Kronosnet:
The new face of Corosync communications

Why we changed everything … AGAIN

Christine Caulfield
Principal Software Engineer
7th September 2017

Corosync Networking

How it currently works (layers on layers)

Totem PG (process groups) ↓

Totem Redundant Ring protocol ↓

Totem Single Ring protocol ↓

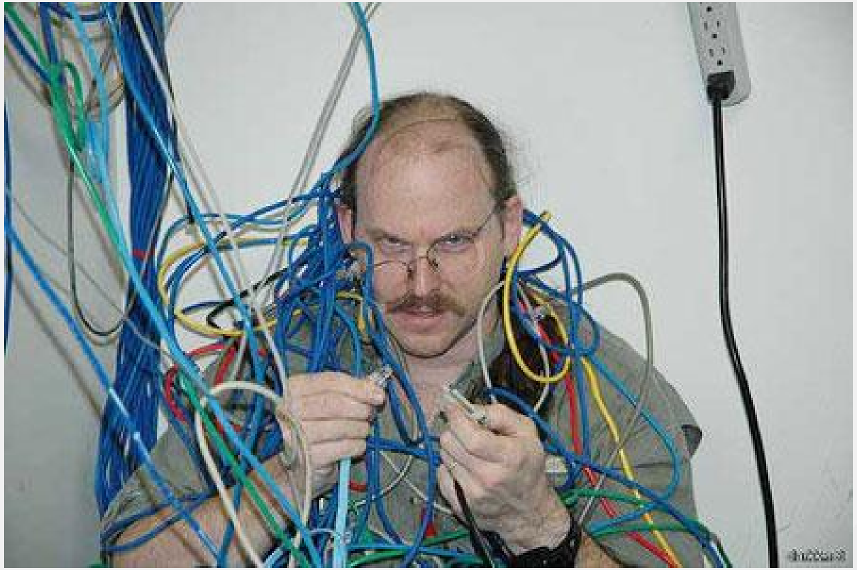
Totem networking ↓

Transports:

- UDP (multicast)
- UDPU (unicast)

There is a Totem Multi Ring protocol but I don't think anyone has ever been mad enough to implement it.

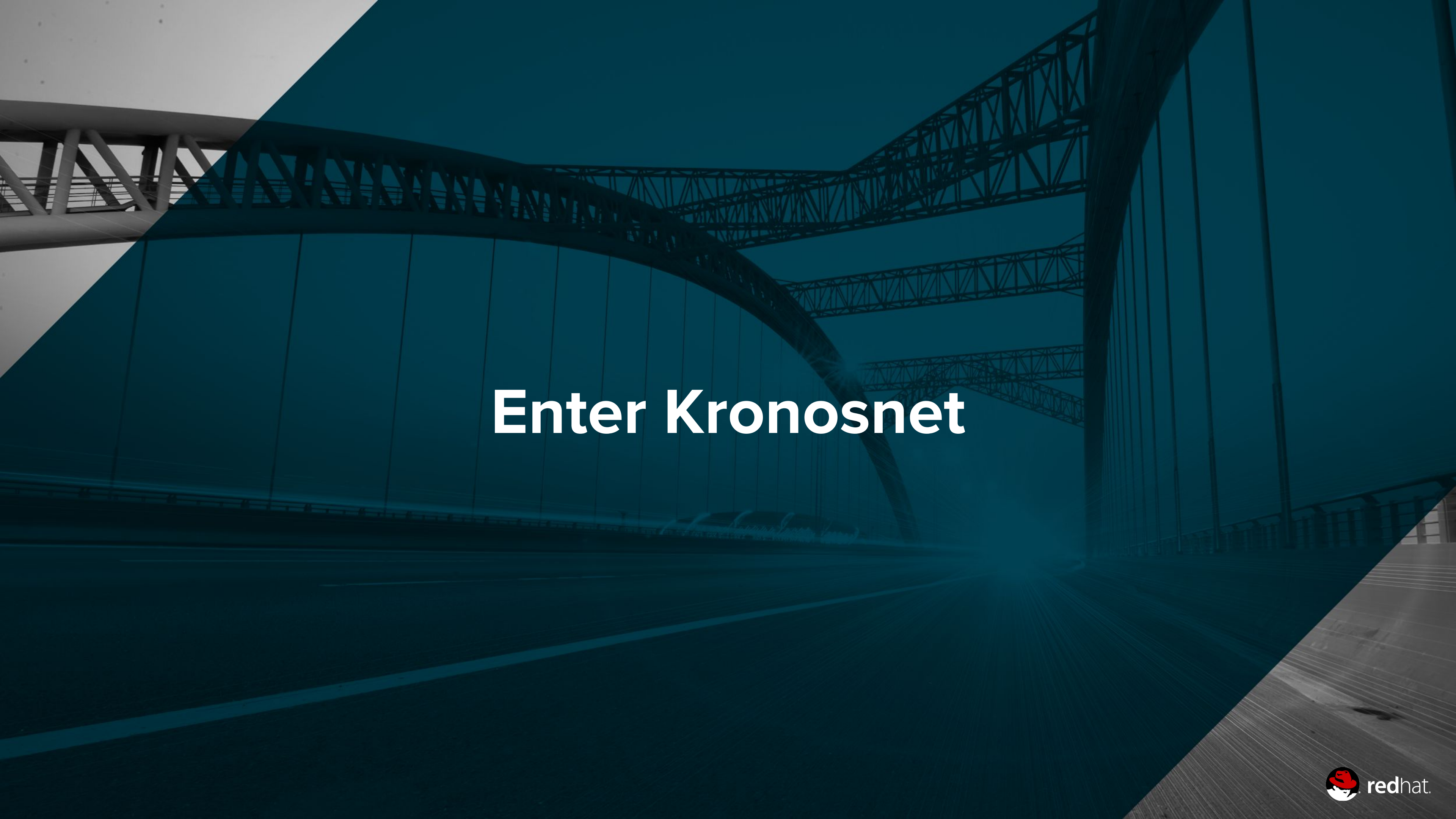
Though the layer is still there in the v2 code



Corosync Networking

What’s Broken?

