Software Routers: Click

Hakim Weatherspoon
Assistant Professor, Dept of Computer Science
CS 5413: High Performance Systems and Networking
September 29, 2014

Slides used and adapted judiciously from COS-561, Advanced Computer Networks
At Princeton University
Goals for Today

• The Click Modular Router
Click Motivation

- **Flexibility**
  - Add new features
  - Enable experimentation

- **Openness**
  - Allow users/researchers to build and extend
  - (In contrast to most commercial routers)

- **Modularity**
  - Simplify the composition of existing features
  - Simplify the addition of new features

- **Speed/efficiency**
  - Operation (optionally) in the operating system
  - Without the user needing to grapple with OS internals
Router as a Graph of Elements

• Large number of small elements
  – Each performing a simple packet function
  – E.g., IP look-up, TTL decrement, buffering

• Connected together in a graph
  – Elements inputs/outputs snapped together
  – Beyond elements in series to a graph
  – E.g., packet duplication or classification

• Packet flow as main organizational primitive
  – Consistent with data-plane operations on a router
  – (Larger elements needed for, say, control planes)
Click Elements: Push vs. Pull

• Packet hand-off between elements
  – Directly inspired by properties of routers
  – Annotations on packets to carry temporary state

• Push processing
  – Initiated by the source end
  – E.g., when an unsolicited packet arrives (e.g., from a device)

• Pull processing
  – Initiated by the destination end
  – E.g., to control timing of packet processing (e.g., based on a timer or packet scheduler)
Click Language

• Declarations
  – Create elements

• Connections
  – Connect elements

• Compound elements
  – Combine multiple smaller elements, and treat as single, new element to use as a primitive class

• Language extensions through element classes
  – Configuration strings for individual elements
  – Rather than syntactic extensions to the language

```
src :: FromDevice(eth0);
ctr :: Counter;
sink :: Discard;

src -> ctr;
ctr -> sink;
```
User-level routing daemons

Forwarding plane

Control plane

- **Elements**
  - Small building blocks, performing simple operations
  - Instances of C++ classes

- Packets traverse a directed graph of elements
  
  \[
  \text{FromDevice(eth0)} \rightarrow \text{CheckIPHeader(14)} \rightarrow \text{IPPrint} \rightarrow \text{Discard};
  \]

Elements

- **element class**
- **input port** \(\rightarrow\) **Tee(2)** \(\rightarrow\) **output ports**
- **configuration string**
• **Push connection**
  – Source pushes packets downstream
  – Triggered by event, such as packet arrival
  – Denoted by filled square or triangle

• **Pull connection**
  – Destination pulls packets from upstream
  – Packet transmission or scheduling
  – Denoted by empty square or triangle

• **Agnostic connection**
  – Becomes push or pull depending on peer
  – Denoted by double outline
Push and pull violations
Handlers and Control Socket

• Access points for user interaction
  – Appear like files in a file system
  – Can have both read and write handlers

• Examples
  – Installing/removing forwarding-table entries
  – Reporting measurement statistics
  – Changing a maximum queue length

• Control socket
  – Allows other programs to call read/write handlers
  – Command sent as single line of text to the server
    – http://read.cs.ucla.edu/click/elements/controlsocket?s=llrpc
Example: EtherSwitch Element

• Ethernet switch
  – Expects and produces Ethernet frames
  – Each input/output pair of ports is a LAN
  – Learning and forwarding switch among these LANs

• Element properties
  – Ports: any # of inputs, and same # of outputs
  – Processing: push

• Element handlers
  – Table (read-only): returns port association table
  – Timeout (read/write): returns/sets TIMEOUT

http://read.cs.ucla.edu/click/elements/etherswitch
Implicit vs explicit queues

Implicit queue
• Used by STREAM, Scout, etc.
• Hard to control

Explicit queue
• Led to push and pull, Click’s main idea
• Contributes to high performance
An Observation...

• Click is widely used
  – And the paper on Click is widely cited
• Click elements are created by others
  – Enabling an ecosystem of innovation

• Take-away lesson
  – Creating useful systems that others can use and extend has big impact in the research community
  – And brings tremendous professional value
  – Compensating amply for the time and energy 😊
Can you build a Tbps router out of PCs running Click?
  - Not quite, but you can get close

RouteBricks: high-end software router
  - Parallelism across servers and cores
  - High-end servers: NUMA, multi-queue NICs
  - RB4 prototype
    • 4 servers in full mesh acting as 4-port (10Gbps/port) router
    • \( 4 \times 8.75 = 35 \text{Gbps} \)
  - Linearly scalable by adding servers (in theory)

Dobrescu, M., Egi, N., Argyraki, K., Chun, B., Fall, K., Iannaccone, G., Knies, A., Manesh, M., and Ratnasamy, S. RouteBricks: exploiting parallelism to scale software routers, SOSP 2009
Bolla, R. and Bruschi, R., PC-based software routers: high performance and application service support, PRESTO 2008
NetFPGA  

Network processor

Before Next time

- **Project Progress**
  - Need to setup environment as soon as possible
  - And meet with groups, TA, and professor
- **Lab0b – Getting Started with Fractus**
  - Use Fractus instead of Red Cloud
    - Red Cloud instances will be terminated and state lost
  - **Due Monday, Sept 29**

- **Required review and reading for Friday, October 3**
  - RouteBrics

- Check website for updated schedule