

Click Concepts

Click Modular Router Concepts and Philosophy

Bart Braem — Johan Bergs

University of Antwerp
iMinds - PATS Research Group



Outline

- 1 Background
- 2 Routers and Elements
- 3 Implementation Philosophy
- 4 Handlers
- 5 Running Click
- 6 References



Network coding

- Inside router: packets being processed while flowing through routers
- Classical network software: operate on packets, procedural programming style
- Code reuse? Lots of returning concepts



Click Modular Router

- Extensible toolkit for writing packet processors
- PhD thesis dr. Eddie Kohler (MIT)
- Architecture centered on elements
 - Small building blocks
 - Perform simple operations e.g. decrease TTL
- Click routers
 - Directed graphs of elements



Why not a daemon?

- I'll just code it as a daemon.
 - How modular is your result?
- Designing elements is hard, in the beginning
 - Follow the framework, it's there already
- Other alternatives:
 - System calls are hard
 - Which libraries for the packet formats?
 - The Linux kernel API is hard
 - Let's not talk about kernel development



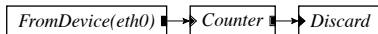
This has already been done

- So have linked lists
 - Did you do real network programming before?
 - E.g., do you know how ARP really works?
- Click is never used in real life
 - (Neither is Oberon)
 - It works at layer 3, it is in the network
 - Being used in production at large companies, links with e.g. Google



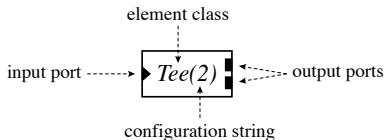
Click Routers

- Router: Elements connected by edges
- Output ports to input ports
- Describes possible packet flows
- `FromDevice(eth0) -> Counter -> Discard`;



Element Classes

- Class: element type (reuse!)
- Configuration string: initializes this instance
- Input port(s): Interface where packets arrive, triangles
- Output port(s): Interface where packets leave, squares
- Instances can be named: `myTee :: Tee`



Element ports I

Push port:

- Filled square or triangle
- Source initiates packet transfer: event based packet flow

Pull port:

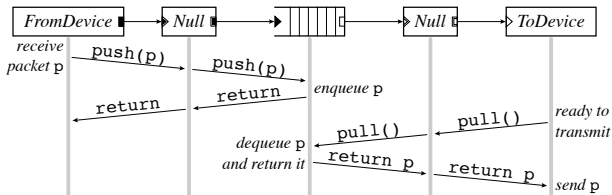
- Empty square or triangle
- Destination initiates packet transfer
- Used with polling, scheduling, ...

Agnostic port:

- Square-in-square or triangle-in-triangle
- Becomes push or pull (inner square/triangle filled or empty)



Element ports II



Push-pull violations I

Push port

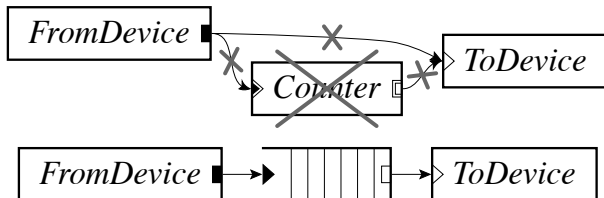
- has to be connected to push or agnostic port
- Conversion from push to pull with push-to-pull element
- E.g. queue

Pull port

- Has to be connected to pull or agnostic port
- Conversion from pull to push with pull-to-push element
- E.g. unqueue

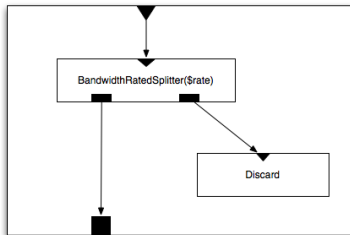


Push-pull violations II



Compound elements

- Group elements in larger elements
- Configuration with variables
- Pass configuration to the internal elements, can be anything (constant, integer, elements, IP address, ...)
- Motivates reuse



Element classes - Routers

Element classes?

- Elements (actually element classes): C++ classes
- Element instantiations: C++ objects

Click Routers:

- Click router configurations (or short Click routers): text files
- parsed when starting Click, Click builds object graph of elements



Click Graphs

Text files describing the Click graph:

- Elements with their configurations
- Compound elements
- Connections between elements

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

or

```
FromDevice(eth0) -> Counter -> Discard;
```



Input and output ports

Identified by number (0,1,..)

- Input port: $\rightarrow [nr1]Element \rightarrow$
- Output port: $\rightarrow Element[nr2] \rightarrow$
- Both: $\rightarrow [nr1]Element[nr2] \rightarrow$
- Only one port: number can be omitted

Motivates instance naming

```
mypackets::IPClassifier(dst host $myaddr,-);  
FromDevice(eth0)  $\rightarrow$  mypackets;  
mypackets[0]  $\rightarrow$  Print(mine)  $\rightarrow$  [0] Discard;  
mypackets[1]  $\rightarrow$  Print("the others")  $\rightarrow$  Discard;
```



Compound elements in Click scripts

```
elementclass DumbRouter { $myaddr |  
  mypackets :: IPClassifier(dst host $myaddr,-);  
  input[0]-> mypackets; mypackets[0]-> [1] output;  
  mypackets[1]-> [0] output;  
}  
u :: DumbRouter(1.2.3.4);  
FromDevice(eth0)-> u;  
u[0]-> Discard;  
u[1]-> ToDevice(eth0);
```



Element Configuration

Listed in click script

- First required arguments
- Then optional arguments
- Then arguments by keyword (after keyword)

Lots of data types supported

- Integers
- Strings e.g. "data"
- IP addresses 143.129.77.30
- Elements



Element Configuration Examples

- SimpleElement("data")
- SimpleElement("data",ACTIVE false)
- SimpleElement("moredata",800)
- SimpleElement("data",800,DATASIZE 67, SOURCE 1.2.3.4)



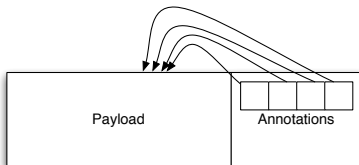
Packets

Packet consists of payload and annotations, payload:

- raw bytes (char*)
- Access with struct*

Annotations: metadata to simplify processing, “post-its”

- E.g. start of IP header or TCP header
- Paint annotations
- User defined annotations



Handlers

Like function calls to an element

- ReadHandler: request a value from an element
- WriteHandler: pass a string to an element
- (There is no ReadWriteHandler: you can't call a ReadHandler with arguments)

Can be called from other elements or through socket

- Take a look at Pokehandlers!



Use the source

- Source code: Open source
- Runs on Linux, Mac OS X, BSD and partially in Windows
- Makefile based
- Very little external dependencies (in-kernel)



Kernel module

- Completely overrides Linux routing
- High speed, requires root permissions
- Crashing Click = crashing kernel = crashing system

Userlevel

- Runs as a daemon on a Linux system
- Easy to install and still fast
- Recommended for this course

nsclick

- Runs as a routing agent within the ns-2 network simulator
- Multiple routers on 1 system
- Difficult to install but less hardware needed



References

- Click website: <http://www.read.cs.ucla.edu/click/>
- Element documentation (by name or category)
- Programming Concepts
- Doxygen documentation ('Internals documentation')



Click PhD thesis

PhD thesis: comprehensive documentation of every concept

Interesting chapters:

- Introduction
- Architecture: elements, packets, connections, push and pull, element implementation
- Language: syntax, configuration strings, compound elements

Click mailinglist:

<https://amsterdam.lcs.mit.edu/mailman/listinfo/click>
(Large source of information, mainly for Click developers)

