



Click Router: Hands on

Alex Newman

Arvind Venkatesan

Shivkumar Kalyanaraman

Knowing about your network

- Helpful Unix commands
 - ifconfig: Find about IP address and MAC address for all the network interfaces on this machine
 - ifup: Bring up a network interface
 - ifdown: Bring down a network interface
 - lspci: Lists all PCI devices in the system

Modular software router

- Click exists in User land or a Kernel Module for:
 - Linux
 - FreeBSD
- Click is modular !
 - I.e. we can remove/add components of Click (eg: queues etc)
 - Interconnected collection of modules called *elements*
 - Elements are written in **C++ !!**
- **It is also extensible - we can prototype our own designs!**
- Provides high level coding interface:
 - Eg: vectors, string buffers, ip address, queues etc

Features of Click

- Provides a high level programming interface to the developer
- Provides a number of router building blocks that can be used/modified to build custom router configurations
- Click is darn fast, in fact it often reaches the theoretical limits of the hardware.

Click Scripts

- *Before we have much understanding lets setup our environment and work with a basic Click Script.*
- *This should demonstrate how fast and human readable click is.*
- *How about a program to pass packets from one Ethernet card to another?*
- *FromDevice(eth0) -> Queue() -> ToDevice(eth1);*
- *But this doesn't do anything right, as what will happen to the packet once it gets sent out of ETH1?*

Exercise 1

Problem: Use Encapsulation elements that do header manipulation like strip, prepend etc. provided in the Click Toolkit to fix the router config.

You should probably consult the element reference on the Click webpage

Hint it is linked off of

<http://pdos.lcs.mit.edu>

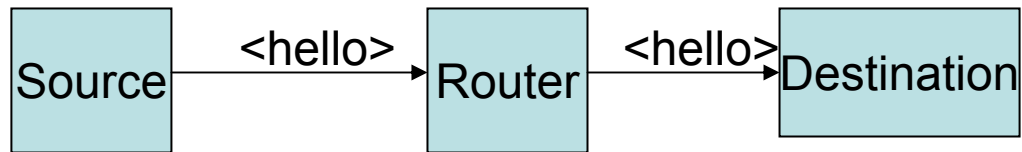
THINKING ALOUD (Exercise 1)

1. Receive!
2. Every incoming packet has an ethernet and an IP Header
3. Remove existing ethernet header
4. Put next hop's Ethernet Header
5. Send!

Exercise 2

Problem: Write a small configuration file to implement a packet source and a packet destination. The packet source sends a "<hello>" in an IP packet. The destination simply drops the "<hello>" packet 😞

Figure: Exercise 2



THINKING ALOUD (Exercise 2)

1. SOURCE Machine

- Make a packet with the contents
- Prepend IPHeader
- Prepend EthernetHeader
- Send

2. DESTINATION Machine

- Receive a packet
- Discard

Elements

- FromDevice , ToDevice and Queue are all click elements.
- Each Elements is an object which inherits some basic attributes.
- Types of elements
 - Push: Pushes packet to the next element. E.g. FromDevice();
 - Pull: Pulls packet from the previous element E.g. ToDevice();
 - Agnostic: May act as push or pull. E.g. Paint();
- Method Interfaces
 - Methods exported to other elements. For example, methods for transferring packets like push(), pull() etc.
- Handlers
 - Methods exported to the user rather than to other elements in the router configuration
 - Support simple, text based read/write semantics

The Click Configuration File

- Describes connectivity between elements
- Acyclic directed graph
- Allows configuration arguments to be specified for the elements
- Configuration Language
 - Declaration: **name :: class (config);**
E.g. Buffer::Queue(500);

This is actually closer to typedefing.

- Connections: **name1 [port1] -> [port2] name2;**
E.g. IPClassifier[0] -> Discard; //Firewalled!
IPClassifier[1] -> Buffer -> ToDevice(eth0); //Forward

Notice each pipeline is its own atomic event driven system.

