Click modular router

Click Modular Router: A Brief Introduction
Andrea Bianco, Robert Birke, Nanfang Li
Telecommunication Network Group
http://www.telematica.polito.it/

Stile prototipico taken from
Bart Braem – Michael Voorhaen: Click Modular Router Concepts

What is Click?
• Click is a software architecture that ease the task of programming routers (more in general, providing packet processing features)
• Key idea
– Modular architecture: a click router is built from small components, named elements, which allow to have a fine-grained control over the forwarding path (divide and conquer concept)
• Main benefit:
– Programmability
– Flexibility
• Website
http://read.cs.ucla.edu/click/

Click Architecture
• Click routers are build from small components called elements
• Each element implements a "simple" operation
– Example: decreasing the TTL field, checking a CRC, queuing packets, …
• Connections between two elements represent possible paths followed by packets
• Therefore, a Click router configuration is a directed graph, using elements as vertexes
– The graph describes the connections among elements

Click Configuration: An Example
• A click configuration files describes a directed graph
– Vertices
– Elements processing packets
– Each element has one or more input/output ports
– Directed arcs
– Connect ports
InfiniteSource(…)
→ Strip(14)
→ Align(4,0)

Elements
• Basic building blocks
– Each elements implements a given operation
• Implemented as C++ classes
– Easy to reuse
• Symbol
– Elements are represented by Rectangles
• Have I/O ports
• Initialized through configuration string
• Large pool of elements available
– http://read.cs.ucla.edu/click/elements
• Possible to create/customize its own element

Elements: ports
• Each element may have several input and output ports
• Input ports
– Interface where packets arrive
– Symbol: triangle
• Output ports
– Interface where packets leave
– Symbol: rectangle
• Each port might have a different semantic
– Example: ARPQuerier element (2 inputs, 1-2 outputs)
– Receive IP packets on input 0 and ARP replies on input 1
– Sends Ethernet encapsulated IP packets on output 0 and ARP requests on output 1
Elements: configuration

• Each element instance can have a configuration string (set of parameters)
  – To set initial state
  – To customize element behavior
• Available parameters are specified in each element description
• Example:

An available element: Queue

• Packets are normally not stored by elements
• To store packets use queues
• Queues are also implemented as elements
  – Explicit data storage element
  – More flexibility on where packets are stored
    • E.g. possibility to put multiple queues at the output

Port Types

• Push ports:
  – Source initiates data transfers (packets are being pushed)
  – Connections using push ports:
    • Upstream element hands packet to downstream element
    • Symbol: filled port
  • Better suited:
    • Packet generation or reception from the NIC
• Pull ports:
  – Destination initiates the data transfer (packets are being pulled)
  – Connections using pull ports:
    • Downstream element requests packet from upstream element
    • Symbol: empty port
  • Better suited:
    • Packet scheduling or NIC transmission

Push pull constraints

• Outputs must be connected to inputs of the same type
  – Pull with pull, push with push: i.e. ports must have the same color
  – Agnostic ports can be only used as either push or pull
• Conversion possible using specific elements:
  – Push-to-pull: e.g. queues
  – Pull-to-push: e.g. unqueue
• Example

Click Modular Router Configuration Language
Click modular router

Click Configuration: Elements

- A click configuration is a text file describing a click graph:
  - elements + connections

- Elements
  - Named
  - Unnamed

- Configuration string
  - Initialize elements
  - Specified between brackets as a comma separated list
  - Ordering
    - optional arguments
    - arguments with keyword

- Lots of types supported
  - Integers
  - Strings e.g. "data"
  - IP addresses 143.129.77.30

- Elements

- Examples
  - SimpleElement (argument, [argument], KEYWORD value)
  - SimpleElement ("data")
  - SimpleElement ("data", ACTIVE false)
  - SimpleElement ("moredata", 800)
  - SimpleElement ("data", 800, DATASIZE 67, SOURCE 1.2.3.4)

Click Configuration: Connections

- Connections
  - Specified with an arrow "->" between elements
  - E.g.: FromDevice(eth0) -> Counter -> Discard;
  - Ports specified by numbers inside square brackets:
    - Numbering starts from 0
    - Input ports: [nr1]SimpleElement
    - Output ports: SimpleElement[nr2]
    - Both: [nr1]SimpleElement[nr2]
    - Motivates element instance naming

- Possible to create compound elements by grouping simple elements
  - Configuration can be passed to internal elements
  - Motivates reuse

- Example:
  - Keywords in bold
  - Variables in italic

  Def:
  ```
  elementclass SplitTrafficCounter {
  $ethType|
  class :: Classifier(12/$ethType, -);
  input[0] -> class;
  class[0] -> Counter -> [0]_output;
  class[1] -> Counter -> [1]_output;
  }
  
  Use:
  stc :: SplitTrafficCounter(0806 /*ARP */);
  ```

Compound elements

- Most elements are independent of the mode used
- Some elements are specific of a certain mode or have different implementation depending on the running mode

Running click

- Click can be run in different modes
  - Kernel module
  - User level
  - NSclick (used with NS2 for simulations)

- Most elements are independent of the mode used

- Some elements are specific of a certain mode or have different implementation depending on the running mode

  - The driver column specifies the required mode:
    - Click Modular Router: Running Click

Network management and QoS provisioning - 13
Andrea Bianco – TNG group - Politecnico di Torino

Network management and QoS provisioning - 14
Andrea Bianco – TNG group - Politecnico di Torino

Network management and QoS provisioning - 15
Andrea Bianco – TNG group - Politecnico di Torino

