DESERT Underwater: tcl script

UNWis - Padova (Italy)

30th of January – 3rd of February 2023

Filippo Campagnaro, Roberto Francescon, Angela Soldà, Antonio Montanari, Michele Zorzi

filippo.campagnaro@unipd.it

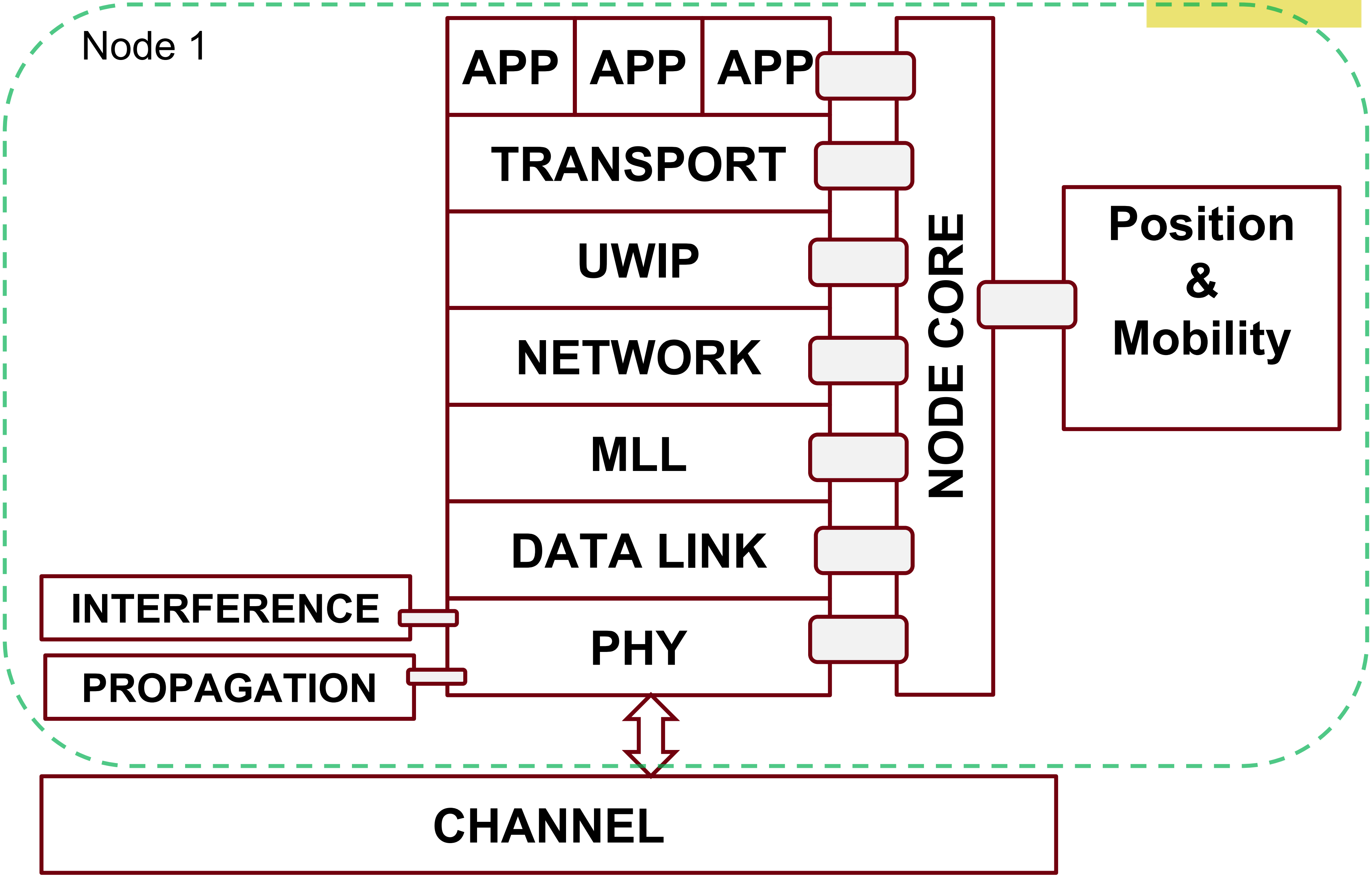
(O)TCL example

Libraries – random generator – scheduler – modules – node – modules connections

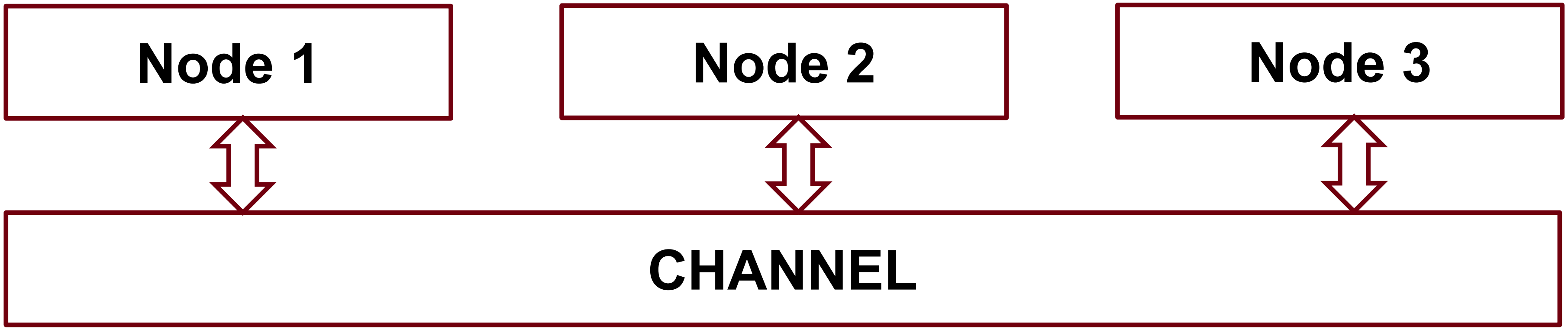
OTCL <source: wikipedia>

- object oriented extension of Tcl
- Tcl (pronounced "tickle) is a high-level, general-purpose, interpreted, dynamic programming language. Tcl casts everything into the mold of a command, even programming constructs like variable assignment and procedure definition.
- It is commonly used embedded into C applications, for rapid prototyping, scripted applications, and testing.
- DESERT Tcl scripts create and bind the modules and structures written in C++

DESERT Underwater stack



DESERT Underwater stack



TCL structure

- Load libraries, in order
- Set parameters (hardcoded in the script or passed from bash)
- Set random generator
- Create a node with all modules
- Set ARP table with MLL
- Set nodes position
- Create routing tables, if any
- Set the simulation events and run the simulation
- Finish the simulation and get the results