- Multi-homing doesn’t work well
  - RRP is not always the answer
  - And only supports 2 rings
    - Compile-time constant, affects on-wire protocol
  - Not dynamic enough - fixed links
- 127.0.0.1 binding when an interface goes down
  - Major nuisance
  - Every week we have to tell someone not to do ‘ifdown’ in testing
- Over-sensitive to timeouts
- Big job to add or change low level protocols
- Hard-coded MTU per protocol



Enter Kronosnet

Kronosnet

Fabio’s “VPNs on Steroids”

- Pluggable protocols
- Multi-homing
- Multi-link
- Multi-protocol
- Dynamic MTUs
- Pluggable crypto & compression
- Extensive statistics

Corosync Networking

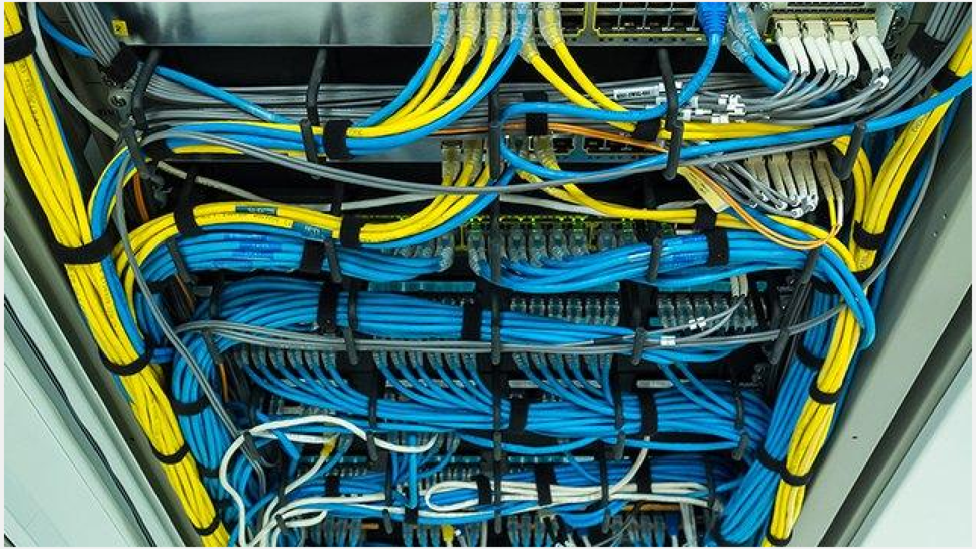
Slightly tidier

Totem PG (process groups) ↓

Totem Single Ring protocol ↓

Totem networking ↓

- Knet
- UDPU
- UDP



Fitting it into Corosync

Corosync over knet (1)

The new order

- Replaces the main protocols
  - transport=knet
  - udp & udpu are still there for backwards compatibility
- RRP is gone
  - Cleans up the code hugely
    - I deleted *loads* of code, and whole files.
  - But does mean udp/udpu are single-home only now
- No broadcast in knet
  - Though this could be added if *really* needed
- totemknet.c is a thin layer into the knet API
- This does NOT replace SRP (Single Ring Protocol)

Corosync over knet (2)

What it buys us

- Fixes the ifup/ifdown test!
  - For some people that would be enough
- Better performance and lower latencies
- Copes better with short network outages
- Multiple links between hosts (up to 8)
  - Can even have links with different protocols
  - active/passive - active/active
  - Link properties (priority for now)
- Delegates low level network details
  - Including pluggable crypto and compression
  - We might not use compression with corosync

Corosync over knet (3)

What it buys us

- More flexible MTU
  - Different ones per link
  - Dynamic
- Dynamic adding/removing of links
  - In progress for corosync
- More stats…
  - Lots more stats ………

Stats!

So many stats we had to invent a new ‘map’ for cmap

```
stats.knet.node1.link0.down_count (u32) = 12
stats.knet.node1.link0.latency_ave (u32) = 353
stats.knet.node1.link0.latency_max (u32) = 893
stats.knet.node1.link0.latency_min (u32) = 79
stats.knet.node1.link0.latency_samples (u32) = 6126
stats.knet.node1.link0.rx_data_bytes (u64) = 2007170
stats.knet.node1.link0.rx_data_packets (u64) = 11520
stats.knet.node1.link0.rx_ping_bytes (u64) = 159458
stats.knet.node1.link0.rx_ping_packets (u64) = 6133
stats.knet.node1.link0.rx_pmtu_bytes (u64) = 1748202
stats.knet.node1.link0.rx_pmtu_packets (u64) = 2526
stats.knet.node1.link0.rx_pong_bytes (u64) = 159276
stats.knet.node1.link0.rx_pong_packets (u64) = 6126
stats.knet.node1.link0.rx_total_bytes (u64) = 4074106
stats.knet.node1.link0.rx_total_packets (u64) = 26305
stats.knet.node1.link0.rx_total_retries (u64) = 4
stats.knet.node1.link0.tx_data_bytes (u64) = 4495908137
stats.knet.node1.link0.tx_data_errors (u32) = 0
stats.knet.node1.link0.tx_data_packets (u64) = 3131398
stats.knet.node1.link0.tx_data_retries (u32) = 4
stats.knet.node1.link0.tx_ping_bytes (u64) = 159562
stats.knet.node1.link0.tx_ping_errors (u32) = 0
stats.knet.node1.link0.tx_ping_packets (u64) = 6137
stats.knet.node1.link0.tx_ping_retries (u32) = 0
stats.knet.node1.link0.tx_pmtu_bytes (u64) = 1712337
stats.knet.node1.link0.tx_pmtu_errors (u32) = 0
stats.knet.node1.link0.tx_pmtu_packets (u64) = 1254
stats.knet.node1.link0.tx_pmtu_retries (u32) = 0
stats.knet.node1.link0.tx_pong_bytes (u64) = 159458
stats.knet.node1.link0.tx_pong_errors (u32) = 0
stats.knet.node1.link0.tx_pong_packets (u64) = 6133
stats.knet.node1.link0.tx_pong_retries (u32) = 0
stats.knet.node1.link0.tx_total_bytes (u64) = 4497939494
stats.knet.node1.link0.tx_total_errors (u64) = 0
stats.knet.node1.link0.tx_total_packets (u64) = 3144922
stats.knet.node1.link0.up_count (u32) = 12
stats.knet.node2.link0.down_count (u32) = 1
stats.knet.node2.link0.latency_ave (u32) = 57
stats.knet.node2.link0.latency_max (u32) = 663
stats.knet.node2.link0.latency_min (u32) = 14
stats.knet.node2.link0.latency_samples (u32) = 6137
stats.knet.node2.link0.rx_data_bytes (u64) = 578738691
stats.knet.node2.link0.rx_data_packets (u64) = 403085
stats.knet.node2.link0.rx_ping_bytes (u64) = 159562
stats.knet.node2.link0.rx_ping_packets (u64) = 6137
stats.knet.node2.link0.rx_pmtu_bytes (u64) = 161175715
stats.knet.node2.link0.rx_pmtu_packets (u64) = 5330
stats.knet.node2.link0.rx_pong_bytes (u64) = 159562
stats.knet.node2.link0.rx_pong_packets (u64) = 6137
stats.knet.node2.link0.rx_total_bytes (u64) = 740233530
stats.knet.node2.link0.rx_total_packets (u64) = 420689
stats.knet.node2.link0.rx_total_retries (u64) = 0
stats.knet.node2.link0.tx_data_bytes (u64) = 4495908137
stats.knet.node2.link0.tx_data_errors (u32) = 0
stats.knet.node2.link0.tx_data_packets (u64) = 3131398
stats.knet.node2.link0.tx_data_retries (u32) = 0
stats.knet.node2.link0.tx_ping_bytes (u64) = 159562
stats.knet.node2.link0.tx_ping_errors (u32) = 0
stats.knet.node2.link0.tx_ping_packets (u64) = 6137
stats.knet.node2.link0.tx_ping_retries (u32) = 0
stats.knet.node2.link0.tx_pmtu_bytes (u64) = 161151730
stats.knet.node2.link0.tx_pmtu_errors (u32) = 0
stats.knet.node2.link0.tx_pmtu_packets (u64) = 2665
stats.knet.node2.link0.tx_pmtu_retries (u32) = 0
stats.knet.node2.link0.tx_pong_bytes (u64) = 159562
stats.knet.node2.link0.tx_pong_errors (u32) = 0
```

etc...

