Ruru: Real-Time Wide-Area TCP Latency Monitoring
About Netlab

- University of Glasgow, United Kingdom
  - Fourth oldest university in the English-speaking world and one of Scotland's four ancient universities. Founded in 1451.

- Networked System Research Laboratory “Netlab”, School of Computing Science
  - Website: https://netlab.dcs.gla.ac.uk
  - Team: 3 academics, 4 researchers, 7 PhD students
  - Director: Dr. Dimitrios P Pezaros

- Research on SDN, NFV, mobile edge, network security and data plane programmability, resilient infrastructure …

- Project partners include: BT, Google, airbnb, AIRBUS, Microsoft, ARM
• New Zealand’s NREN
• Connecting universities and research labs
• International links to Sydney, Los Angeles
• Based in Wellington, NZ
Ruru (morepork):
A native New Zealand bird

“a watchful guardian”
Motivation

As a network operator, the goal is to understand the performance of the network we provide for our customers.

• Most of today’s network monitoring tools are either
  • Too coarse-grained (e.g., port statistics collected every 5 minutes)
  • Or rely on synthetic, generated traffic (e.g., PerfSonar)

• Individual user-perceived performance (especially real-time end-to-end latency) has not been monitored yet
  • No easy-to-use, free tools were available
  • Techniques were too slow, constrained, proprietary
  • Would require special hardware – expensive, not customized
  • Results were not visual / analyzed
Why end-to-end latency?

- Increasing number of real-time applications (e.g., online games using virtual reality, multi-site financial transaction processing, etc.)
- 5G mobile architecture use-cases (e.g., robotics, tactile Internet) require interactive back-and-forth communication
- New Zealand’s isolated geographical location

As a result, user-perceived end-to-end latency is becoming an all-important factor for both users and network providers
What is Ruru?

- Ruru is a measurement pipeline that runs on a commodity server
- Ruru measures **actual, accurate** end-to-end (e.g., device-to-server) latency in real-time and maps it to geo-locations
  - What we see is exactly what a user experiences
- Ruru visualizes measurements in **real-time on a world map**
- It is using today’s cutting-edge, **open-source** technologies
  - Intel DPDK – high speed packet processing
  - Zero MQ – zero copy socket communication
  - Influx DB – time series data storage
  - WebGL – high-performance 3D graphics library in a web browser
Measuring end-to-end latency

- **Round-trip time (RTT)**
  - In telecommunications, the round-trip delay time (RTD) or round-trip time (RTT) is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received.

- **TCP only**
  - Web browsing, e-mail, chat, etc.
  - But usually not media

- **IPv4 only for now**
  - Geolocation is only available for IPv4

- **RTT guidelines**
  - NZ to South Africa: 500ms
  - NZ to US: 130ms
Architecture (high level)

10Gbit/s traffic
Passive optical fiber tap

Per-flow latency measurement
Analytics, geo-mapping, filtering
Database
Frontends (Ruru map, Grafana)

Ruru Host (Dell server with Intel x520 10Gbit/s network card)
Challenge 1/3

How do we process 10Gbit/s international backbone traffic per flow in real-time?
Linux kernel performance

Goal: to process 10Gbit/s traffic in real time

Can’t we use just libraries such as Scapy or tcpdump?

Source: Intel DPDK overview
Some calculations

- Why do we need to bypass the kernel?
  - Minimum Ethernet packet: 64 bytes + 20 preamble
  - Maximum number of pps at 10Gbit/s: 14 880 952 \((10^{10}/84 \text{ bytes}*8)\)
  - Time to process a single packet: 67.2 ns
  - CPU cycles required on a 3Ghz CPU: 201 cycles \((1 \text{ GHz} \rightarrow 1 \text{ cycle/ns})\)

<table>
<thead>
<tr>
<th>Packet size</th>
<th>1024 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets / sec</td>
<td>1.2 million</td>
</tr>
<tr>
<td>Arrival rate</td>
<td>835 ns</td>
</tr>
<tr>
<td>2 GHz</td>
<td>1620 cycles</td>
</tr>
<tr>
<td>3 GHz</td>
<td>2505 cycles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet size</th>
<th>64 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets / sec</td>
<td>14.8 million</td>
</tr>
<tr>
<td>Arrival rate</td>
<td>67.2 ns</td>
</tr>
<tr>
<td>2 GHz</td>
<td>135 cycles</td>
</tr>
<tr>
<td>3 GHz</td>
<td>201 cycles</td>
</tr>
</tbody>
</table>
Some calculations…

- Why do we need to bypass the kernel?
  - L3 cache hit: 40 cycles
  - L3 cache miss: 200 cycles (no budget for this with 64 byte packets)

<table>
<thead>
<tr>
<th>Packet size</th>
<th>1024 bytes</th>
<th>64 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets / sec</td>
<td>1.2 million</td>
<td>14.8 million</td>
</tr>
<tr>
<td>Arrival rate</td>
<td>835 ns</td>
<td>67.2 ns</td>
</tr>
<tr>
<td>2 GHz</td>
<td>1620 cycles</td>
<td>135 cycles</td>
</tr>
<tr>
<td>3 GHz</td>
<td>2505 cycles</td>
<td>201 cycles</td>
</tr>
</tbody>
</table>
DPDK overview

- Set of libraries and drivers for fast packet processing
- Runs on any processors, mostly in Linux userland
- Main libraries:
  - multicore framework
  - huge page memory
  - ring buffers
  - poll-mode drivers for networking, crypto and eventdev
- These libraries can be used to:
  - receive and send packets within the minimum number of CPU cycles (usually less than 80 cycles)
  - develop fast packet capture algorithms (tcpdump-like)
  - run third-party fast path stacks
Why did I choose DPDK?