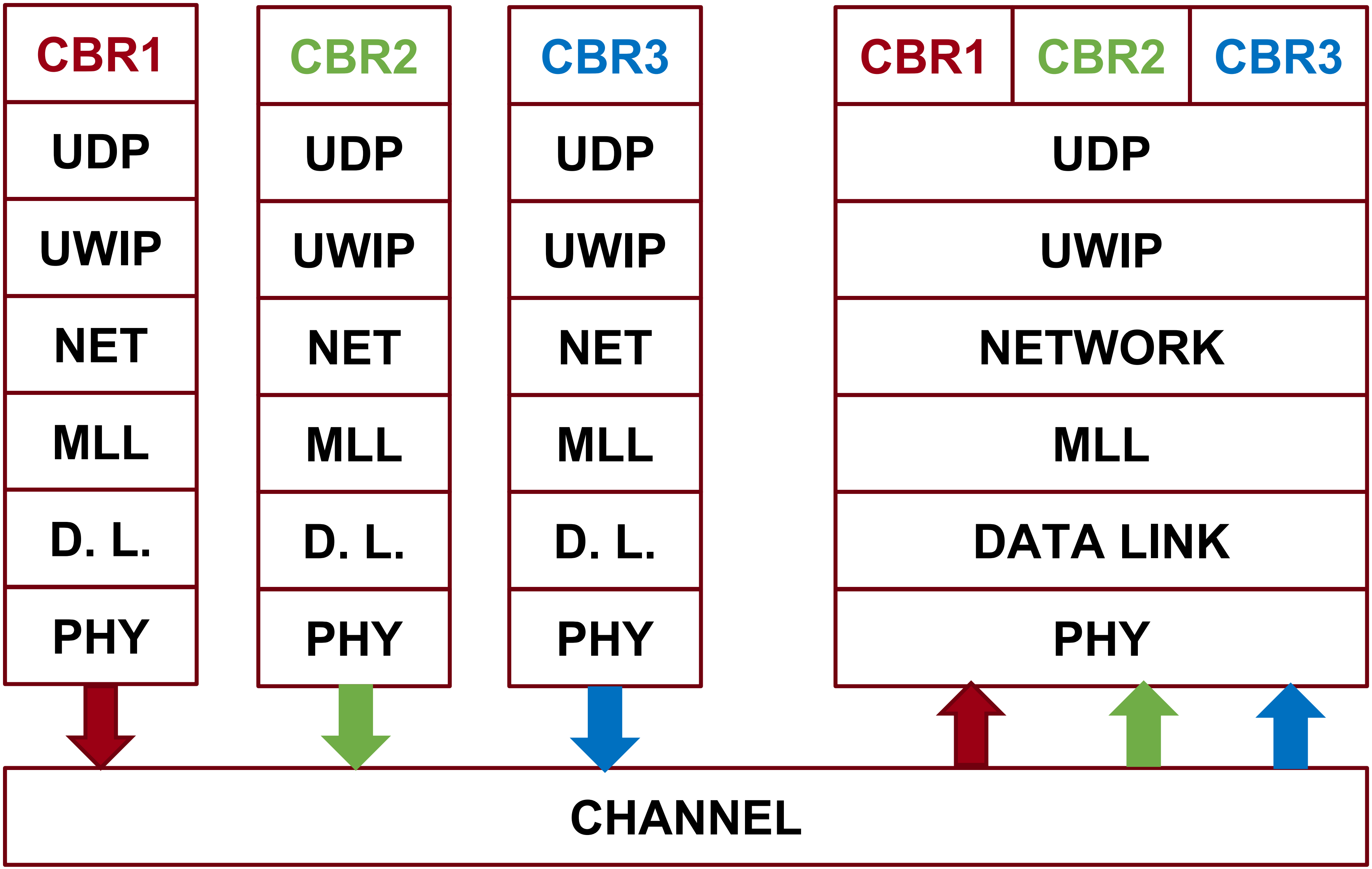
TCL example: test_uwcbr.tcl

- Nodes sending cbr data (fixed period, fixed size)
- A unique receiver (sink)

- To run the test, go with your terminal inside the *samples/desert_samples/Application* folder and type:

  ns test_uwcbr.tcl

test_uwcbr.tcl protocol stack



Load libraries, in order

load libMiracle.so
load libMiracleBasicMovement.so
load libmphy.so
load libmmac.so
load libUwmStd.so

Miracle

load libuwcsmaaloha.so
load libuwip.so
load libuwstaticrouting.so
load libuwmll.so
load libuwudp.so
load libuwcbr.so

DESERT

Create the simulator and use miracle

```
set ns [new Simulator]
$ns use-Miracle
```

Set parameters

opt is an associative array

```
set opt(string1,string2) 5

set opt(nn)                    2.0 ;# Number of Nodes
set opt(starttime)             1
set opt(stoptime)              100000
set opt(txduration) [expr $opt(stoptime) - $opt(starttime)]
set opt(freq)                  25000.0
set opt(bw)                    5000.0
set opt(ack_mode)              "setNoAckMode“
```

Set parameters – bash params

```
set opt(bash_parameters) 0; #1 for activate bash params
if {$opt(bash_parameters)} {
	if {$argc != 2} {
		puts "The script requires two inputs
	} else {
		set opt(pktsize)	[lindex $argv 0]
		set opt(cbr_period)	[lindex $argv 1]
	}
} else {
	set opt(pktsize)	125; #bytes
	set opt(cbr_period) 60; #seconds
}
```

Set random generator

```
set opt(rngstream) 1;  # better to pass it via bash

global defaultRNG

for {set k 0} {$k < $opt(rngstream)} {incr k} {
        $defaultRNG next-substream
}
```

