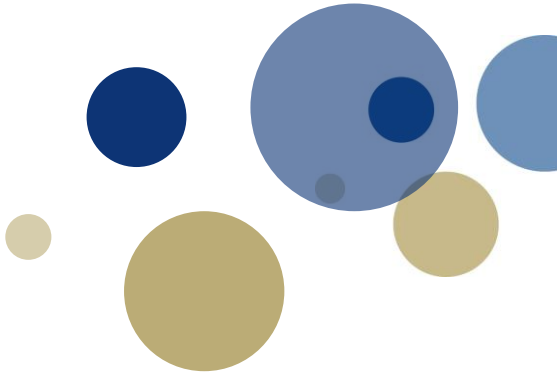
NTNU – Trondheim
Norwegian University of
Science and Technology



Communication Network Modeling for Real-Time HIL Power System Protection Test Bench

(Charles M Adrah, Zhou Liu, Øivind Kure, Hans Kristian Høidalen), IEEE PES, 2017

Charles M Adrah
Department of Telematics
23rd May, 2017

Outline

- **Introduction**
  - Challenges & Motivation
- **Network emulator design & functionalities**
  - Architecture & Features
  - Integration of emulator and HIL Test bed
  - New elements
- **Case Studies, Results & Evaluations**

1. Introduction

- **Challenges & Motivation**
  - Hardware-in-the-loop & Protection relay testing
  - Deploying power system protection applications with communication technology
  - OPAL-RT simulator limited in the ability to simulate communication network impairments
- Need to factor communication effects in power system protection applications

Proposed Solution

- Communication network emulation
  - Click Modular Router as the tool

Network Emulator Design & Functionalities

- Emulator enable us to change delay, jitter and packet corruption as a function of time.
- Control communication properties between multiple source relays and destination relays
- Impairing specific subsets of the network traffic
- Vary the properties of communication parameters in real time using - using handlers
- Bandwidth restriction
- Emulate different queueing schemes and traffic priorities

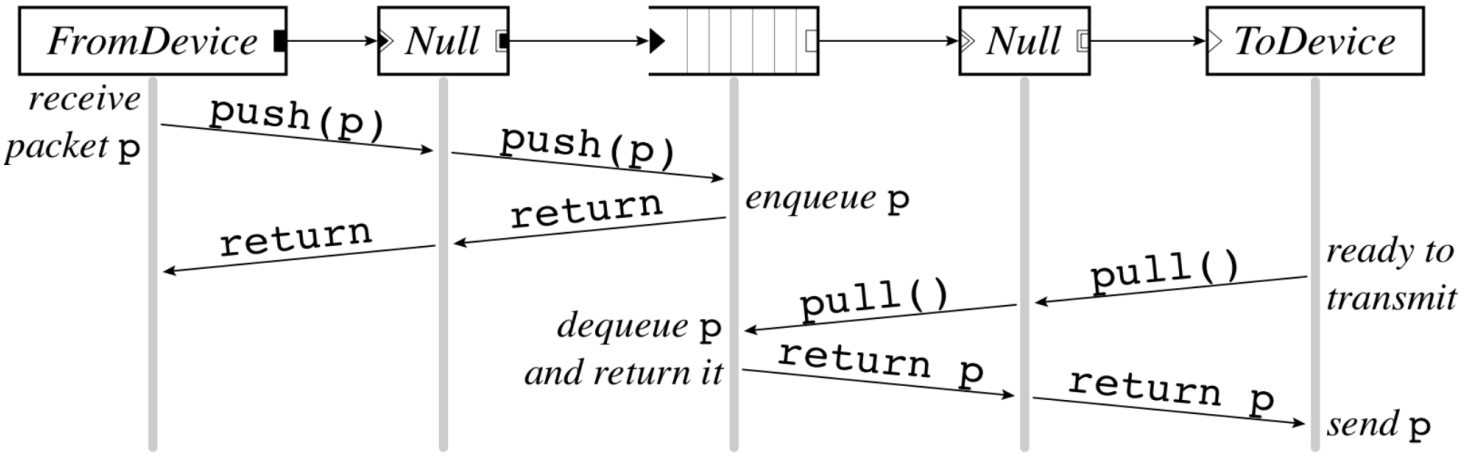
Protection application use cases

- Investigate effects of packet delays, packet delay variations on power system protection applications.
- Investigate effects of message lost (SMV/GOOSE) in protection application

Motivation for choice of tool: Click Modular Router

- Flexibility
  - Adding new features to enable experimentation
- Openness
  - Allow users to build and extend
- Modularity
  - Simplify the composition of existing features & addition of new features
- Speed
  - User does not need to know internals of operating system operations.

