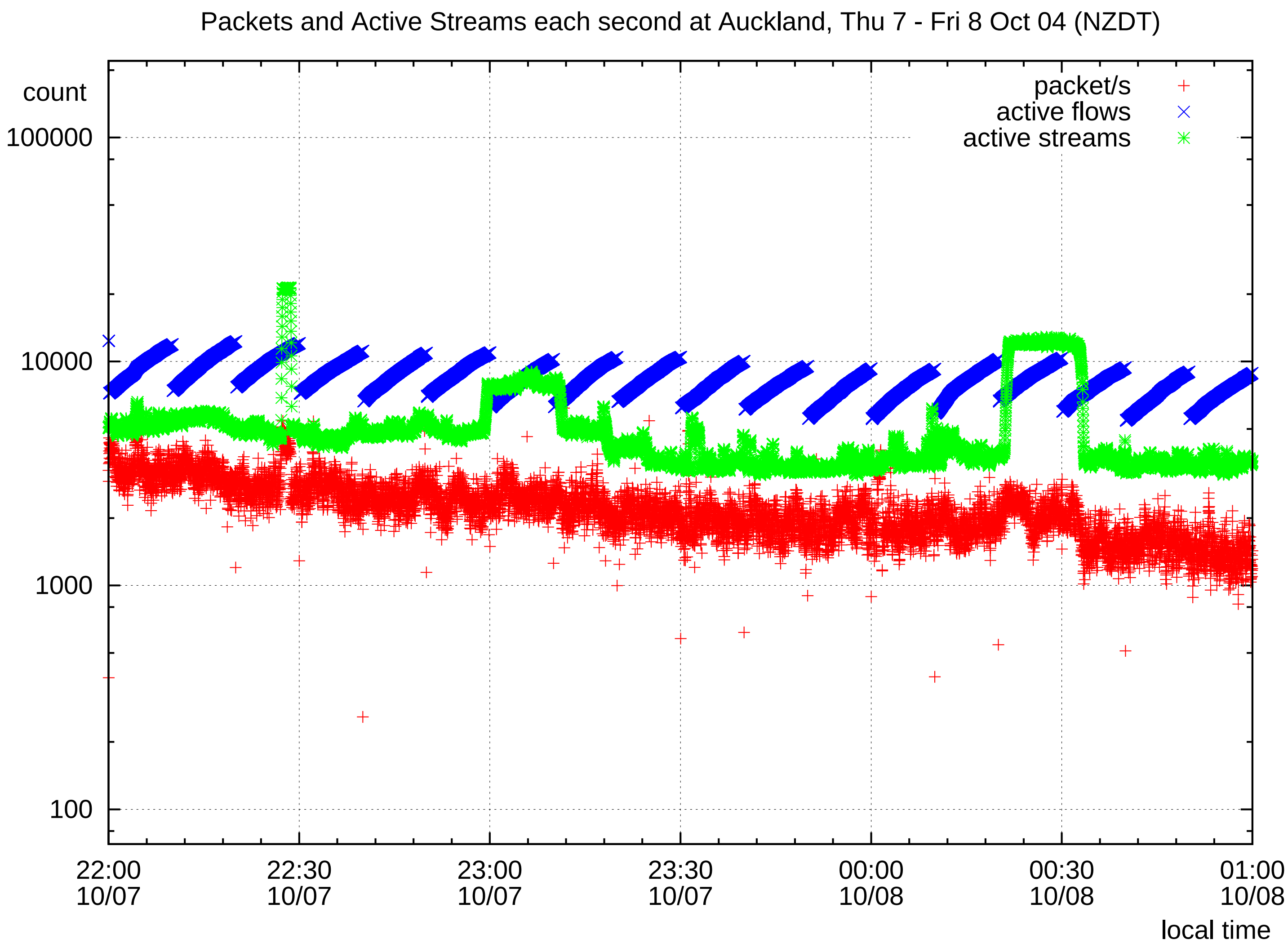
- ⋆ Two days of data, 10-minute intervals
- ⋆ ‘Outlier’ traces similar to previous plot
- ⋆ Need to understand the small streams
- ⋆ Can’t just *focus on the elephants*

What happens if we ignore small streams?



- ⋆ Similar to earlier plot
- ⋆ Here we show number of flows too
- ⋆ Flows track streams, no spikes
- ⋆ Confirms that spikes come from short streams