Set common objects

```
set channel [new Module/UnderwaterChannel]
set propagation [new MPropagation/Underwater]
set data_mask [new MSpectralMask/Rect]
$data_mask setFreq       $opt(freq)
$data_mask setBandwidth  $opt(bw)
```

Common modules config - bind

```
Module/UW/CBR set packetSize_  $opt(pktsize)
Module/UW/CBR set period_          $opt(cbr_period)
Module/UW/CBR set PoissonTraffic_  1

Module/MPhy/BPSK  set BitRate_   $opt(bitrate)
Module/MPhy/BPSK  set TxPower_ $opt(txpower)
```

Create a node: create the modules

```
proc createNode { id } {
global <variables list> propagation
set node($id) [$ns create-M_Node $opt(tracefile) $opt(cltracefile)]
set cbr($id)  [new Module/UW/CBR]
set udp($id)  [new Module/UW/UDP]
set ipr($id)  [new Module/UW/StaticRouting]
set ipif($id) [new Module/UW/IP]
set mll($id)  [new Module/UW/MLL]
 set mac($id)  [new Module/UW/CSMA_ALOHA]
set phy($id)  [new Module/MPhy/BPSK]
…
}
```

Create a node: add the modules to the node

$node addModule <layer> <module> <cl_trace> <tag>

- layer: number of the layer

- module: object with the layer

- cl_trace: number with the maximum depth of the crosslayer that can be traced: set to 0 (we do not use ns2 trace)

- tag: string identifier of the layer of the node

Create a node: add the modules to the node

```
proc createNode { id } {
…
$node($id) addModule 7 $cbr($id)   0  "CBR"
$node($id) addModule 6 $udp($id)   0  "UDP"
$node($id) addModule 5 $ipr($id)   0  "IPR"
$node($id) addModule 4 $ipif($id)  0  "IPF"
$node($id) addModule 3 $mll($id)   0  "MLL"
$node($id) addModule 2 $mac($id)   0  "MAC"
$node($id) addModule 1 $phy($id)   0  "PHY "
…
}
```

Create a node: connect the layers

$node setConnection <upper> <lower> <trace>

- upper: upper layer
- lower: lower layer
- trace: flag set to 0 if the packet will not be traced: set to 0 (we do not use ns2 trace)

Create a node: connect the layers

```
proc createNode { id } {
…
$node($id) setConnection $cbr($id)   $udp($id)     0
$node($id) setConnection $udp($id)   $ipr($id)     0
$node($id) setConnection $ipr($id)    $ipif($id)    0
$node($id) setConnection $ipif($id)   $mll($id)     0
$node($id) setConnection $mll($id)    $mac($id)     0
$node($id) setConnection $mac($id)    $phy($id)     0
$node($id) addToChannel  $channel    $phy($id)     0 …
}
```

Create a node: position, interference and other parameters

```
proc createNode { id } {
…
set tmp_ [expr ($id) + 1]
$ipif($id) addr $tmp_
set position($id) [new "Position/BM"]
$node($id) addPosition $position($id)
set interf_data($id) [new "MInterference/MIV"]
$interf_data($id) set maxinterval_ $opt(maxinterval)
$phy($id) setPropagation $propagation
$phy($id) setSpectralMask $data_mask
$phy($id) setInterference $interf_data($id)
}
```

Create the sink: one app and one port number per node

```
for {set cnt 0} {$cnt < $opt(nn)} {incr cnt} {
    set cbr_sink($cnt)  [new Module/UW/CBR]
}…
for { set cnt 0} {$cnt < $opt(nn)} {incr cnt} {
    $node_sink addModule 7 $cbr_sink($cnt) 0 "CBR"
}
for { set cnt 0} {$cnt < $opt(nn)} {incr cnt} {
    $node_sink setConnection $cbr_sink($cnt)  $udp_sink      0
} …
for { set cnt 0} {$cnt < $opt(nn)} {incr cnt} {
    set portnum_sink($cnt) [$udp_sink assignPort $cbr_sink($cnt)]
}
```