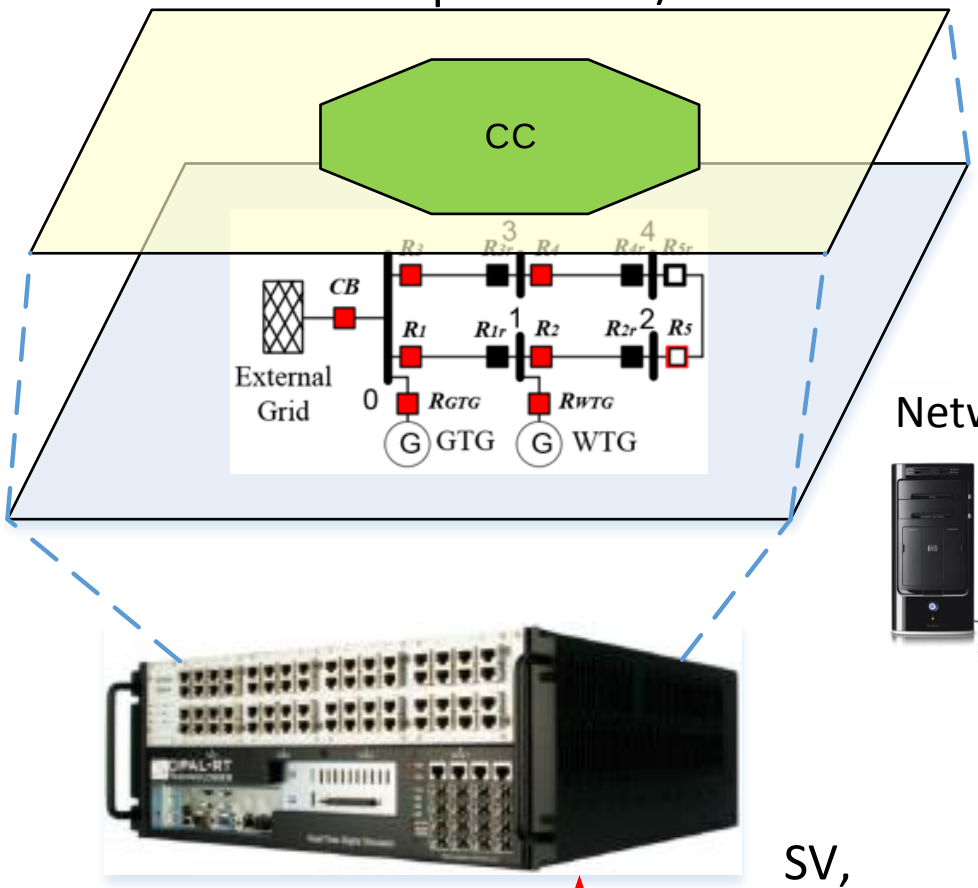
Router as a Graph of Elements & Push and Pull Connections