Configuring it

Not much has changed - apart from the details

- Nodelist is now compulsory
- Most corosync.conf params still there
- Knet links defined in interface{} stanza
- So each link can have different params
  - Or even a different transport
- Links can have priorities assigned
  - For ‘passive’ mode
  - Lowest priority is used if available
  - otherwise use link number

```
totem {
    … the usual stuff ...
    transport: knet
    interface {
        linknumber: 0
        knet_transport: udp
        Knet_link_priority: 2
        knet_ping_timeout: 2500
    }
}
```

Transports

What do you mean ‘transports’ I thought this was Kronosnet?

- Knet uses IP protocols underneath
  - You probably guessed that
  - But no reason why they should be
  - No, I am not doing a DECnet transport
    - But technically it’s feasible

- Currently supported
  - UDP (unicast)
  - SCTP (connection-oriented)
  - Loopback (for localhost only … obviously)
  - No multicast, but could be added if really wanted
  - No broadcast
    - We are no longer *that* insane

Do you know how hard it is to find decent multi-transport pictures that are free to use? © Virgin Trains East Coast

So, about that ‘stats’ map?

Splitting the maps

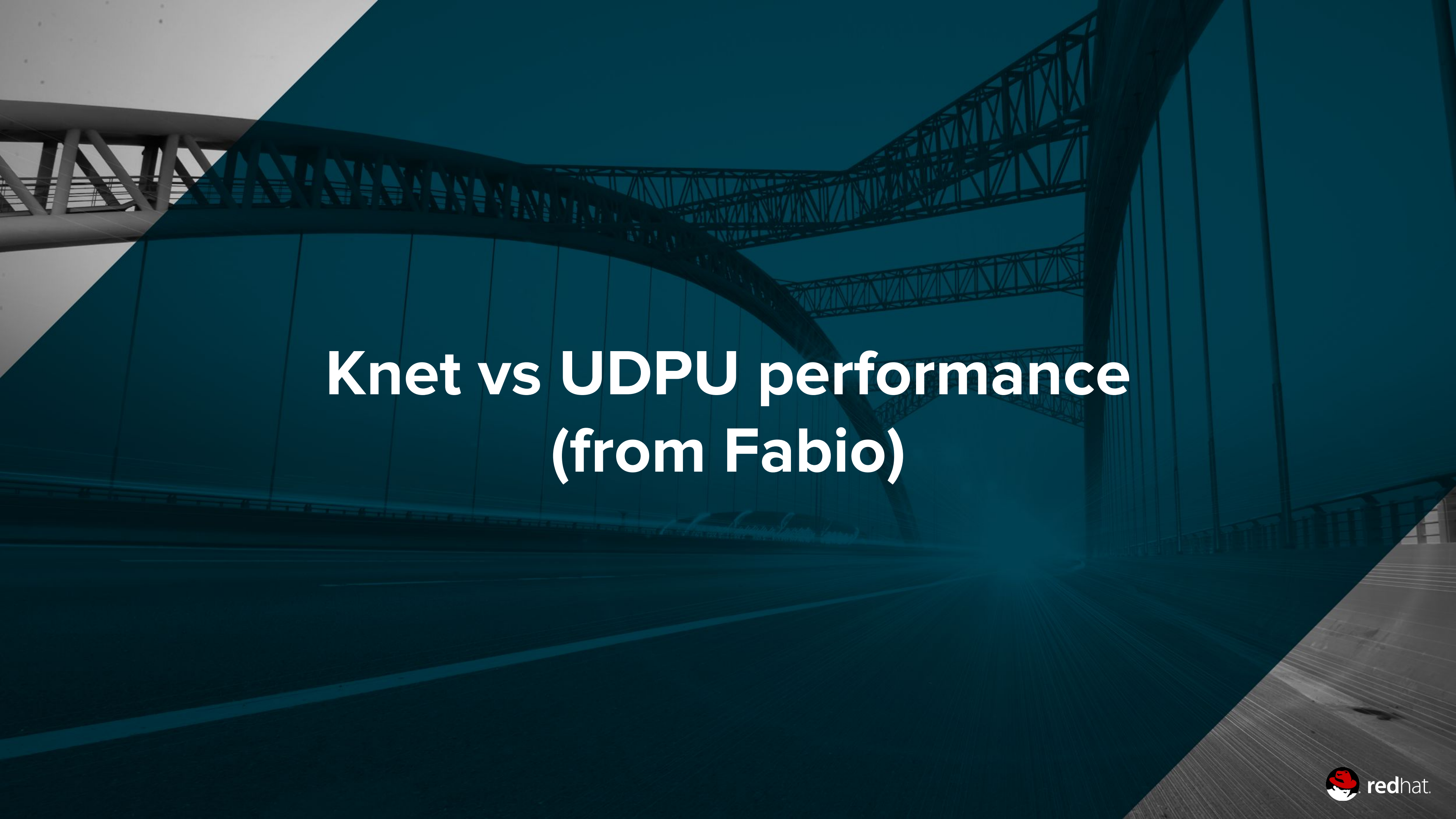
It’s all just toooooo much

- icmap is now just one ‘map’ accessed from cmap (corosync-cmapctl)
  - Now it just contains configuration information and some state
- ‘stats’ is the other
  - Fairly easy to add new ones if necessary too
- This means we don’t have to store the stats twice in memory
- And every time we retrieve them they are up-to-date
- corosync-cmapctl -mstats
- You can’t strictly use trackers on the stats numbers
  - It works but uses a timer
  - But you can track add & delete for new knet links & ipc connections

Available now

Seems reliable, but don’t use in production

- https://github.com/fabbione/kronosnet
  - master is the latest, soon to be 1.0
  - Not currently packaged - volunteers?
- https://github.com/corosync/corosync
  - master will eventually be 3.0
  - When I get dynamic links working

