DON’T SHARE, DON’T LOCK: LARGE-SCALE SOFTWARE CONNECTION TRACKING WITH KRONONAT

NICOLAS LE SCOUARNEC
JOINT WORK WITH FABIEN ANDRÉ,
STÉPHANE GOUACHE, ANTOINE MONSIFROT
Typical internet access network

Access Network

Internet

Customer Premise Equipment (Residential Gateway)
vCPE Rationale

Simplify software running on millions of embedded devices
► Easier upgrades
► Better integration

Provide visibility into home network
► Secure IoT
► Remote troubleshooting
Building middleboxes for residential networks

Customer Premise Equipment (Residential Gateway)

Access Network

Middlebox (e.g., NAT and Firewall)

Internet
What (not) to use?

NFV approach (virtualized appliances)
- One VM/container per customer
- Running existing software (e.g., OpenWRT or Linux)
- As done for example in R-CORD

Virtual Switches for traffic dispatching to VM

Does not scale to millions of VMs/containers
Not cost effective
## Which equipment to use?

<table>
<thead>
<tr>
<th></th>
<th>Vendor B (HW Appliance)</th>
<th>Vendor C (HW Appliance)</th>
<th>2x Xeon E5 v4 (40 cores)</th>
<th>Vendor A (VM)</th>
<th>StatelessNF (NSDI’17)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L4 Throughput</strong></td>
<td>58 Mpps</td>
<td>63 Mpps</td>
<td>180 Mpps</td>
<td>4.5 Mpps</td>
<td>4 Mpps</td>
</tr>
<tr>
<td><strong>(Simple IMIX)</strong></td>
<td>130 Gbps</td>
<td>140 Gbps</td>
<td>400 Gbps</td>
<td>10 Gbps</td>
<td>10 Gbps</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>65 K$ (HW+SW)</td>
<td>200 K$ (HW+SW)</td>
<td>30 K$ (HW)</td>
<td>21 K$ (SW)</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Redundancy model</strong></td>
<td>1+1</td>
<td>1+1</td>
<td>N+1</td>
<td>1+1</td>
<td>N+1</td>
</tr>
</tbody>
</table>

**Objective:**
- 180 Mpps / server
- 4.5 Mpps / core

**Available SW for running on COTS server**
The residential vCPE challenge

Build a middlebox (firewall, NAT, …) for residential networks from COTS hardware

Efficient, Reliable, Scalable
L4 connection tracking
For millions of users
Best practices for high-performance networking software

Avoid context switches
► Use kernel-bypass systems (e.g., DPDK)

Don’t lock, don’t share
► Cross-core sharing is expensive even without explicit locking

Run-to-completion model
► Receive, process, transmit, without buffering nor blocking

Applying all these principles everywhere is non-trivial
Reliable - sharding and replication

Provide reliability by design, not as an afterthought

Assign both a master server and a slave server to each shard

Replicate state from master to slave for each shard
Replication - Availability rather than Consistency

No external DB
- Faster insertion and lookup rate (450M lookups/second on 18 cores)
- Non-blocking (no remote memory access)

Availability rather than consistency
- Networks are unreliable, applications will recover
- Yet, even short unavailabilities are noticed by user
- Master does not wait for acknowledgment from slave

Efficient lock-less replication
- Batching for improved performance
- **Same thread for packet processing and replication**
- Traffic not interrupted during slave initialization, using support from hash table
Efficient (I) – Sharding to the core

Shard 1
Shard 2
Shard 3
Shard 4
Shard 5
**Enforce share-nothing** by binding each shard exclusively to a single CPU core. All packet processing & management done by the corresponding thread.
Efficient (II) - Expose each core to the network

Expose an independent identity for each core (not server nor NIC) on the network

One single mechanism to address between and within servers

Each core appears in the system as an independent router
Efficient (II) - Scalable load-balancing by NICs and Switches

Leverage existing top-of-rack switches and server-class NIC to entirely offload load-balancing. *Physical L3 Switches are much more efficient than virtual switches.*
IP routing allows precise control on reverse path and also failover path. Traffic is highly asymmetrical, use VLAN to improve hardware usage. IP routing allows more control than RSS or ECMP based distribution.
Our design: benefits

Distribution across servers and across cores identical

► Simplified implementation
► Performance scale linearly across cores and across servers

Dynamic load-balancing included (dynamic routing + replication)

► Re-balance the load between servers
► Scale-out and in as demand evolve: elasticity

Daily Internet Traffic

Unused resources
(75% potential savings energy, cooling, ...)

00:00  03:00  06:00  12:00  15:00  18:00  21:00  00:00
Benchmarking

Multi-core, multi-server benchmarking tool following the same principles

Traffic generator

System under test
(large-scale and multi-server)
Multi-core, multi-server benchmarking tool following the same principles

System under test
(large-scale and multi-server)
Performance

Performance (12 cores) for established connections

<table>
<thead>
<tr>
<th></th>
<th>Mpps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>0</td>
</tr>
<tr>
<td>Krononat</td>
<td>100</td>
</tr>
</tbody>
</table>

Linux (e.g., R-CORD)
Performance for established connections

**Objective:**
4.5 Mpps/core
Availability - Server departure

Less than 600 ms → below network timeouts
Conclusions

Resilient distributed middlebox using **COTS hardware**

- **77 million packets per second on only 12 cores**
  - 6.4 Mpps/core above objective (4.5 Mpps/core)
- Recover from failures automatically without users noticing
- **Cost-effective N+1 redundancy**
- Redundancy and dynamic load-balancing allow **elasticity**

**Re-usable design**

- Expose each core as a distinct entity to the network
- Push per-core traffic steering to the networking equipments (NIC, switches)
- Applied to multi-server multi-core benchmarking tool
References

Don’t share, Don’t lock: Large-scale Software Connection Tracking with Krononat
*Fabien André, Stéphane Gouache, Nicolas Le Scouarnec and Antoine Monsifrot*
USENIX ATC’18

Cuckoo++ Hash Tables: High-Performance Hash Tables for Networking Applications
*Nicolas Le Scouarnec*
ACM/IEEE ANCS’18
APPENDIX
Building a distributed software CG-NAT/FW/…

Access network

- Bi-directional traffic
- Must filter unknown connections

L4 Load-balancers

- Maglev
- Ananta
- Fastly@NSDI
- SilkRoad
- …

- No-reverse path traffic (DSR)
- Leverage deterministic hashing
Availability: Graceful departure

Service interruption duration (ms)

Occurrences

- Load rebalancing
- Failure detection and recovery
Availability: Hard Failure

Less than 7s → below many network timeouts

Load rebalancing

Failure detection and recovery

Service interruption duration (ms)