- 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

Integration of Emulator and HIL : Setup

Test power system network with CC in Opal RT-Lab/Matlab



Practical protective relays

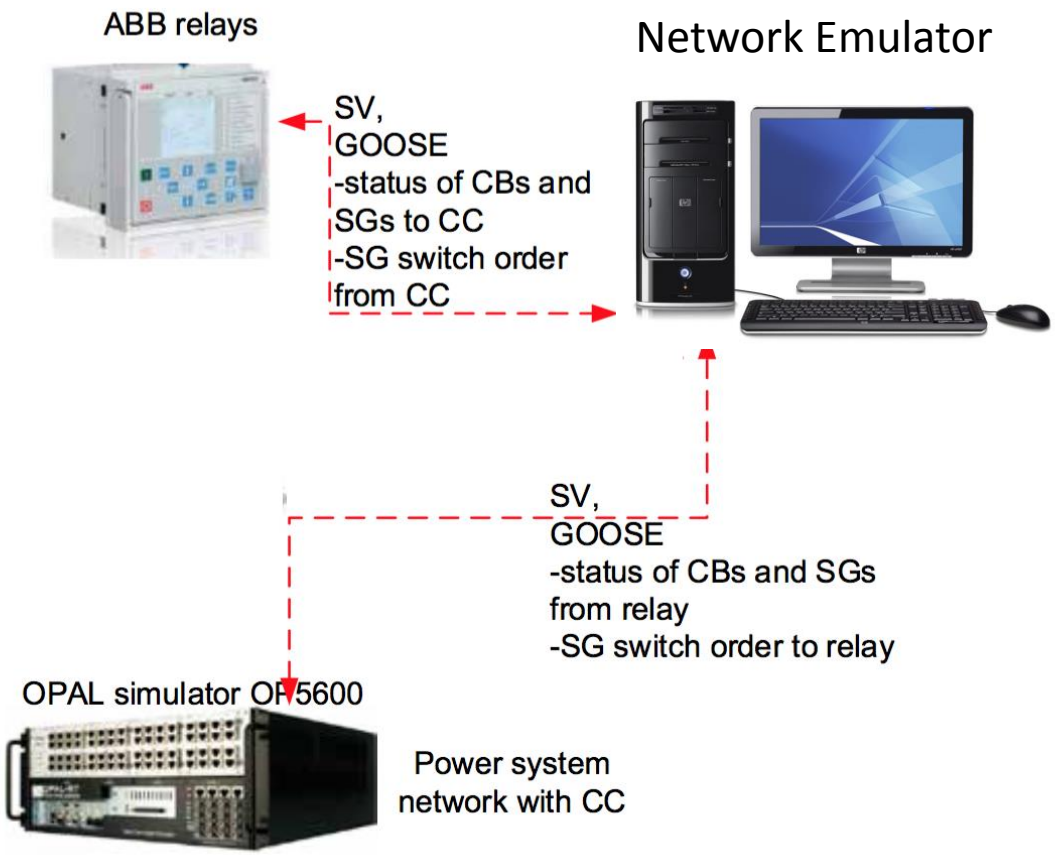
Network Emulator

SV,
GOOSE
-status of CBs and SGs to CC
-SG switch order from CC

OPAL simulator OP5600

SV,
GOOSE
-status of CBs and SGs from relay
-SG switch order to relay

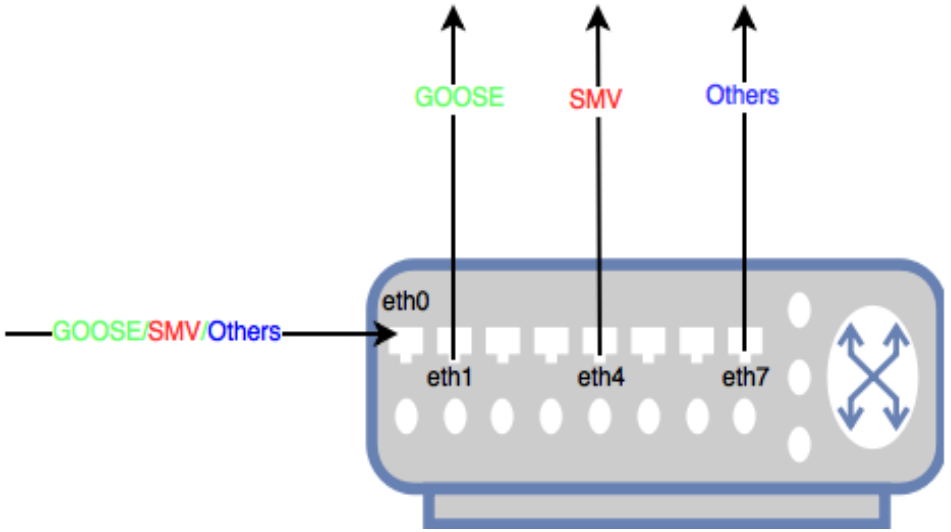
Integration of Emulator and HIL : Setup



Composing new elements: Emulator with IEC 61850 Capabilities

- Classifying IEC 61850 GOOSE & SMV packets

```
require(clicklocal);


        FromDevice(eth1)-> ic :: IECClassifier(GOOSE, SMV, -)

        ic[0]-> Queue(10000)-> ToDevice(eth0);

        ic[1]-> Discard;

        ic[2]-> Discard;

ClickyInfo(STYLE @import test-clicky.ccss);
```