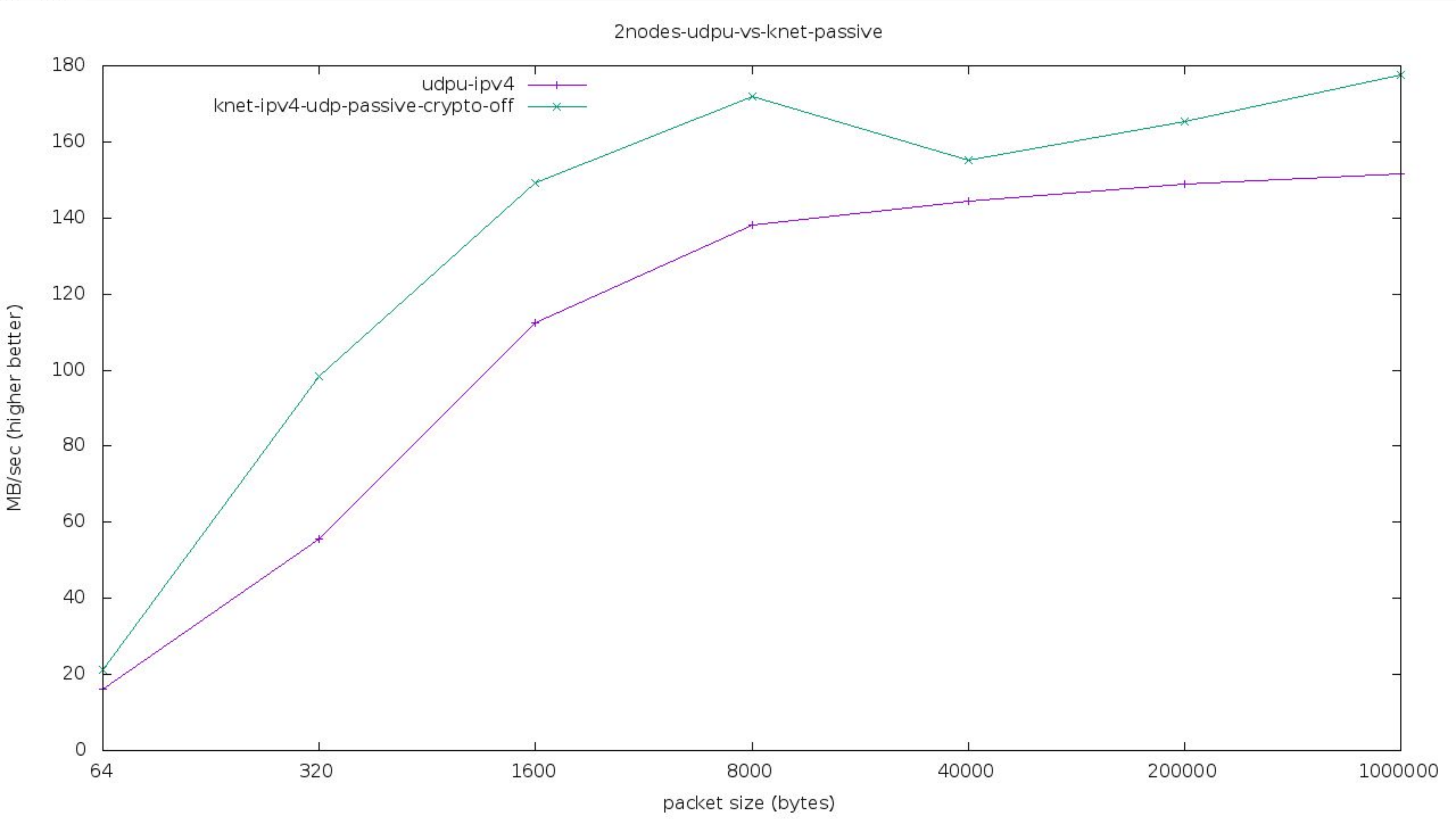


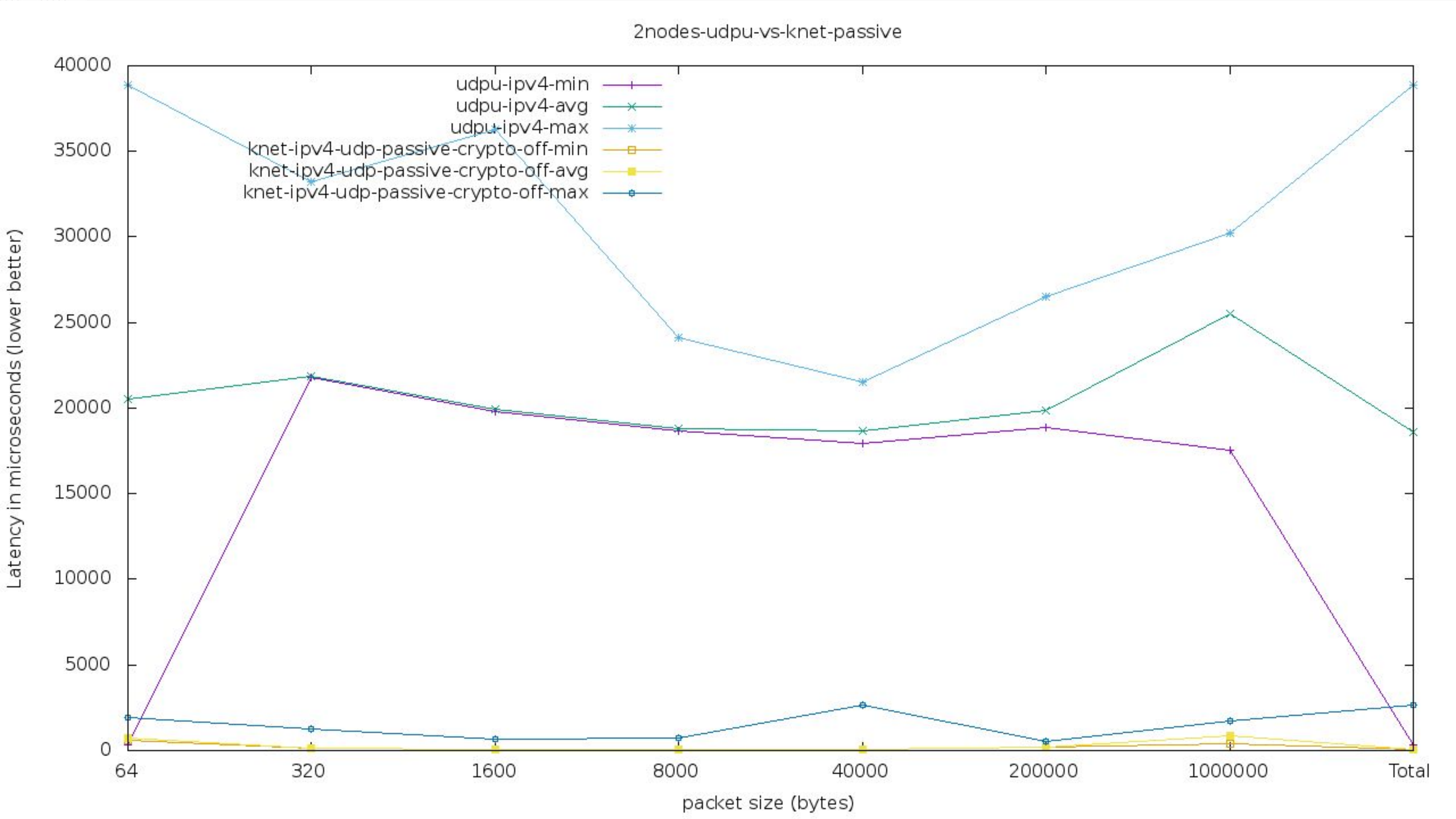
Knet vs UDPU performance (from Fabio)