Create the nodes and connect them

```
for {set id 0} {$id < $opt(nn)} {incr id}  {
            createNode $id
}
proc connectNodes {id1} {
global ipif ipr portnum cbr cbr_sink ipif_sink portnum_sink ipr_sink
      $cbr($id1) set destAddr_ [$ipif_sink addr]
      $cbr($id1) set destPort_ $portnum_sink($id1)
      $cbr_sink($id1) set destAddr_ [$ipif($id1) addr]
      $cbr_sink($id1) set destPort_ $portnum($id1)
}
for {set id1 0} {$id1 < $opt(nn)} {incr id1}  {
            connectNodes $id1
}
```

Fill the ARP table

```
for {set id1 0} {$id1 < $opt(nn)} {incr id1}  {
    for {set id2 0} {$id2 < $opt(nn)} {incr id2}  {
        $mll($id1) addentry [$ipif($id2) addr] [$mac($id2) addr]
    }
    $mll($id1) addentry [$ipif_sink addr] [ $mac_sink addr]
    $mll_sink addentry [$ipif($id1) addr] [ $mac($id1) addr]}
```

Set position

```
$position(0) setX_ 0
$position(0) setY_ 0
$position(0) setZ_ -1000
$position(1) setX_ 500
$position(1) setY_ 500
$position(1) setZ_ -1000
```

Routing table and start simulation

```
$ipr(0) addRoute [$ipif_sink addr] [$ipif(1) addr]
$ipr(1) addRoute [$ipif_sink addr] [$ipif_sink addr]

for {set id1 0} {$id1 < $opt(nn)} {incr id1}  {
        $ns at $opt(starttime)    "$cbr($id1) start"
        $ns at $opt(stoptime)     "$cbr($id1) stop"
}
```

Finish the simulation

```
proc finish {} {
#the procedure to call at the end of the simulation
…
}
$ns at [expr $opt(stoptime) + 250.0]  "finish; $ns halt"
$ns run
```

Run the simulation

The simulation summary is printed to standard output

```
------------------------------------------------------------
Simulation summary
number of nodes   : 2.0
packet size       : 125 byte
cbr period        : 60 s
number of nodes   : 2.0
simulation length: 99999 s
tx frequency      : 25000.0 Hz
tx bandwidth      : 5000.0 Hz
bitrate           : 4800.0 bps
------------------------------------------------------------
cbr_sink(0) throughput                    : 16.811553
cbr_sink(1) throughput                    : 16.147135
Mean Throughput            : 16.479343999999998
Sent Packets               : 3309.0
Received Packets           : 3293.0
Packet Delivery Ratio      : 99.516470232698694
IP Pkt Header Size         : 2
UDP Header Size            : 2
CBR Header Size            : 12
```

How can I get the results, analyze and plot them?

Get the results: log parsing

1) Redirect the simulation standard output to file

ns test_uwcbr.tcl > new_file.txt

creates a new file, deletes new_file.txt if already exists

ns test_uwcbr.tcl >> existing_file.txt

Attaches the output to an existing file

2) Parse the file reading the lines and the columns you are interested

Log parsing with bash

Lines

grep "Mean Throughput" existing_file.txt > temp.txt

Columns

awk -F ":" 'BEGIN { } { } END {printf("%.2f\n",$2)}’ temp.txt > out.csv

All in one line (using pipe |)

grep "Mean Throughput" out.txt | awk -F ":" 'BEGIN { } { } END {printf("%.2f\n",$2)}’ > out.csv

out.csv and parsing alternatives

out.csv now contains one single column of data, that you can open with excel, libreoffice, matlab, gnuplot, python or whatever other tool you are familiar with, and do your data analysis

If you are already familiar with log parsing, e.g., with python, feel free to use that

Perform several simulation runs

Ex1: several simulation runs with different rng sequence

For loop in bash script:

- Create a file .sh with #!/bin/bash as first line and

```
for i in {0..100}
do
    ns test_uwcbr.tcl 125 60 $i >> log_file.txt
done
```

SIGNET

Perform several simulation runs

Ex2: several simulation runs with different rng sequence for generation period = 5, 10, 20, 30, 40, 50, and 60 seconds

Suggestion: you need to modify the tcl

Ex3: plot throughput average and CI vs number of nodes

Ex4: do the same for PER