Composing new elements : GOOSE Classifier

```
require(clicklocal);


         FromDevice(eth0)

              -> ic :: IECClassifier(GOOSE, SMV, -)

        ic[0]-> Strip(14)

              -> CheckGOOSEHeader()

              -> gc :: GOOSEClassifier(GOOSE_TRIP1, GOOSE_OV2PTOV, -)

        gc[0]-> ToDevice(eth1);

        gc[1]-> ToDevice(eth2);

       gc[2] -> Discard;

        ic[1]->Discard;
        ic[2]->Discard;
```

```
CLICK_DECLS


GOOSEClassifier::GOOSEClassifier() { };
GOOSEClassifier::~GOOSEClassifier() { };


int
GOOSEClassifier::initialize(ErrorHandler *errh)
{

    return 0;
}


void GOOSEClassifier::push(int,Packet *p) {
      unsigned  _textlength = TEXTLENGTH;

      int out_port = 2; // Default output for others

         const unsigned char *data = (p->data()+_textlength);
         unsigned  goCBRefLength=0;
         //  sa<<((*(data-1)) & 0xff);
         if((*data & 0xff) == goCBRefTag){
           data++;
          goCBRefLength = ((*data) & 0xff);
          //  unsigned char goCBRefData[goCBRefLength];
          /* for (unsigned i = 0; i < goCBRefLength ; i++){
             data++;
           goCBRefData[i] = { (*data));
           }*/
            }

        const unsigned char *data2 = (p->data()+_textlength+2+goCBRefLength);
        unsigned talLength=0;
         if((*data2 & 0xff) == talTag){
           data2++;
           talLength = ((*data2) & 0xff);
         }

       const unsigned char *data3 = (p->data()+_textlength+4+goCBRefLength+talLength);
       unsigned datsetLength=0;
         if((*data3 & 0xff) == datsetTag){
           data3++;
           datsetLength = ((*data3) & 0xff);
         }

       const unsigned char *data4 = (p->data()+_textlength+6+goCBRefLength+talLength+datsetLength);
```

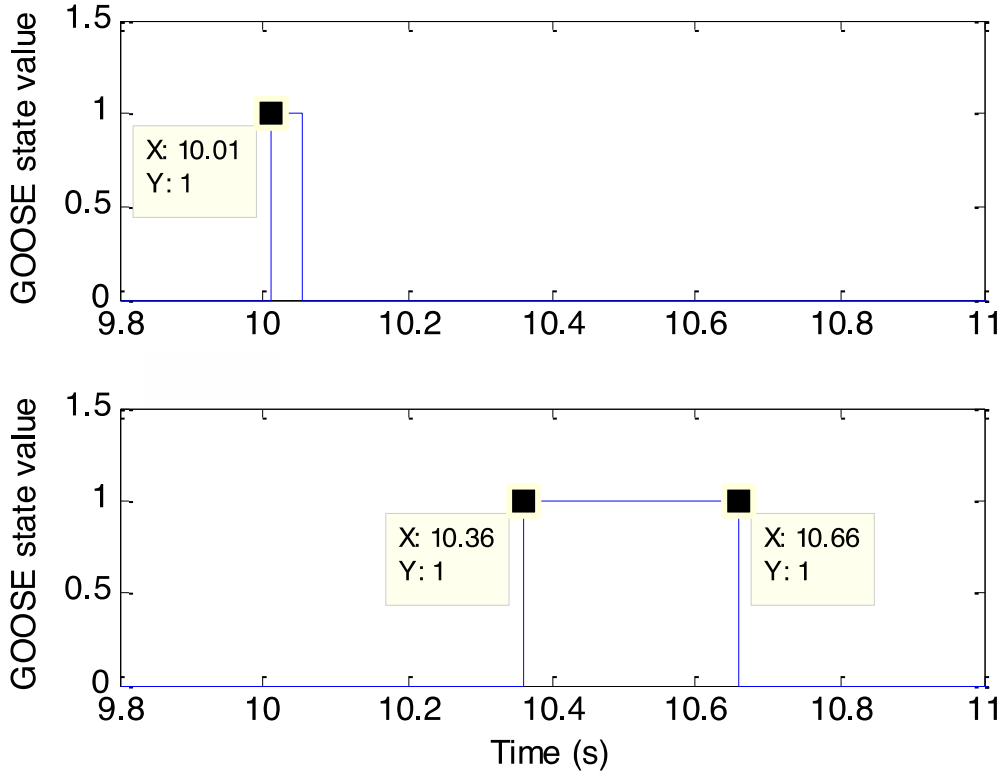
Composing new elements : Random Delay Element

- An element that applies random delay time on packets passing the router
- Assumes uniform distribution

```
require(clicklocal);

     FromDump(/home/charles/Documents/tracedumps/udpsample1.pcap, STOP true)
    -> Strip(14)-> CheckIPHeader()

    -> ip :: IPClassifier(udp,tcp,icmp,-)
    ip[0] -> tp :: Tee(2)
    tp[0] ->  q1::Queue(1000);


    tp[1] -> Queue(1000) -> RandomDelay(1,5)-> q1;

       q1 -> Unstrip(14) -> ToDump(/home/charles/Documents/tracedumps/drop.pcap);
    ip[1] -> Discard;
    ip[2] -> Discard;
    ip[3] -> Discard;

ClickyInfo(STYLE @import test-clicky.ccss);
```