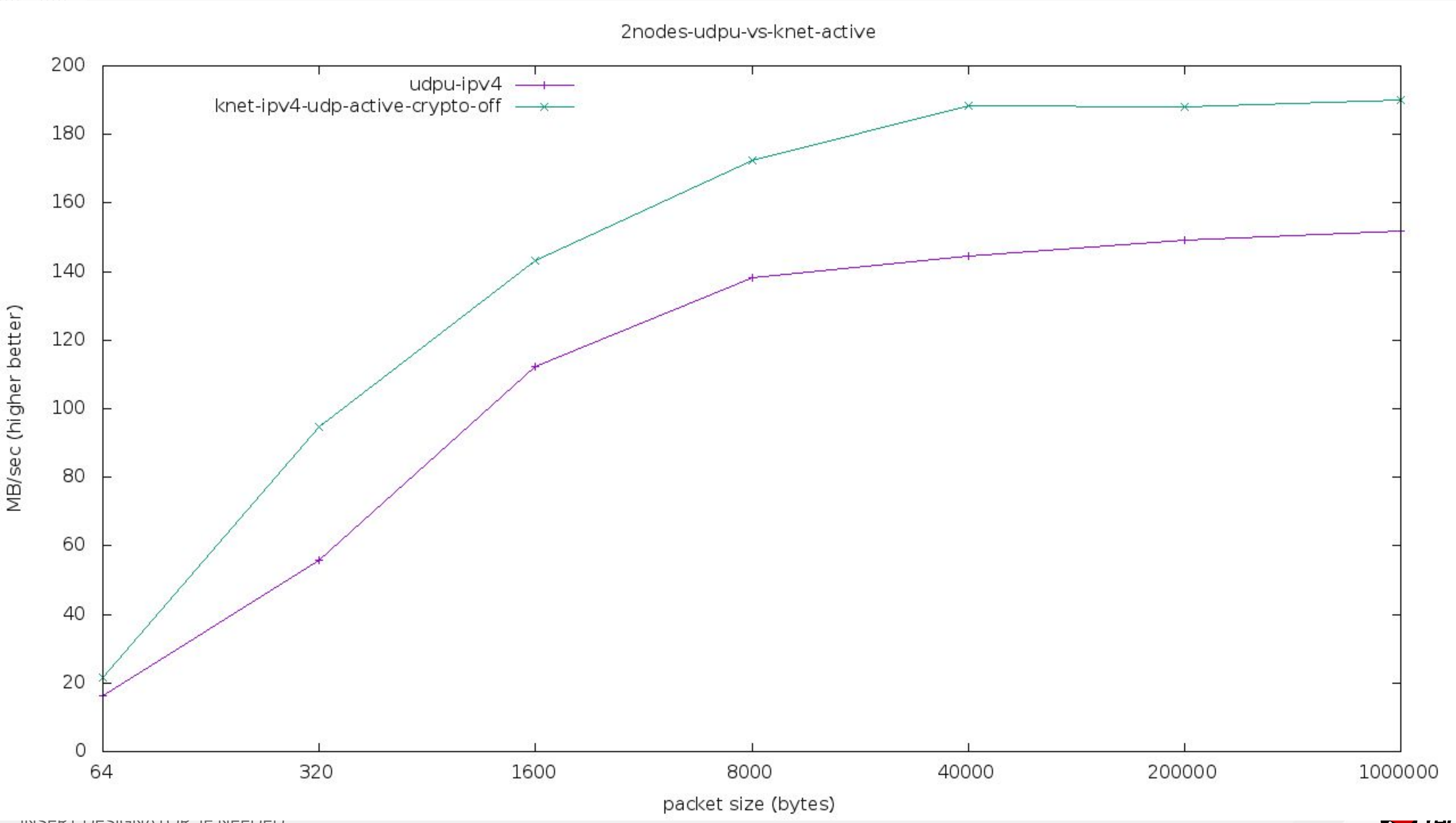
Testing setup

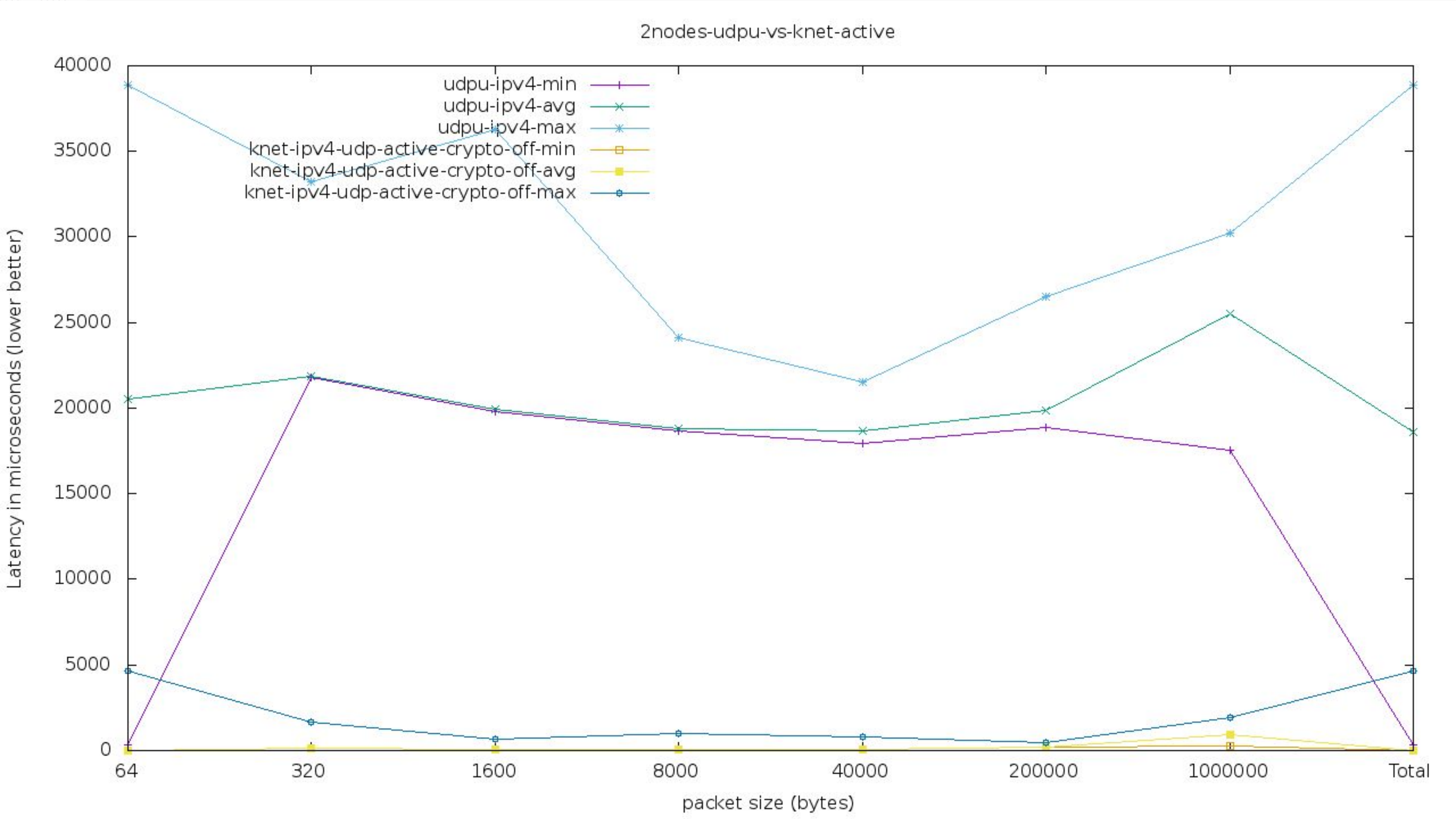
Don´t look at the absolute numbers, look at the graphs!