Network management and QoS provisioning - 16
Andrea Bianco – TNG group - Politecnico di Torino

Network management and QoS provisioning - 17
Andrea Bianco – TNG group - Politecnico di Torino

Network management and QoS provisioning - 18
Andrea Bianco – TNG group - Politecnico di Torino

Pag. 3
Kernel Mode

• Click runs directly inside the kernel
• The Linux stack is completely overridden
• Best performance
  – Custom drivers for specific NICs (mainly Intel e1000 driver) are available boosting even further the performance
• However
  – Requires a patched kernel
  – Crashing click, crashes the kernel and the system

User Level

• Click runs as a normal program
• Click can read/write packets from/to the NIC (Network Interface Card) using Berkeley Packet Filter or BPF dumps
• Slower
  – Packets are processed by both the kernel and Click
• However
  – Easy to run: no modified kernel/drivers needed
  – Crashing Click dose not affect the system

Compiling Click

• Source code available from website:
  http://read.cs.ucla.edu/click/download
• Decompress archive (may not be needed)
  – tar xzvf click-1.8.0.tar.gz
• Change directory
  – cd click-1.8.0
• Compile source code using:
  – ./configure
  – make
• Install click on system
  – make install

Clicky

• GUI for click
• Run clicky
  – clicky
• Clicky
  – Generates graphs from click-configuration
  – Provides access to the handlers
  – Real-time overview of running configuration
  – Allows to export diagrams to PDF

Kernel Mode

• To load a click configuration
  – click-install [configuration_file]
• To remove a running configuration
  – click-uninstall
• The above commands require root privileges
  – Either become root or use sudo
• Man pages are available for additional help on usage
  – man click-install
  – man click-uninstall

User Level

• To run a click configuration
  – click [configuration_file]
• Notes:
  – click will normally run until interrupted
  – -p [port] can be used to control a running click router over a TCP connection (i.e. using clicky)
  – See also man click
  – Some functionalities need root privileges
    • E.g.: activation of promiscuous mode
Click modular router

Compiling Click: Notes

- Kernel module needs (patched) kernel sources to compile
  - Kernel to use can be specified with:
    - `./configure --with-linux=[/path/to/linux_source] --with-linux-map=[/path/to/system.map]

- Other config options can be seen using:
  - `./configure --help`

- To recompile after modifying an existing file
  - `make`
  - `make install`

- Elements source code is made of two files (.cc and .hh) found in:
  - `elements/[package_name]` (see element documentation)
  - Example: elements/standard

Handlers

- Some elements allow interaction at execution time

- Interaction based on handlers
  - Read handlers: request a value from an element
    - E.g.: packets stored in the queue
  - Write handlers: pass a string to the element
  - Handlers can be called from other elements or from socket (or procfs)

Handlers Example: User Level

- Run click (enable interaction on port 5555)
  - `click -p 5555 [click_conf]`
- Connect to running router using telnet:
  - `telnet localhost 5555`
- Once connected, issue commands:
  - Read handlers
    - `read <elementname>.<handlername>`
  - Write handlers
    - `write <elementname>.<handlername> <values>`
  - List elements
    - `read list`

Handlers Example: Kernel Mode

- Run click
  - `click-install [click_conf]`
- Handlers called using the /proc filesystem
  - Info on running configuration found in
    - `/proc/click`
  - Read handler: read file
    - `cat /proc/click/<element_name>`
  - Write handler: write values to file
    - `echo "values" > /proc/click/<element_name>`

Creating a New Element

- Easiest way to copy the .cc and .hh file from an existing element and modify it
  - Change the class name
  - Common functions to override
    - `char *class_name()`
      - Return element's name
    - `void push(int, Packet*)`
    - `Packet* pull(int)`
      - process pull request on port I
    - `Packet* simple_action(Packet*)`
      - for agnostic element
Creating a New Element

- Common functions to override
  - int configure(Vector<String> & c, ErrorHandler *)
    - process element configuration string
  - char *port_count()
    - return port count code
  - char *processing()
    - return processing code
  - void add_handlers()
    - add element handlers

- int initialize(ErrorHandler * eh)
  - initialize element

- void cleanup(CleanupStage)
  - cleanup element state

- Check that the element is exported using the macro
  - EXPORT_ELEMENT(ElementName)
- Set requirements as needed
  - e.g.: ELEMENT_REQUIRES(linuxmodule)

Creating a New Element

- Copy the two files to the elements to the most appropriate subdir of elements/
  - elements/local is thought for local elements
- Run:
  - make element
- Check if the new element has been detected correctly
  - .cc file should be found in userlevel/elements.conf or in linuxmodule/elements.conf
- Compile and install:
  - make
  - make install

- Note: a single .cc/.hh file pair might contain more elements

Packets

- Packet consists of
  - Payload (header+data)
    - char*
    - Access with struct*
  - Annotations (metadata introduced to simplify processing)
    - "post-it"
    - IP header information
    - TCP header information
    - Paint annotations
    - User defined annotations

Live (Ubuntu + Click) DVD

- The DVD can be run without any modification to the system (just boot from the DVD)
- To create an additional copy of the DVD just burn the iso found here
  - http://www.telematica.polito.it/click.iso
- Using the "USB startdisk creator" from Ubuntu the iso can also be installed on a USB drive
  - (currently LAB PCs do not support this boot option)
  - System -> Administration -> Create a USB startup disk
Live DVD

• Click source found in
  – /home/ubuntu/click-1.8.0
• Click examples found in
  – /home/ubuntu/click-example-confs
• Click documentation found in
  – /home/ubuntu/click-docs