Case Studies, Results & Evaluations

- **Case Study 1**: **HIL POTT application**
  - Emulator introduces delay of 0.35 seconds on GOOSE packets
    - ABB Relion 670 relay GOOSE publisher
    - Relay in OPAL-RT GOOSE subscriber
  - Zone 2 backup protection to clear fault, 0.3 seconds

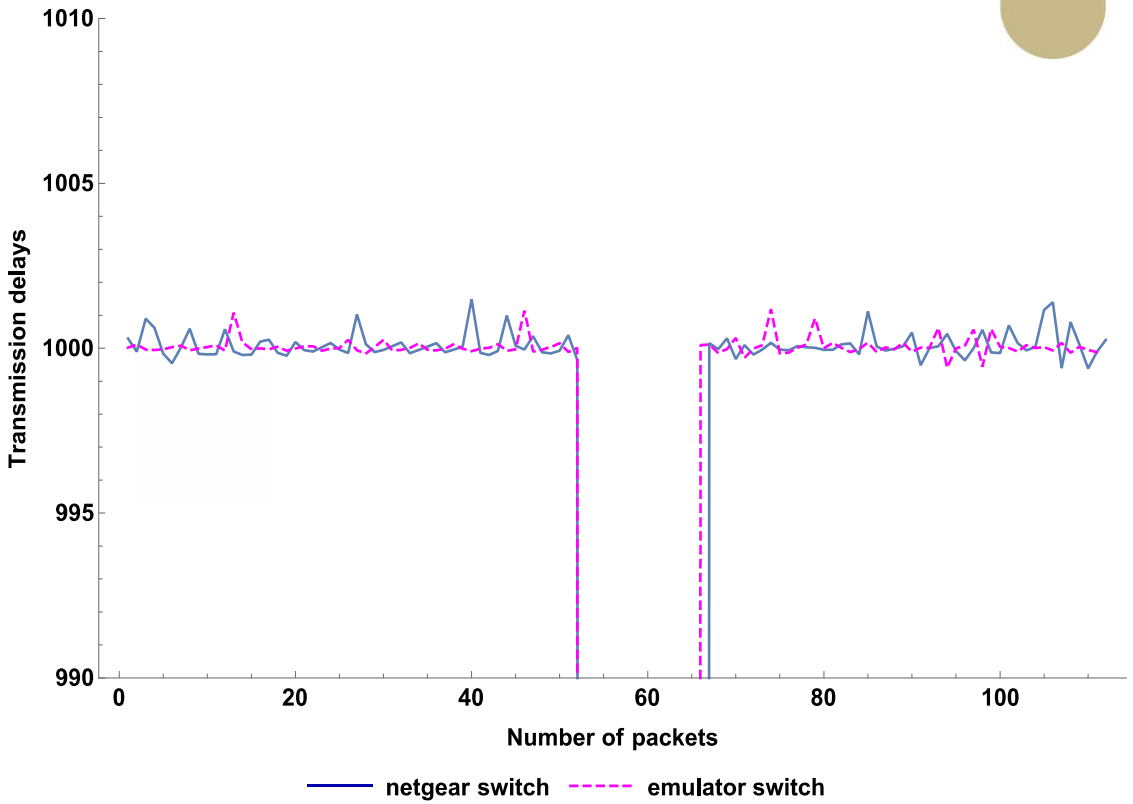


GOOSE data signals recorded by OPAL-RT

- Normal operation: detected fault cleared by the permissive trip signal
- Abnormal operation(delay introduced); POTT failed to clear fault in time
  - Backup protection was triggered at 0.3s to clear fault

Case Studies, Results & Evaluations

- **Case Study 2** :
  - Monitor and record GOOSE packet flow for a POTT–HIL test application setup. Two scenarios:
    - practical switch (Netgear switch)
    - Emulator switch
  - Wireshark to record timestamp difference of GOOSE packets



- State change behaviour as expected in both scenarios
- Stable retransmission of 1000msec in both scenarios

Conclusion

- Emulator can be used to change communication properties of delay, jitter, packet loss, packet corruption and bandwidth for WAN power system applications
  - Investigate effects of network congestion and packet loss on different protection algorithms
  - Model time delays and data loss to study effects of relay performance compromised by distributed generation and non-zero fault impedances.

Thanks for the attention