Element Classes

- Click elements are atomic objects which can inherit from each other.
- C++ supports Multiple Inheritance, therefore so does click.
- An elements ability to support push and pull depends on whether or not that function was written mainly.

Element Classes: A look at the C++ code

```
class NullElement: public Element {
public:
    NullElement();
    const char *class_name() const;
    NullElement *clone() const;
    const char *processing() const;
    void push(int port, Packet *p);
    Packet *pull(int port);
};
```

Element Classes: A look at the C++ code

```
NullElement() {
    add_input();
    add_output();
}
const char *class_name() const {
    return "Null";
}
NullElement *clone() const {
    return new NullElement;
}
const char *processing() const {
    return AGNOSTIC;
}
void push(int port, Packet *p)    {
    output(0).push(p);
}
Packet *pull (int port) {
    return input(0).pull();
}
```

Making own elements

STEPS:

1. Descend into the Click main directory
2. *# make elemlist*
3. *# make install*

Running router configuration

Kernel Space

- *# rmmmod click* (within CLICKDIR)
- *# insmod linuxmodule/click.o* (within CLICKDIR)
- To start : *# cat "filename.click" > /proc/click/config* (whereever config file is)
- To stop: *# cat "null.click" > /proc/click/config* (whereever null.click is)

User Space

- *\$ click "filename.click"*

Debugging Aids

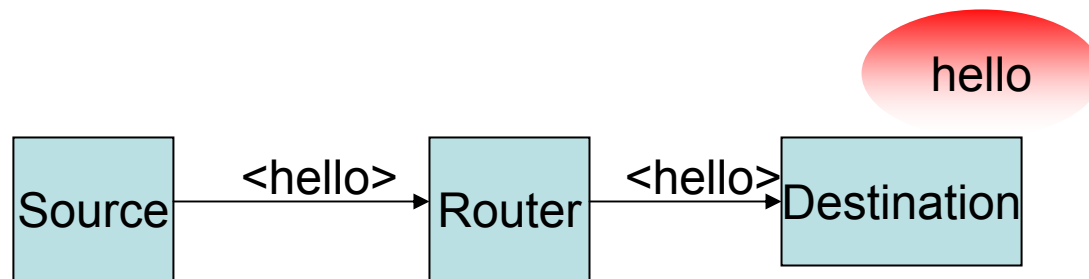
- Print messages with `click_chatter()`
- Use `dmesg` at the prompt
- Read `/var/log/messages`
- Use `ksymoops` (oops! What's that? 😊)

Exercise 3

Problem: Instead of dropping the packet at the destination, just echo "<hello>" on the screen and drop the packet.

- Modify the Null element shown previously to echo the packet contents.
- Use this "modified" NullElement" in your configuration file

Figure: Exercise 3



THINKING ALOUD (Exercise 3)

DESTINATION Machine (MODIFIED)

- Receive a packet
- Strip its Ethernet address
- Strip its IP Address
- Print the packet contents using your element (use `click_chatter()` function)
- Kill the packet

Exercise 4

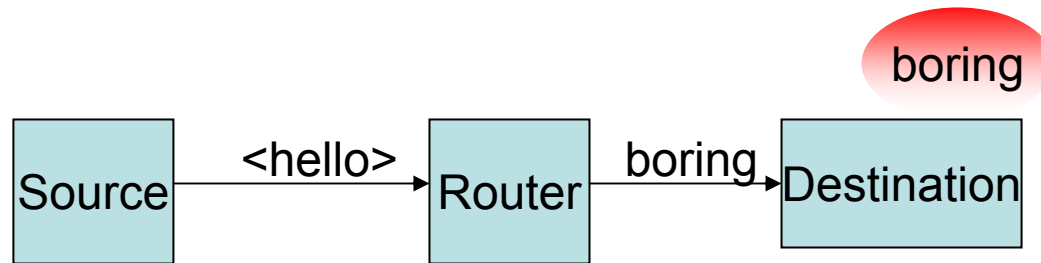
Problem: Man in the middle attack!!!!!!
Let router R forward "boring" instead of
"<hello>"