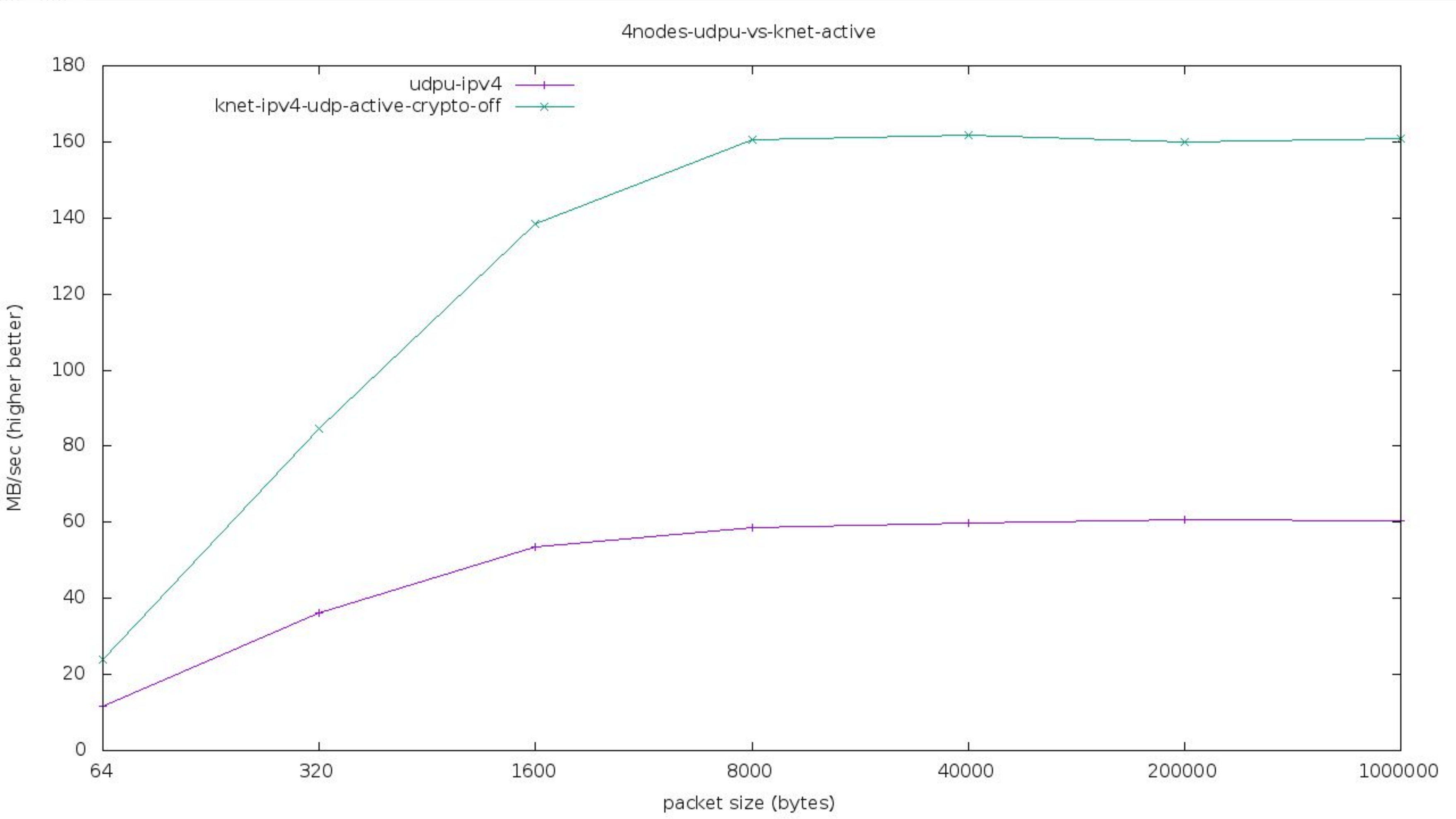
- RHEL7.4 + updates
- Only BaseOS installed, no tuning or optimizations, everything is default.
- 2 to 4 nodes
- 2x40 Gbic networks
- Latest libqb/libknet/corosync
- Data collected via knet-ansible-ci

