Implementation of Stateless Transport Protocols in ns-3

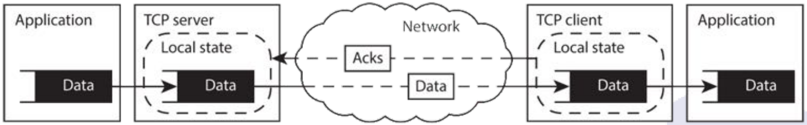
D.Ju. Chalyy
e-mail: chaly@uniyar.ac.ru

P.G. Demidov Yaroslavl State University,
Yaroslavl, Russia

May 14, 2015

P.G. DEMIDOV
YAROSLAVL STATE UNIVERSITY

TCP Architecture



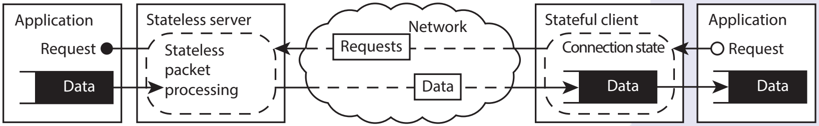
- Originally from 1980s: in-order delivery, loss recovery, congestion control;
- Many major improvements: new congestion control algorithms etc.;
- Tons of minor improvements: bug fixes, specific corrections etc.

Stateless architecture

- **Store no application state on your servers**

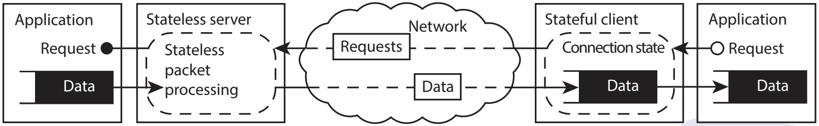
AWS Tips I Wish I'd Known Before I Started

(https://wblinks.com/notes/aws-tips-i-wish-id-known-before-i-started/)



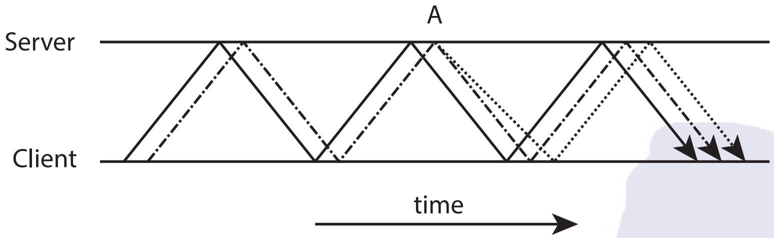
Must retain in-order delivery, loss recovery, congestion control.

Trickles protocol



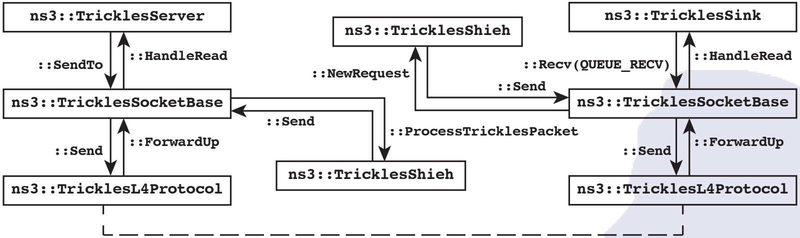
- Developed in Cornell in 2000s:
  A. Shieh, A.C. Myers, E.G. Sirer, "A Stateless Approach to Connection-Oriented Protocols", ACM Trans. Comput. Syst., Vol. 26, No 3, Sept., 2008, 50 p.
- Sample implementation exists for Linux kernel (last commit in 2006)

Trickles congestion control



- Trickle management: split, continue, terminate;
- Follows Reno congestion control algorithm: slow start, congestion avoidance, fast retransmit.