Ignoring small streams – detail plot

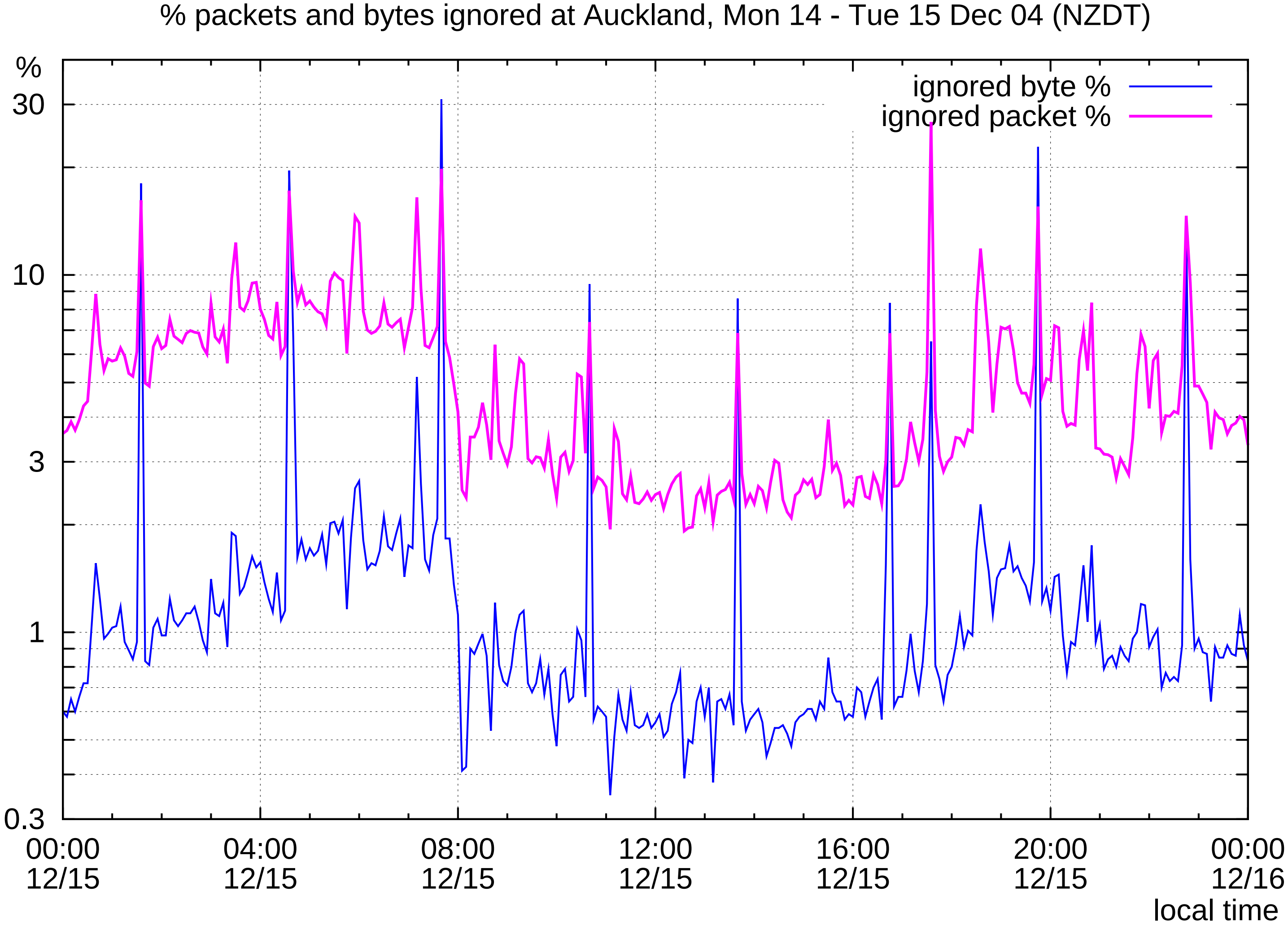


- ⋆ Flows build up during interval, then drop when meter is read
- ⋆ Average number of flows remains stable even during stream spikes

Counting the ignored packets

- We modified the NeTraMet meter to count bytes from ignored streams
- Counts are in *LtMinStreamPDUs* and *LtMinStreamOctets* distributions, held in a special *LtMin* flow
- We plotted the sum of these distributions for two days of 10-minute intervals . . .

***Packets & bytes ignored* in small streams**



- ⋆ Ignored bytes below 2% except during spikes
- ⋆ Ignored packets stays below 10% similarly
- ⋆ Less than 7% of intervals (about 1 in 15) are spikes

Summary

- ‘Ignore short streams’ strategy is simple and effective
- Our approach is *not* sampling –
  - we don’t completely ignore short streams
  - we count them in the LtMin distributions
- We’re continuing to look at *plague of dragonflies* behaviour, so we can count them as special cases. LtMin distributions are only our first attempt at this
- Sampling, using adaptive parameter setting (Moore et al., [7]) is an alternative technique, especially at very high line rates
- It would be interesting to compare the two techniques

Conclusion

- Operators and researchers need to understand traffic patterns so as to recognise special cases
- Adaptive sampling will probably work well in all cases
- But we can't do that if we want a complete record for investigating security incidents

- Different tools provide different views of the network
- Operators need to *run several different tools* so as to build up an *ongoing collection of traffic data*
  - for long-term traffic engineering and planning
  - for post-mortem analysis of network events

- *Thanks* to my colleagues at CAIDA and Auckland *!*