Report of Click Labs

Hong-Nam Hoang, Manh-Ha Nguyen and Xuan-Thu Thi Le

February 2, 2011

1 Introduction

2 ClickLabs package

2.1 File organization

elements/ This directory contains all the additional click elements using in the lab.

plot-template/ This directory contains templates used for plotting data by gnuplot. These files are used by draw-graph.sh

bin/update-elements.sh Run this file to update the new elements implemented in directory elements (above). For more information, type: ./update-elements.sh -h

bin/visual-clicky.sh Shell script to visualize click experiment using clicky. For more information, type: ./visual-clicky.sh -h

bin/init.sh Initialize Click environment for lab. Just run init.sh in the first time you get this source or click source directory changed.

bin/eclick-compile.sh Extend the Click file. A click file can include another one to reuse some compound elements (similar include in C, or import in Java). File eclick-compile.sh is used to translate (or flatten) these extended-click file to a normal click file.

bin/convert-click-dump.sh This script used to transform dump files from click (binary files) into text files. Note: this is one-way transformation, the binary files cannot be recovered from the text files.

bin/draw-graph.sh This script is used to draw graphs from data extracted in CLICK dump files. Just provide the dump files, this script will generate a graph for you. Note: No need to use convert-click-dump.sh before using draw-graph.sh.

bin/draw-graph-framerelay.sh Based on draw-graph.sh, this script helps to show the characteristics of verifying a conformant flow (which is deal with CIR, CBS, EBS).

clicky.ccss File supporting Clicky Cascading Style Sheets. It controls the appearance of a Clicky diagram with style sheets written in a CSS-like language.

1-test-config/

2-tcp-udp-generation/

3-shaper-policer/

4-scheduler/

2.2 Some introductions before surfing click configurations

1. First of all, initialize the click environment for these stuffs. Run file init.sh:
   ```
   chmod +x init.sh
   ./init.sh
   ```
   Normally, init process takes long time for the first finding Click source path. To save time, you can create file */.clickrc* with the content similar to this:
   ```
   export CLICK_SRC=/home/iizke/click/click-1.8.0
   ```

2. While finishing to code some Click elements, put it in directory 'elements', and then run file update-elements.sh to compile and install new elements:
   ```
   chmod +x update-elements.sh
   update-elements.sh
   ```

3. Explore the click configuration by using tool visual-clicky.sh. Simple way to use:
   ```
   visual-clicky.sh $CLICK_CONFIGURATION_FILE
   ```

4. To support easy-reading and team-working activities, we developed a tool to allow including some click files into a click file. If you write some click files as library files, you can reuse it by using 'include statement'. For example, we have TCP_Source.click to implement a TCP-generator, and UDP_Source.click to implement an UDP-generator. In TCP_UDP.click, we reuse the implementation of these generator by adding these lines at anywhere in TCP_UDP.click file (but should be in the top for easy reading):
   ```
   ------------------------ file:  TCP_UDP.click ---------------------------
   //include "TCP_Source.click"
   //include "UDP_Source.click"
   ...
   ------------------------------------------------------------------------
   ```
   The syntax of include statement is simple:
   ```
   //include "CLICK_PATH"
   ```
   where CLICK_PATH can be relative or absolute path. After that, you have to use our tool (eclick-compile.sh) to precompile this file before simulating it by CLICK, for example:
   eclick-compile.sh -o extend-TCP_UDP.click [-f] TCP_UDP.click
   Note: if using tool visual-clicky.sh, you don't have to pre-compile the extended-click file. It will do automatically.

5. To visualize your packet stream at input or output, we have developed draw-graph.sh to generate graph as picture (using gnuplot). So, before using this function, make sure that you have installed gnuplot. The second, you have to provide the data. Normally, we usually generate data from CLICK with elements ToDump. This data follows the format of tcpdump. When you get the data, the last action you need is running this command:
   ```
   draw-graph.sh -f data1.dump -f data2.dump -f data_n.dump
   o PNG_FILES
   [--plot-type COUNT (default) | RATE | DENSITY]
   [--xrange 233:23221]
   [--yrange 282:2922]
   [--xlabel XYZ]
   [--ylabel ABC]
   [--xcol 2]
   [--ycol 1]
   ```
   After program "draw-graph.sh" finishes its work, it will create a picture file (PNG file). If user doesn't use output option (-o), this program will export to screen (using default output file /dev/output). This tool is young, so the plot-template is hard code. It will be change to be able to use another templates.

3 Test configuration

In the first time of using click, we try to implement `Counter_test` element, `Random_IP_generator` element using basic Click elements, such as `Print, InfiniteSource, RatedSource, Script`, also trying to modify a part of source code of `InfiniteSource` to generate packets that randomize byte value at a specific location in payload.

