SIMULACIÓN DE PROTOCOLOS DE ENRUTAMIENTO PARA REDES MÓVILES AD-HOC MEDIANTE HERRRAMIENTA DE SIMULACIÓN NS-3

Services in ns-3

Outline

1. Services in ns-3
   • Sockets
   • UDP
   • TCP
   • Bulk Application
   • FTP
   • On off Application

2. Exercises

3. References
Services in ns-3. Socket

- Socket definition: the combination of an IP address and a port number [RFC 793]
  - Socket pair consists of the client IP address, client port number, server IP address and server port number
  - Socket pair identifies the two end points that identifies a TCP connection in Internet.
- A Ns-3 socket is a dynamic object often created by Applications to communicate between Nodes.
- It is based on BSD socket API (Berkeley sockets).
  - Berkeley sockets (or BSD sockets) is a computing library with an programming interface (API) for internet sockets and Unix domain sockets used for inter-process communication.
  - It became the standard interface for connecting to the Internet.
  - The BSD sockets API is written in the C.
  - Some functions: socket(), bind(), listen(), connect(),..
Services in ns-3. Socket Example

//Receiver socket on n1
Ptr<Socket> recvSink = Socket::CreateSocket (n.Get (1),
  "ns3::UdpSocketFactory");
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 4477);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

//Sender socket on n0
Ptr<Socket> source = Socket::CreateSocket (n.Get (0),
  "ns3::UdpSocketFactory");
InetSocketAddress remote = InetSocketAddress (i.GetAddress (1),
  4477);
source->Connect (remote);
//Schedule SendPacket
Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
  Seconds (1.0), &SendPacket,
  source, packetSize, packetCount,
  interPacketInterval);

Services in ns-3. UDP I

- **Ns-3** implements User Datagram Protocol described in RFC 768.
  - It implements a connectionless, unreliable datagram packet service. Packets may be reordered or duplicated before they arrive. UDP generates and checks checksums to catch transmission errors.

- **UDP client** sends packets to a UDP server

- **A UDP client.** Sends UDP packet carrying sequence number and time stamp in their payloads.
  - MaxPackets, Interval, RemoteAddress, RemotePort, Packet Size, StartTime, StopTime

- **A UDP server.** Receives UDP packets from a remote host.
  - Port, PacketWindowSize, StartTime, StopTime
Services in ns-3. UDP Server configuration

**Server configuration**

```cpp
#include "ns3/applications-module.h"

// Create one UDP Server application on node one.

uint16_t port = 4000;
UdpServerHelper server (port);

ApplicationContainer apps = server.Install (n.Get (1));

apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));
```

Services in ns-3. UDP Client Configuration

**Client configuration**

```cpp
// Create one UDP Client application to send UDP datagrams from
// node zero to node one

uint32_t MaxPacketSize = 1024;
Time interPacketInterval = Seconds (0.05);
uint32_t maxPacketCount = 320;
UdpClientHelper client (serverAddress, port);
client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
client.SetAttribute ("Interval", TimeValue (interPacketInterval));
client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));

apps = client.Install (n.Get (0));

apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));
```
Services in ns-3. TCP I

- Ns-3 implements Transport Control Protocol described in RFC 793
- TCP Reno, New Reno, Tahoe, Westwood, Westwood ++
- TCP attributes can be tuned by ns3::TcpSocket Class

Reference
• This class exists solely for hosting TcpSocket attributes
• SndBufSize: maximum transmit buffer size (bytes)
• RcvBufSize: maximum receive buffer size (bytes)
• SegmentSize: TCP maximum segment size in bytes
• SlowStartThreshold: TCP slow start threshold (bytes)
• InitialCwnd: TCP initial congestion window size (segments)
• ConnTimeout: TCP retransmission timeout when opening connection (seconds)

Services in ns-3. TCP II

• ConnCount: Number of connection attempts (SYN retransmissions) before returning failure
• DelAckTimeout: Timeout value for TCP delayed acks, in seconds
• DelAckCount: Number of packets to wait before sending a TCP ack
• TcpNoDelay: Set to true to disable Nagle’s algorithm
• PersistTimeout: Persist timeout to probe for rx window

Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (tcp_adu_size));
Services in ns-3. Variants TCP

- TCP Tahoe. This class contains the Tahoe implementation of TCP.
  - Tahoe is not officially published in RFC.
  - In summary, Tahoe has slow start, congestion avoidance and fast retransmit.
  
  ```
  Config::SetDefault("ns3::TcpL4Protocol::SocketType",
  TypeIdValue (TcpTahoe::GetTypeId()));
  ```

- TCP Reno. This class contains the Reno implementation of TCP, according to RFC2581.
  
  ```
  Config::SetDefault("ns3::TcpL4Protocol::SocketType",
  TypeIdValue (TcpReno::GetTypeId()));
  ```

- TCP NewReno. This class contains the NewReno implementation of TCP, as of RFC2582.
  
