ClickNF: a Modular Stack for Custom Network Functions

Massimo Gallo, Rafael Laufer
Introduction
Network Function evolution

TLS/SSL Server
Proxy Server
L7 Load Balancer
Firewall

NOKIA Bell Labs
Can we build a fully modular and customizable stack (L2-L7) for NF deployment without sacrificing performance?
Introduction
Network Function evolution

Can we build a fully modular and customizable stack (L2-L7) for NF deployment without sacrificing performance?
Click Router

Overview

- Elements that initiate packet flow have a **task** scheduled periodically
- *FromDevice* and *ToDevice* call push and pull within their tasks
Click Router
Features & Limitations

- **MODULARITY**
- **PROGRAMMABILITY**
- **MULTITHREADING**
- **KERNEL/USER SPACE**
- **NO TRANSPORT**
- **NO NETWORK ACCELERATION**
- **NO BLOCKING I/O**
- **NO APPLICATION (MODULARITY)**
ClickNF Design

- Monolithic Network stacks:
  - Lack of flexibility
  - Hard to customize and debug

- ClickNF, decompose it into several simple elements:
  - Unfettered access to the network stack
  - Module reuse
  - Easier cross-layer optimization
ClickNF
1. High-Speed Packet I/O

**DPDK MODULE**
- 10/40 GbE cards
- Polling to fetch packets
- Wrap into Click packet and push

**HARDWARE OFFLOADING**
- Hardware flow control
- IP checksum offloading
- TCP checksum offloading

**MULTICORE**
- Receive Side Scaling
- Dedicated per-core data structure
ClickNF
2. Click TCP

- Full compliance with standards
- Support for:
  - TCP options
  - Congestion control
  - RTT estimation
- Composed of 60+ elements
- Elements modify TCP Control Block stored in \textit{TCPInfo}
- Packet annotations and flags
ClickNF

3. Blocking I/O

• Click natively supports nonblocking I/O
  – Complex logic for applications
• Blocking I/O
  – Simpler, equivalent performance
  – Blocking for socket I/O multiplexing
• ClickNF introduces **blocking tasks**
  – If I/O request cannot be completed, task **yields** CPU
  – Task resumes → illusion of continuity
  – Backwards compatible
ClickNF

4. Application modularity

• ClickNF provides 4 building blocks:
  – TCPEpollServer
  – TCPEpollClient
  – SSLServer
  – SSLClient

• Application layer elements use packet annotations to control lower layers

• Simplify NF programming and code reuse
ClickNF

4. Application modularity

• ClickNF provides 4 building blocks:
  – TCPEpollServer
  – TCPEpollClient
  – SSLServer
  – SSLClient
• Application layer elements use packet annotations to control lower layers
• Simplify NF programming and code reuse
ClickNF
Experimental results

• **Hardware:**
  – 3 machines Intel Xeon® 40-core E5-2660 v3 2.60GHz, 64 GB RAM
  – Intel Xeon® 82599ES NICs with two 10 GbE interfaces

• **Software:**
  – Ubuntu 16.10 (GNU/Linux 4.4.0-51-generic x86 64)
  – Click 2.1
  – DPDK 17.02, DPDK Traffic generator

• **Tests:**
  – Microbenchmarks (Packet I/O, modularity, HW offloading, Performance)
  – Applications (HTTP Cache/Server, SOCKS4 proxy)
ClickNF
Microbenchmarks

- 64 bytes packet, increasing # elements
  - Modularity has a cost!
  - Mitigated by batch processing

I/O & Modularity

![Graph showing throughput vs number of elements]
ClickNF
Performance

• Small TCP connection (64bytes packet). Echo request <-> Echo reply
  – Slightly better than state of the art user-space stack
ClickNF Application

• SOCKS4 Proxy server, increasing page size
  – High performance
  – Composable i.e., add remove SSL/TLS
  – Simple modules i.e., 100s LoC
ClickNF

Conclusion

• Compose custom, modular NF (end-host or in-network)

• Good scaling properties

• Flexibility

• Open source https://github.com/nokia/ClickNF