Write a new element that:

- Creates a new packet with the new contents
- Kills the incoming packet (use the kill() function)
- Forwards the new packet

Include this element in your config file

Figure: Exercise 4



THINKING ALOUD (Exercise 4)

- ROUTER

- Receive a packet
- Strip its ethernet address
- Strip its IP address
- Within your element make a new packet with the new contents (use `make()` function on a `WritablePacket`)
- Kill the received packet
- Send the new packet out

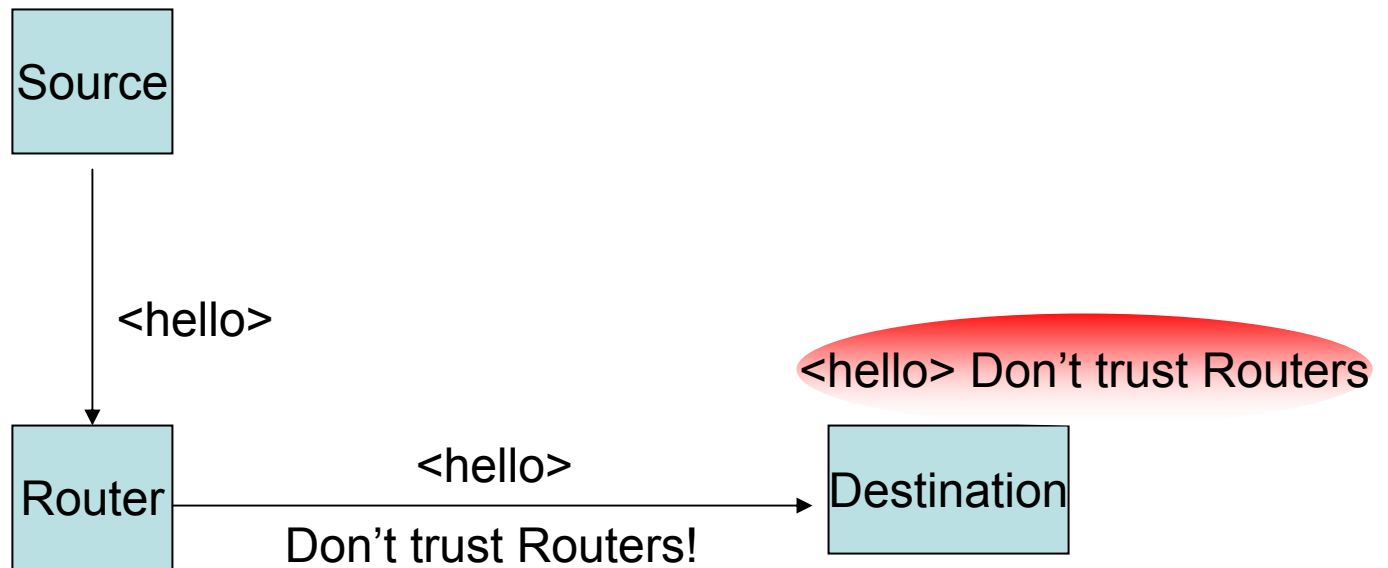
Exercise 5

Problem: Mission TimeBomb

You are secret agent 0049! With all the networking powers, you get access to the Router code that corrupts the packet. Now put a timer in that router code that times out every 1000ms and sends a packet to the destination saying "Don't trust Routers!"



Figure: Exercise 5



Hints: Exercise 5

- Use a Timer object (include click/timer.hh)
- Initialize the timer
- Implement the function `run_scheduled()` that is called every time there is a time out. In this function
 - Make a packet with the new contents
 - Push the packet out
- Reset the timer every 1000ms using the `reschedule` method on a Timer object.

What did we do today?

- Learnt innards of click router
- Make new elements and of course router!
- Make your own source, destination nodes
- Processed packet within a router
- Blew the router up!!! 😊