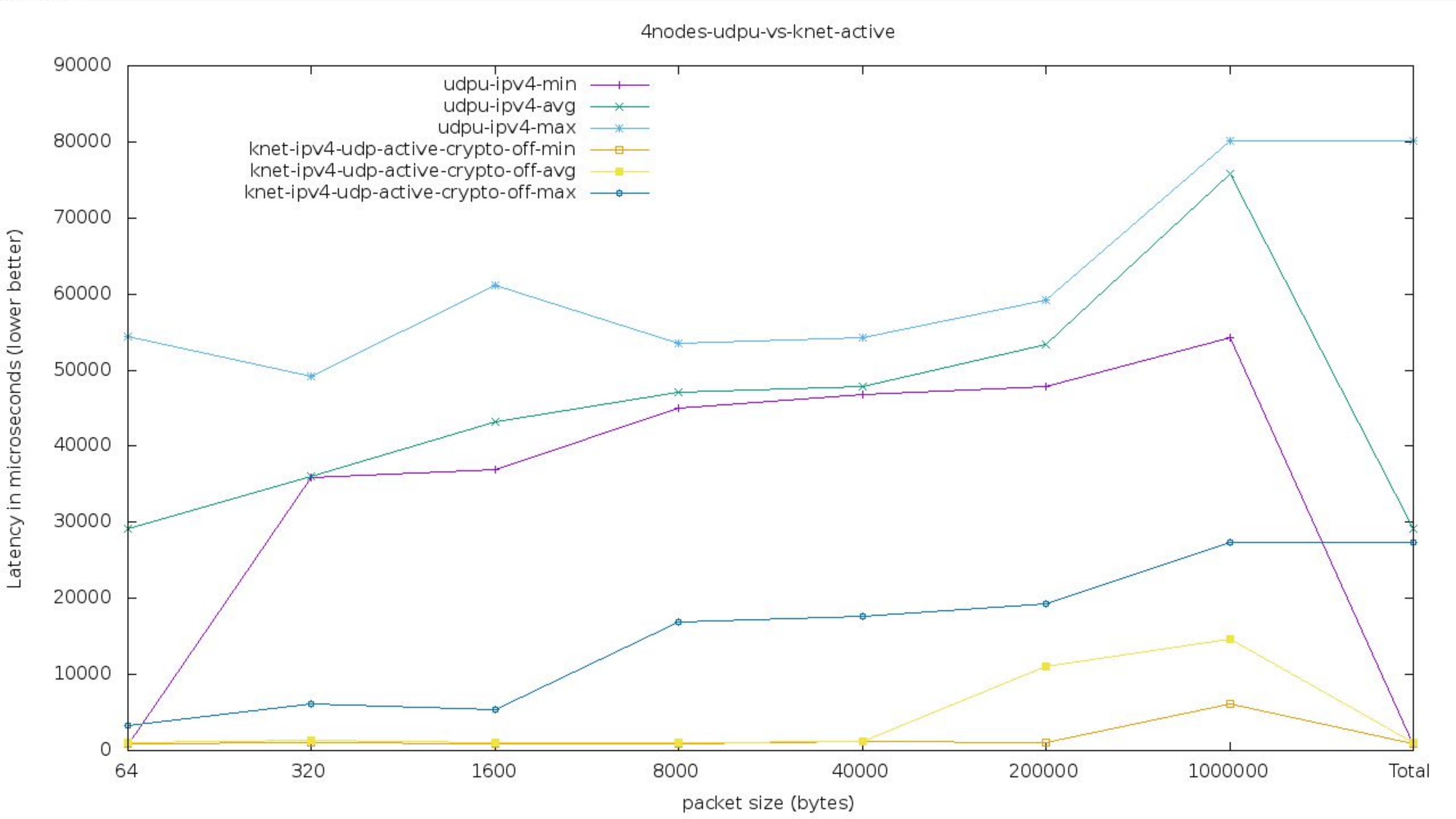

2nodes-udpu-vs-knet-passive
udpu-ipv4
knet-ipv4-udp-passive-crypto-off
MB/sec (higher better)
180
160
140
120
100
80
60
40
20
0
64
320
1600
8000
40000
200000
1000000
packet size (bytes)

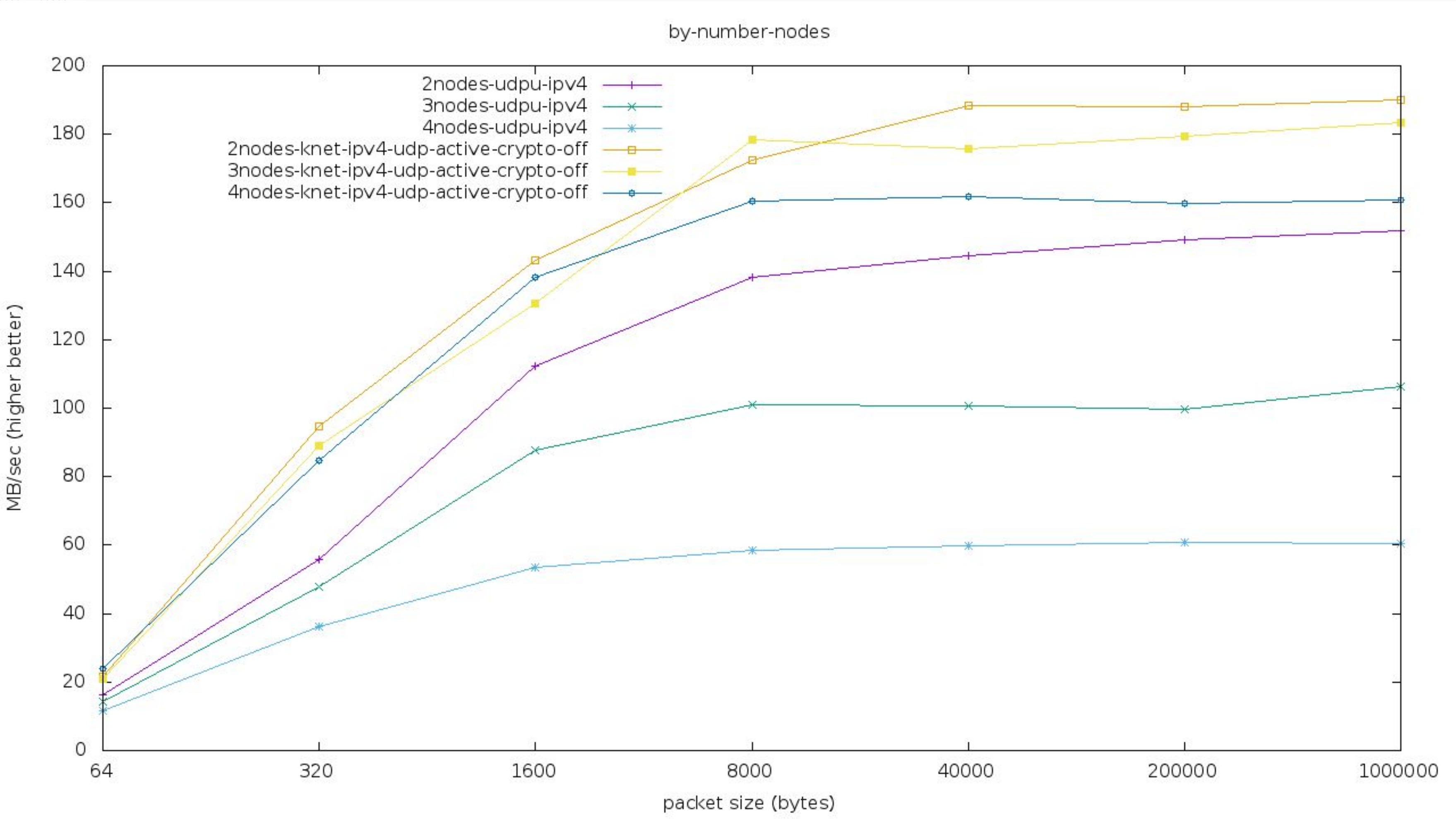

2nodes-udpu-vs-knet-passive
udpu-ipv4-min
udpu-ipv4-avg
udpu-ipv4-max
knet-ipv4-udp-passive-crypto-off-min
knet-ipv4-udp-passive-crypto-off-avg
knet-ipv4-udp-passive-crypto-off-max
Latency in microseconds (lower better)
40000
35000
30000
25000
20000
15000
10000
5000
0
64
320
1600
8000
40000
200000
1000000
Total
packet size (bytes)