3.1 `Counter_test` Click configuration

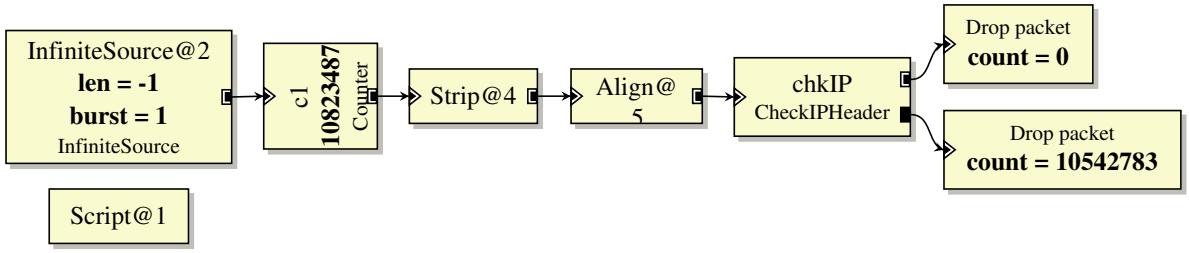

Figure 1: `Counter_test` Click configuration

To avoid IP CRC checking, we temporarily disable CRC checking by using flag "`CHECKSUM false`" in `CheckIPHeader` element. Another solution is to use `SetIPChecksum` to repair CRC in generated IP packets. Since We can visualize the result by using clicky to replace steps that print out screen counting results from Counter elements.

