A tutorial on the implementation of TCP in ns-3

June 2017
Objectives of this tutorial

- To provide an overview of TCP implementation in ns-3
  - Learn about the different implementations of TCP in ns-3
  - Understand the architecture of natively implemented TCP in ns-3
  - Walk through a simple TCP example
  - Introduce how to write new TCP extensions
  - Learn about writing test cases for new extensions
  - Learn about the ongoing work related to TCP in ns-3
Outline of the presentation

• TCP implementations in ns-3
• History of ns-3 TCP
• Algorithms for congestion control and loss recovery
• Implementation of ns-3 TCP
• Demonstration of example programs
• How to add a new TCP extension in ns-3?
• Sample test cases for new TCP extension
• Overview of the ongoing work
• Review
TCP implementations in ns-3
TCP implementations in ns-3

• Presently there are following implementations of TCP available for ns-3:
  – a native implementation of TCP in ns-3 (ns-3 TCP)
  – support for Network Simulation Cradle (NSC)
  – support for Direct Code Execution (DCE)
  – others (e.g., combining virtual machines with ns-3)

• ns-3 TCP model supports:
  – a full bidirectional TCP
  – connection setup
  – connection teardown
History of ns-3 TCP
History of ns-3 TCP

• Until ns-3.10
  – it was a port of TCP model from GTNetS (Georgia Tech Network Simulator)
• For ns-3.10
  – it was substantially rewritten by Adriam Tam in 2011
• For ns-3.25
  – the module was refactored as a part of GSoC 2015 project by Natale Patriciello
  – one of the major changes involved how congestion control algorithms are implemented (more details to follow)
  – other notable change was about automating the tests
  – Target is to align the implementation with that of Linux
Algorithms for congestion control and loss recovery
Congestion control algorithms

- NewReno \textit{(default)}
- Westwood, Westwood+
- Hybla
- HighSpeed
- Vegas
- Scalable
- Veno
- Binary Increase Congestion Control (BIC)
- Yet another HighSpeed TCP (YeAH)
- Illinois
- H-TCP
- Low Extra Delay Background Transport (LEDBAT)
Loss detection and recovery algorithms

- Fast retransmit
- Fast recovery
- Selective Acknowledgements (SACK)
Implementation of ns-3 TCP
TCP implementation in ns-3

- Source code can be found at: src/internet/model/
  - tcp-header.{h,cc}
  - tcp-socket.{h,cc}
  - tcp-socket-base.{h,cc}
  - tcp-socket-factory-impl.{h,cc}
  - tcp-l4-protocol.{h,cc}
  - tcp-congestion-ops.{h,cc}
  - ...

ns-3 Training, June 2017
TcpHeader class

• This class implements the TCP header and contains:
  – port numbers
  – sequence numbers
  – acknowledgment numbers
  – flags
  – …

• It also contains:
  – setters and getters
  – methods for serialization
  – and deserialization
 TcpSocket class

• This class:
  – is an abstract base class for all TcpSockets
  – contains TcpSocket attributes that can be reused across different implementations.

• Examples of such attributes include:
  – SndBufSize
  – RcvBufSize
  – SegmentSize
  – InitialCwnd
  – DelAckCount
  – DelAckTimeout
  – ...

ns-3 Training, June 2017
TcpSocketFactory class

- This class:
  - is an abstract base class
  - defines API for TCP sockets
  - contains global default variables to initialize new sockets
TcpSocketFactoryImpl class

• This class:
  – is an implementation of socket factory for ns-3 TCP
  – creates sockets of type TcpSocketBase
TcpSocketBase class

- This class:
  - is a base class for the implementation of TCP stream socket
  - contains essential components of TCP and provides a socket interface for upper layers to call
- Examples of components include:
  - Connection orientation
  - Sliding window mechanism
  - Fast retransmit
  - Fast recovery
  - Enable/disable window scaling, timestamps
  - Congestion state machine
  - Congestion control interface
TcpSocketState class

• This class:
  – records the congestion state of a connection
  – saves the information that is passed between the socket and the congestion control algorithms

• Examples of such information include:
  – the current value of congestion window
  – the current congestion state (CA_OPEN, CA_RECOVERY, etc)
  – the current value of slow start threshold
  – Last sequence number acknowledged
  – Next sequence number to be transmitted
  – …
TcpCongestionOps class

• This class:
  – is an abstract class for congestion control
  – provides an interface between the main socket code and congestion control; variables are stored in TcpSocketState
  – inspired by the design in Linux

• Some methods implemented in this class include:
  – GetSsThresh (Ptr<TcpSocketState>, uint32_t)
  – IncreaseWindow (Ptr<TcpSocketState>, uint32_t)
  – CongestionStateSet (Ptr<TcpSocketState>, TcpSocketState::TcpCongState_t)
  – PktsAcked (Ptr<TcpSocketState>, uint32_t, Time)
TcpCongestionOps class
Demonstration of example programs: examples/tcp/
How to add a new TCP extension in ns-3?
Steps to add a new TCP extension in ns-3

1. Create tcp-new.{h,cc} files for the new TCP extension in src/internet/model/

2. Create a class for new TCP extension, which can be inherited from TcpCongestionOps (or TcpNewReno as shown before)

3. Some of the following methods may require a specific implementation for the new TCP extension:
   - GetSsThresh
   - IncreaseWindow
   - PktsAcked

4. Make necessary modifications in src/internet/wscript

5. Configure and build ns-3 (resolve errors, if any)

6. Setup an example program for this extension (or use an existing one).

7. Write tests and update the documentation in src/internet/doc/tcp.rst
Sample test cases for new TCP extension
Sample test cases for new TCP extension

1. Some of the following test cases are very commonly used across different TCP extensions
   - CwndIncrementTest
   - CwndDecrementTest

2. Some TCP extensions need exclusive test cases, such as in the case of LEDBAT
   - LEDBAT should be same as NewReno during Slow Start
   - LEDBAT should be same as NewReno when timestamps are disabled

3. Individual algorithms can be tested too
   - test the working of slow start algorithm
   - test the working of window scaling algorithm
Overview of the ongoing work
Ongoing work

- MPTCP model in ns-3
- DCTCP, TCP Prague models in ns-3
- TCP BBR model in ns-3
- TCP Evaluation Suite for ns-3
Review
Review

• Different TCP implementations can be used with ns-3
• ns-3 TCP has been recently refactored
• The new architecture is simple and user friendly for
  – adding new congestion control algorithms
  – testing them
• Scope to develop more extensions
  – e.g., TCP extensions for Data Center Networks
Thank you!