• Direct access to the hardware through userspace
• Open Source API (BSD license)
• Many NICs support it (not just Intel)
• Isolation / security (won’t cause segfault on the machine)
• Proven high performance (20M 64 bytes pps with native DPDK)
Challenge 2/3

How do we geographically map 10Gbit/s international backbone traffic per flow in real-time?
Ruru Analytics

- We need high-performance way of:
  - Mapping to geo-locations
    - Most high-level libraries (e.g., GeoIP) will suffer as they make a REST (very slow) calls for every lookup
  - Filtering results from pipeline

- Solution: multi-threaded C program with offline geo-database and cache
  - I used IP2Location databases (99.5% accuracy)
  - I do ASN and geo lookups for each source and destination IP
  - A small 5000 size cache for the lookups helps a lot
Challenge 3/3

How do we **visualize** 10Gbit/s international backbone traffic per flow *in real-time*?
Map frontend

- Goal: to visualize multiple thousand connections per second
  - High-level libraries can not visualize in this rate (e.g., d3)

- Solution: to use WebGL-powered 3D visualization

- I am using libraries, such as:
  - Deck-gl: rendering stacks of visual overlays over map
  - Luma-gl: WebGL library
  - React-map-gl: Mapbox for the actual map (separate API key is required)

- As a result, we can run the visualization with the speed of 21 fps (using Safari on a 2016 Macbook)
Challenges solved

1. Measuring end-to-end latency
2. Geo-mapping measurements
3. Visualizing traffic

... in 10Gbit/s ...
Architecture (all pieces together)

Hash table
key: hash of the flow
value: timestamps

source IP: ‘1.2.3.4’
destination IP: ‘4.3.2.1’,
external latency: ‘123 ms’,
internal latency: ‘13 ms’

source & dest geolocation
source & dest ASN / AS
source & dest city / country
external and internal latencies

Ruru DPDK packet analysis
using partial parsing of TCP packets
tracks: SYN, SYN-ACK, 1st ACK of each flow

Ruru DPDK packet analysis
using partial parsing of TCP packets
tracks: SYN, SYN-ACK, 1st ACK of each flow

CPU core 1
Queue 1
Queue 2
Queue n
Intel x520 NIC
Symmetric RSS
tap (TX/ RX)

InfluxDB
stats (JSON API)

IP -> geo location lookup
IP -> ASN lookup

zmq
(websocket)

from country: NZ
to city: Los Angeles, US
when: last 10 minutes
average total latency: 140 ms

Ruru Analytics

Live traffic (Auckland - Los Angeles 10Gbit/s)

Grafana UI (live stats)
Frontends

Ruru real-time WebGL live 3D map
(1000s lines/sec)

In: ACM SIGCOMM 2017, Los Angeles, CA, USA; 21-25 Aug 2017
Applications of Ruru

- Fault localization for wide area networks
  - “Immediate notice if latency has started to increase to Facebook’s AS”
  - “Some of our users are getting higher latency compared to others”

- Fault localization in your internal network
  - “A set of our users are getting higher internal latency than others”
    - Could be router / switch issue for those clients
    - Ruru shows that e.g., wireless clients usually get higher latency

- Network planning / auditing
  - Ruru shows where user’s connections are going the most and what latency they are experiencing
Ruru live deployment between Los Angeles and Auckland
Ruru in production

• Using Ruru we found latency related issues, such as:
  1. The “00:48 bug”
     • Increased latency to 4 sec at 00:48 every night for just a few connections
     • Turned out there was a firewall update that time
  2. Software switch issue has also been noticed
     • Only wireless clients were affected

• Offline analysis has also been conducted
  • We can identify CDNs easily (providing very low latency)
  • Clients usually start 5-6 flows to the same destination at once
  • Seasonality is clearly visible (latency increases during the day when the network is utilized)
  • ... (more to come)
Live demo
More on Ruru

- Paper: Cziva, Richard (University of Glasgow), Lorier, Chris (REANNZ) and Pezaros, D. Pezaros (University of Glasgow)
  In: ACM SIGCOMM 2017, Los Angeles, CA, USA, 21-25 Aug 2017
    - Winner of the ACM SRC competition

- Github: [https://github.com/REANNZ/ruru](https://github.com/REANNZ/ruru)

Project page: [https://netlab.dcs.gla.ac.uk/projects/ruru-latency-visualisation](https://netlab.dcs.gla.ac.uk/projects/ruru-latency-visualisation)
Conclusions

• I believe user-perceived end-to-end latency is an important metric

• To do measurements and visualization on this scale, the go-to tools (e.g., Scapy, Geolp, d3.js) are insufficient
  • Careful engineering, parallel processing and low level tools are required

• I have created Ruru, an open-source pipeline that can is able to measure, analyze and visualize user perceived TCP latency on 10Gbit/s live traffic

*Interested in deploying Ruru? – let’s talk!*
Thank you for your attention! Questions?

Richard Cziva | University of Glasgow | Richard.Cziva@glasgow.ac.uk | RIPE 75 | 22 October 2017, Dubai, UAE