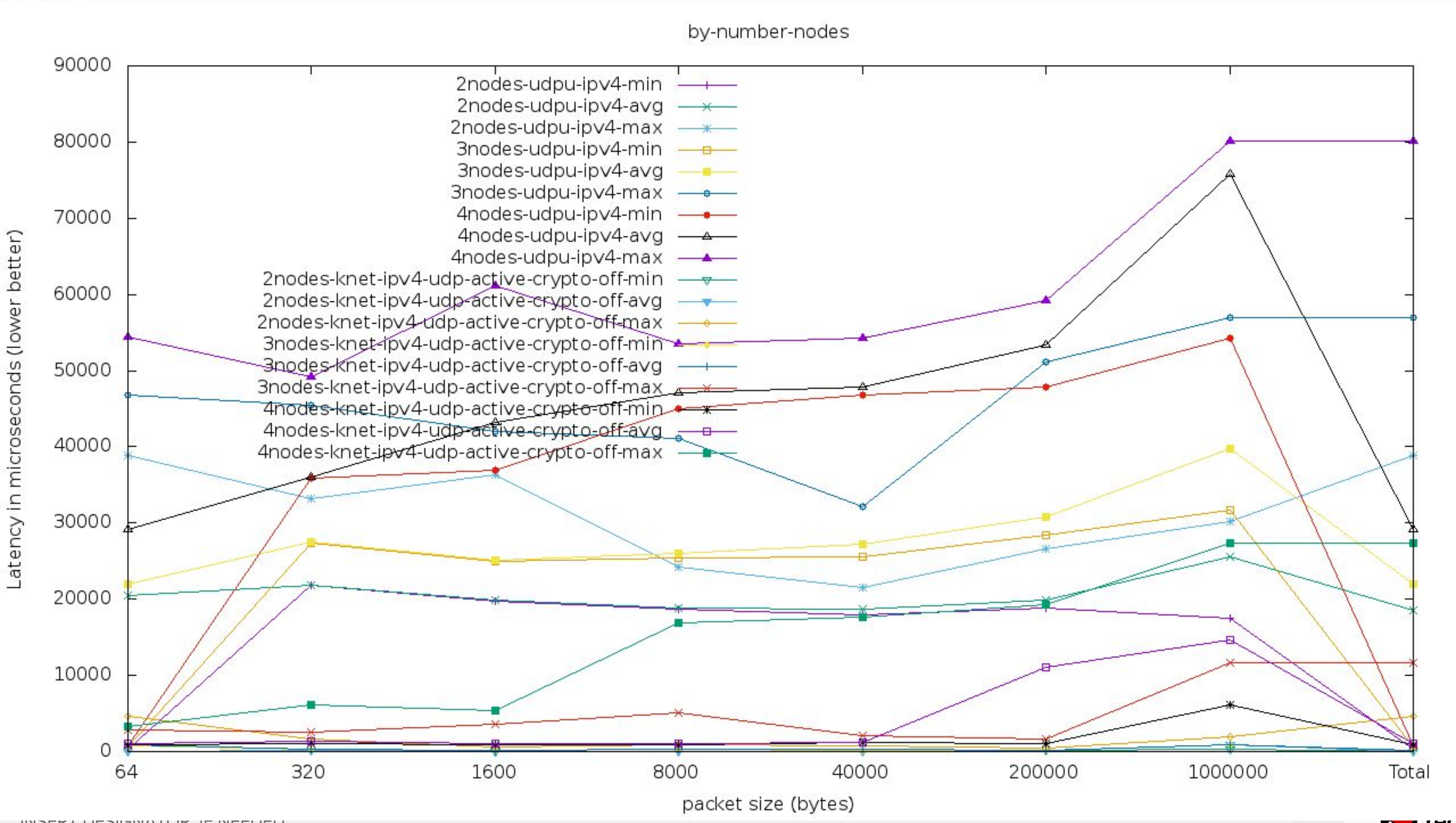

2nodes-udpu-vs-knet-active
udpu-ipv4
knet-ipv4-udp-active-crypto-off
MB/sec (higher better)
0
20
40
60
80
100
120
140
160
180
200
64
320
1600
8000
40000
200000
1000000
packet size (bytes)


2nodes-udpu-vs-knet-active
udpu-ipv4-min
udpu-ipv4-avg
udpu-ipv4-max
knet-ipv4-udp-active-crypto-off-min
knet-ipv4-udp-active-crypto-off-avg
knet-ipv4-udp-active-crypto-off-max
Latency in microseconds (lower better)
40000
35000
30000
25000
20000
15000
10000
5000
0
64
320
1600
8000
40000
200000
1000000
Total
packet size (bytes)


4nodes-udpu-vs-knet-active
udpu-ipv4
knet-ipv4-udp-active-crypto-off
MB/sec (higher better)
180
160
140
120
100
80
60
40
20
0
64
320
1600
8000
40000
200000
1000000
packet size (bytes)


4nodes-udpu-vs-knet-active
udpu-ipv4-min
udpu-ipv4-avg
udpu-ipv4-max
knet-ipv4-udp-active-crypto-off-min
knet-ipv4-udp-active-crypto-off-avg
knet-ipv4-udp-active-crypto-off-max
Latency in microseconds (lower better)
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
64
320
1600
8000
40000
200000
1000000
Total
packet size (bytes)


by-number-nodes
2nodes-udpu-ipv4
3nodes-udpu-ipv4
4nodes-udpu-ipv4
2nodes-knet-ipv4-udp-active-crypto-off
3nodes-knet-ipv4-udp-active-crypto-off
4nodes-knet-ipv4-udp-active-crypto-off
MB/sec (higher better)
packet size (bytes)
0
20
40
60
80
100
120
140
160
180
200
64
320
1600
8000
40000
200000
1000000


by-number-nodes
Latency in microseconds (lower better)
packet size (bytes)
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
64
320
1600
8000
40000
200000
1000000
Total
2nodes-udpu-ipv4-min
2nodes-udpu-ipv4-avg
2nodes-udpu-ipv4-max
3nodes-udpu-ipv4-min
3nodes-udpu-ipv4-avg
3nodes-udpu-ipv4-max
4nodes-udpu-ipv4-min
4nodes-udpu-ipv4-avg
4nodes-udpu-ipv4-max
2nodes-knet-ipv4-udp-active-crypto-off-min
2nodes-knet-ipv4-udp-active-crypto-off-avg
2nodes-knet-ipv4-udp-active-crypto-off-max
3nodes-knet-ipv4-udp-active-crypto-off-min
3nodes-knet-ipv4-udp-active-crypto-off-avg
3nodes-knet-ipv4-udp-active-crypto-off-max
4nodes-knet-ipv4-udp-active-crypto-off-min
4nodes-knet-ipv4-udp-active-crypto-off-avg
4nodes-knet-ipv4-udp-active-crypto-off-max