3.2 `RandInfiniteSource` element

Test RandInfiniteSource

3.3 `RandomQueue` element

Test RandomQueue

3.4 `Random_IP_generator` configuration

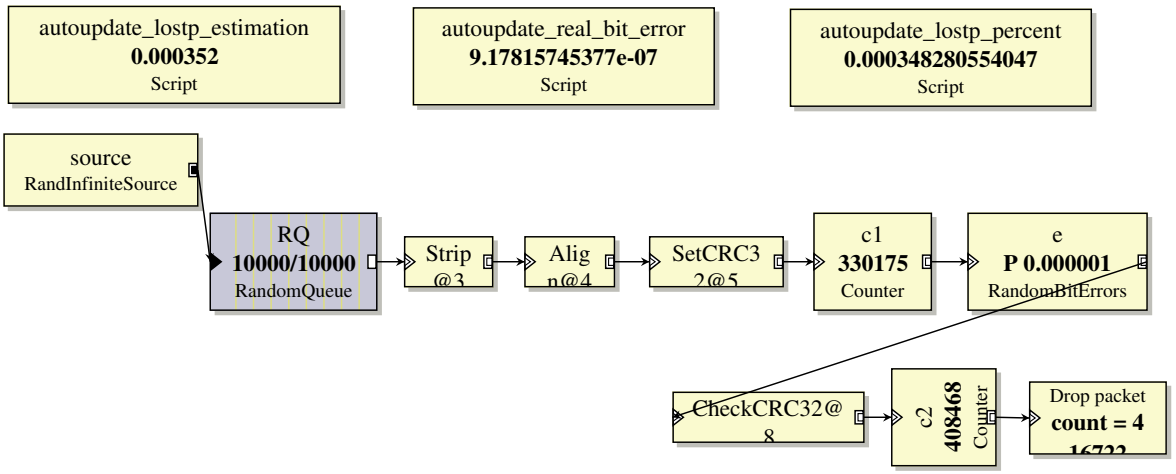

Figure 2: `Random_IP_generator` configuration

4 TCP/UDP traffic generation

4.1 TCP traffic

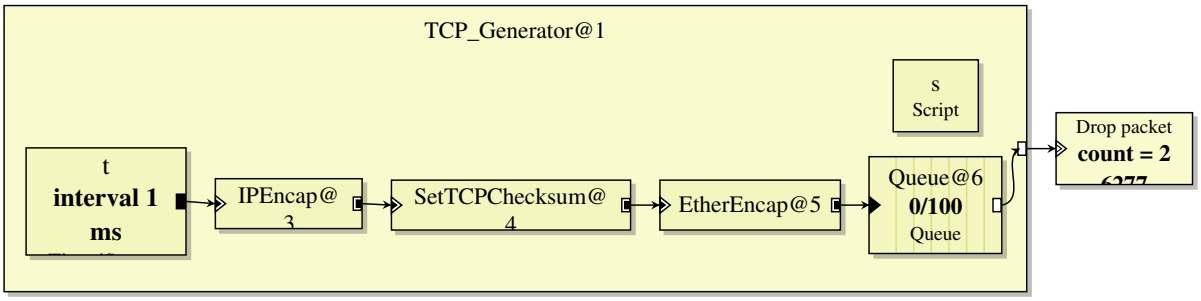

Figure 3: `TCP_Source` element

4.2 UDP traffic

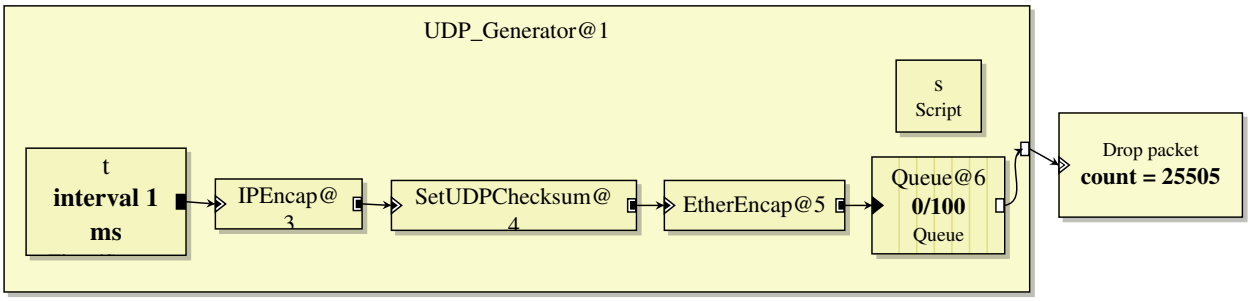

Figure 4: `UDP_Source` element

4.3 `TCP_UDP_generator` element

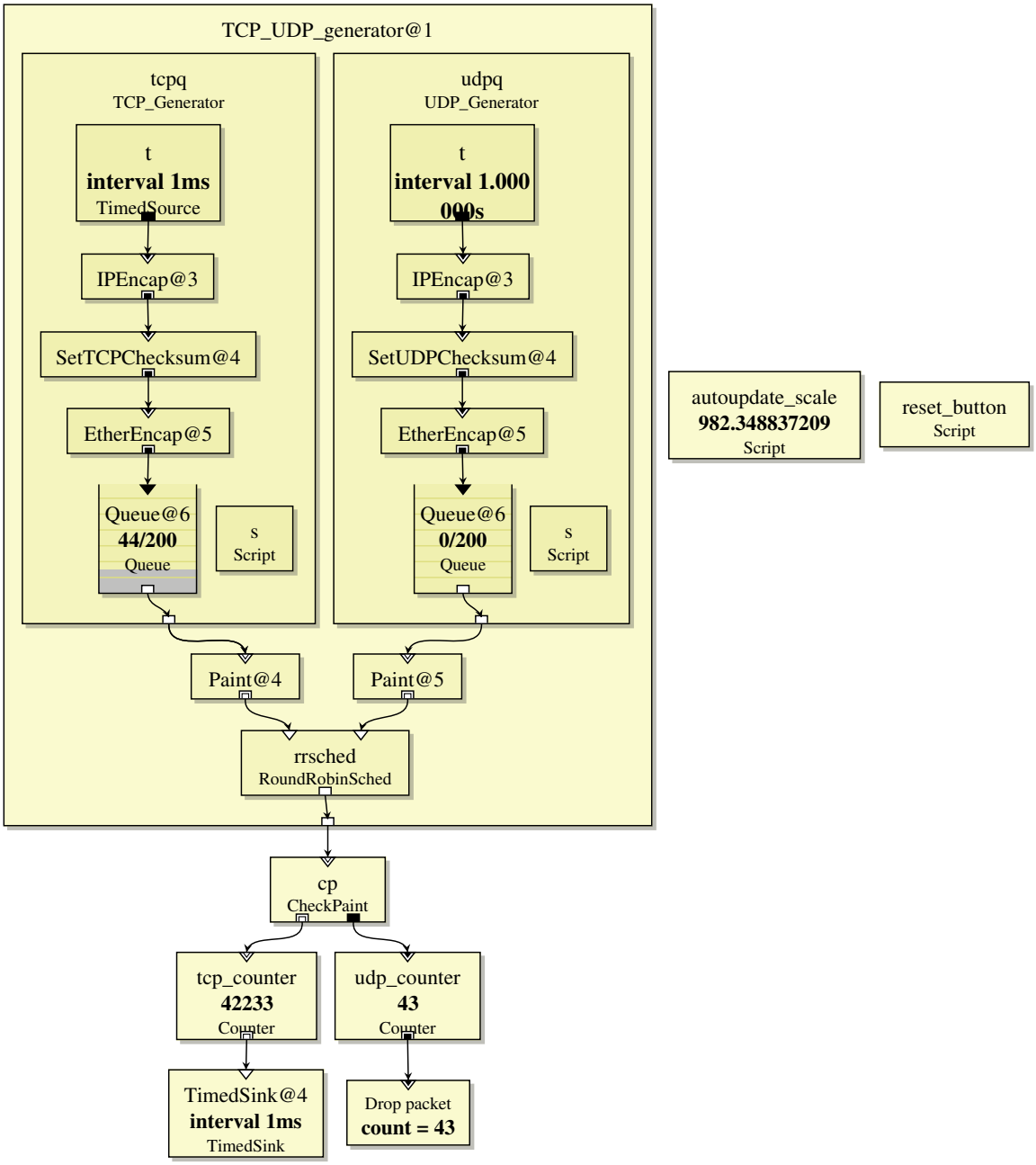

Figure 5: `TCP_UDP_generator` element

5 Shapers and Policers

5.1 Uncontrolled flow

We have tried some implementations of uncontrolled flow but the main idea is that the inter-time (interval) between two consecutive packets is a random number.