ns3 for beginners

A guide towards running Hyrax simulations

Joaquim M. E. Silva
jqmmes@gmail.com

December 2015

Contents

Chapter 1

Introduction

1.1 Installing

1.2 Configurations

1.3 Running a simulation

Chapter 2

Simulations

2.1 Overlay Simulation

2.1.1 Objectives

2.1.2 Parameters

2.1.3 Examples

2.2 Technologies Experiment

2.2.1 Objectives

2.2.2 Parameters

Running:

```
./waf --run="scratch/Experiment/Experiment --Nodes=1 --Servers=1 --Scenario=3 --Seed=$RANDOM --ExclusiveServers"
```

Parameters:

Nodes: Number of Nodes to be used in the simulation

Servers: Number of Servers to be used in the simulation

Scenario: Scenario to run

- 1: 1 Server + AP + n Nodes
- 2: AP + m Mobile Servers + n Nodes (m¡=n)
- 3: AP + TDLS + m Mobile Servers + n Nodes (m¡=n)
- 41: WD + GO as Server + n Nodes
- 42: WD + GO + m Mobile Servers + n Nodes (m¡=n)
- 43: WD + m Mobiles Servers + n Nodes (m¡=n) No groups formed in the beggining
- 51: WD + Legacy AP as Server + n Nodes
- 52: WD + Legacy AP + m Mobile Servers + n Nodes (m¡=n)
- 6: WD + GO + TDLS + m Mobile Servers + n Nodes (m¡=n)

FileSize: File Size to be shared

Debug: Debug socket callbacks

ShowPackets: Show every packet received

ShowData: Show Send/Receive instead of the time a transfer took

Seed: Seed to be used

ExclusiveServers: Use Exclusive Server. (Server Don't act as Client)

SegmentSize: TCP Socket Segment Size

2.2.3 Examples

2.3 CMU Review App Simulation

2.3.1 Objectives

2.3.2 Parameters

2.3.3 Examples

Chapter 3

Post-Processing

3.1 Overlay Simulation

3.2 Technologies Experiment

3.3 CMU Review App Simulation

Chapter 4

Code

4.1 Network Configurations

4.2 *VirtualDiscovery*

Public Methods:

```
void VirtualDiscovery::add(Ipv4Address ip, uint16_t port)
tuple<Ipv4Address,uint16_t> VirtualDiscovery::discover(void)
vector<tuple<Ipv4Address,uint16_t>> VirtualDiscovery::getAll(void)
uint32_t VirtualDiscovery::GetN(void)
void VirtualDiscovery::remove(Ipv4Address ip, uint16_t port)
```

4.3 TDLS

Public Methods:

```
void SendTDLS(Ipv4Address ip, uint16_t port, std::string message)
```

Algorithms:

Algorithm 1: TDLS ns3 Algorithm - Client

```
Data: $message$ - Message to be sent; $socket$ - TDLS (using Wi-Fi Ad-hoc) socket
Result: A message is sent using TDLS or regular Wifi as fallback
Input  : $timeout$ - duration until CheckTDLS fallback occurs
Output: nothing
Function SendTDLS($socket, message$) /* Algorithm to Send a message with TDLS */
    if $ActiveTDLSCons < MAX$ then
        $socket \rightarrow connect(ServerIp)$;
        TDLSData[$socket$] $\leftarrow$ $socket$ ;
        TDLSData[$message$] $\leftarrow$ $message$ ;
        TDLSData[$delivered$] $\leftarrow$ false ;
        $ActiveTDLSCons + +$ ;
        Schedule(CheckTDLS($socket, message$), $timeout$);
    else
        $RegularSocket \rightarrow connect(ServerIp)$;
        $RegularSocket \rightarrow send(message)$;
    end
end
Function CheckTDLS($socket, message$)
    if TDLSData[$socket$] $= socket \wedge$ TDLSData[$delivered$] $= false$ then
        $RegularSocket \rightarrow connect(ServerIp)$;
        $RegularSocket \rightarrow send(message)$;
    end
    /* Deletes Hashmap entry */
    DeleteEntry(TDLSData[$socket$]) ;
end
Callback ConnectSuccess ($socket$) /* Callback called if $socket \rightarrow connect(ServerIp)$
succeeds */
    TDLSData[$delivered$] $\leftarrow$ true ;
end
```

Algorithm 2: TDLS ns3 Algorithm - Server

```
Data: $socket$ - TDLS (using Wi-Fi Ad-hoc) socket
Result: A message is received using TDLS or regular Wifi as fallback
Input  : $MAX$ - Maximum number of simultaneous TDLS sockets opened
Output: nothing
Function TDLSAccept($ListenSocket$)
    if $ActiveTDLSCons < MAX$ then
        $ActiveTDLSCons + +$ ;
        $socket \rightarrow setAcceptCallback$(SecondPhaseAccept) ;
        return true;
    end
    return false;
end
Callback SecondPhaseAccept($socket$)
    $ConnectedTDLS \leftarrow socket$;
end
```

Algorithm 3: TDLS ns3 Algorithm - Closing Socket

```
Function CloseSocket($socket$)
    $socket \rightarrow Close()$ ;
    if $ConnectedTDLS = socket$ then
        $ActiveTDLSCons - -$ ;
        $ConnectedTDLS \leftarrow Null$ ;
    end
end
```

4.4 Wifi-Direct

4.5 Mobility

Chapter 5

Advanced

5.1 Developing a new simulation from scratch

5.2 Tracing

5.3 Parallel Execution

5.4 Direct Code Execution