  ```
  Config::SetDefault("ns3::TcpL4Protocol::SocketType",
  TypeIdValue (TcpNewReno::GetTypeId()));
  ```

- TCP Westwood and Westwood ++.
  - They are the two modifications of TCP Reno that employ the AIAD (Additive Increase/Adaptive Decrease) congestion control paradigm.
  - They try to estimate the network's bandwidth using the received ACKs and use the estimated bandwidth value to adjust the congestion window after a loss occurs.
  - The difference between Westwood and Westwood+ is their bandwidth sampling interval. Westwood samples the bandwidth every ACK reception while Westwood+ performs the sampling every RTT.

  ```
  Config::SetDefault("ns3::TcpL4Protocol::SocketType",
  TypeIdValue (TcpWestwood::GetTypeId()));
  ```

- TCP SACK, TCP Vegas and SCPS-TP Under development
Services in ns-3. BulkSendApplication I

- This traffic generator simply sends data as fast as possible up to MaxBytes or until the application is stopped if MaxBytes is zero.
- Once the lower layer send buffer is filled, it waits until space is free to send more data, essentially keeping a constant flow of data.
- TCP sockets can be used, but UDP sockets cannot be used.
- Attributes:
  - **SendSize**: The amount of data to send each time.
  - **Remote**: The address of the destination
  - **MaxBytes**: The total number of bytes to send. Once these bytes are sent, no data is sent again. The value zero means that there is no limit.
  - **Protocol**: The type of protocol to use.

```cpp
// Create a BulkSendApplication and install it on node 0
uint16_t port = 9;  // well-known echo port number
BulkSendHelper source ("ns3::TcpSocketFactory",
                      InetSocketAddress (i.GetAddress (1), port));
source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
ApplicationContainer sourceApps = source.Install (nodes.Get (0));
sourceApps.Start (Seconds (0.0));
sourceApps.Stop (Seconds (10.0));

// Create a PacketSinkApplication and install it on node 1
PacketSinkHelper sink ("ns3::TcpSocketFactory",
                       InetSocketAddress (Ipv4Address::GetAny (),
                       port));
ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
sinkApps.Start (Seconds (0.0));
sinkApps.Stop (Seconds (10.0));
```
Services in ns-3. FTP Application

- There is not a FTP Application but it can be generated by BulkSendApplication

```cpp
BulkSendHelper ftp("ns3::TcpSocketFactory", Address());
ftp.SetAttribute ("Remote", remoteAddress);
ftp.SetAttribute ("SendSize", UintegerValue (tcp_adu_size));
ftp.SetAttribute ("MaxBytes", UintegerValue (int(data_mbytes*1000000)));

ApplicationContainer sourceApp = ftp.Install (sources.Get(i));
sourceApp.Start (Seconds (start_time*i));
sourceApp.Stop (Seconds (stop_time - 3));
```

Services in ns-3. On Off Application

- Generate traffic to a single destination according to an On Off pattern.
  - This traffic generator follows an On/Off pattern: after `Application::StartApplication` is called, "On" and "Off" states alternate.
  - The duration of each of these states is determined with the onTime and the offTime random variables.
    - During the "Off" state, no traffic is generated.
    - During the "On" state, CBR traffic is generated.
    - This CBR traffic is characterized by the specified "data rate" and "packet size".
Services in ns-3. Example On Off Application I

// On Off Application - UDP Protocol

#include "ns3/applications-module.h"

// Create a UDP packet sink to receive packets.
uint16_t port = 50000;
Address sinkAddress (InetSocketAddress (interfaces.GetAddress (0), port));
PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory", sinkAddress);

ApplicationContainer serverApps = packetSinkHelper.Install
(nodes.Get(0));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (100.0));

Services in ns-3. Example On Off Application II

// On Off Application

Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue 
(137));
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue 
("140kb/s"));

OnOffHelper onOffHelper ("ns3::UdpSocketFactory", Address ());
onOffHelper.SetAttribute ("OnTime", StringValue 
("ns3::ConstantRandomVariable[Constant=1]"));
onOffHelper.SetAttribute ("OffTime", StringValue 
("ns3::ConstantRandomVariable[Constant=0]"));
// On Off Application

ApplicationContainer sourceon_off;
AddressValue remoteAddress (InetSocketAddress (interfaces.GetAddress (0), port));

onOffHelper.SetAttribute ("Remote", remoteAddress);
sourceon_off.Add (onOffHelper.Install (nodes.Get(1)));

sourceon_off.Start (Seconds (5.0));
sourceon_off.Stop (Seconds (50.0));

---

Model description in ns-3. References


- Ns-3 tutorial and manual
Exercises

Services in ns-3. Exercise 1

- Model a 2 nodes connected by a point to point link of 1Mbps and 1ms delay
- On off Aplication over UDP
Services in ns-3. Exercise 2

- Model a 2 nodes connected by a point to point link of 1Mbps and 1ms delay
- Ftp Application
- TCP New Reno