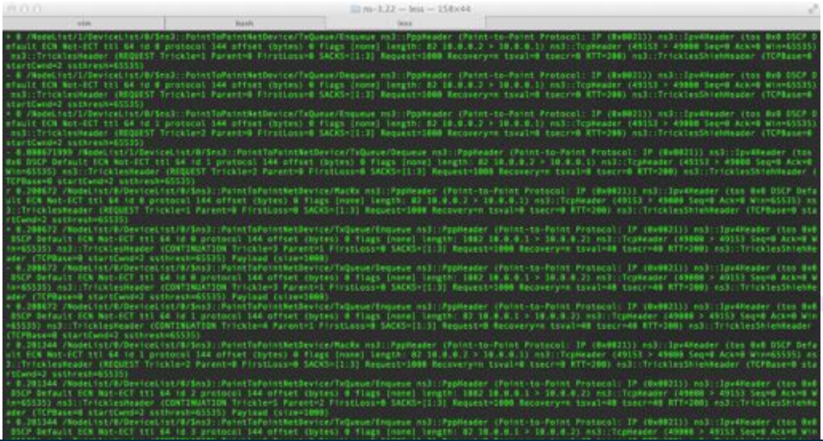
Overview of Trickles model in ns-3



Source code is available at *https://github.com/dchaly/stateless*

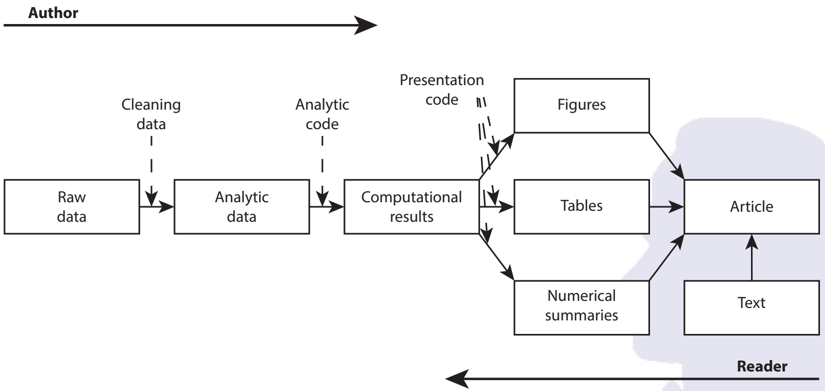
Stages of modeling in ns-3

- Build a model of your own protocol or use existing ones.
- Describe an experiment using ns-3 classes.
- Compile, run and get a trace file:

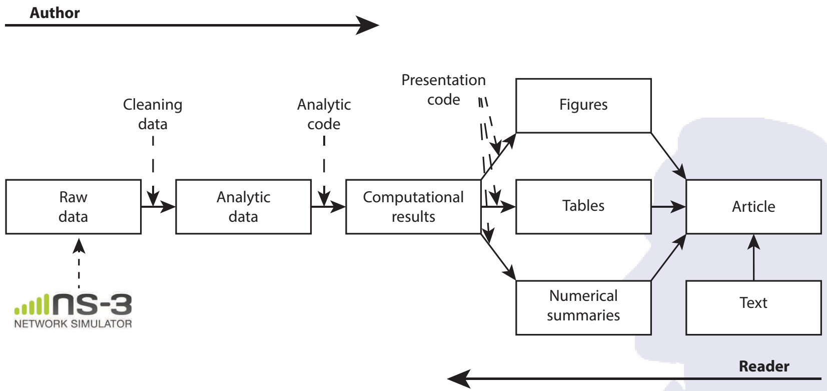


```
+ 0 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP D
efault ECN Not-ECT ttl 64 id 0 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0 Win=65535)
 ns3::TricklesHeader (REQUEST Trickle=1 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (TCPBase=0
startCwnd=2 ssthresh=65535)
- 0 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP D
efault ECN Not-ECT ttl 64 id 0 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0 Win=65535)
 ns3::TricklesHeader (REQUEST Trickle=1 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (TCPBase=0
startCwnd=2 ssthresh=65535)
+ 0 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP D
efault ECN Not-ECT ttl 64 id 1 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0 Win=65535)
 ns3::TricklesHeader (REQUEST Trickle=2 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (TCPBase=0
startCwnd=2 ssthresh=65535)
- 0.000671999 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos
0x0 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0
Win=65535) ns3::TricklesHeader (REQUEST Trickle=2 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (
TCPBase=0 startCwnd=2 ssthresh=65535)
r 0.200672 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Defa
ult ECN Not-ECT ttl 64 id 0 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0 Win=65535) ns
3::TricklesHeader (REQUEST Trickle=1 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (TCPBase=0 sta
rtCwnd=2 ssthresh=65535)
+ 0.200672 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0
 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 144 offset (bytes) 0 flags [none] length: 1082 10.0.0.1 > 10.0.0.2) ns3::TcpHeader (49000 > 49153 Seq=0 Ack=0 W
in=65535) ns3::TricklesHeader (CONTINUATION Trickle=3 Parent=1 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=40 tsecr=40 RTT=200) ns3::TricklesShiehHe
ader (TCPBase=0 startCwnd=2 ssthresh=65535) Payload (size=1000)
- 0.200672 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0
 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 144 offset (bytes) 0 flags [none] length: 1082 10.0.0.1 > 10.0.0.2) ns3::TcpHeader (49000 > 49153 Seq=0 Ack=0 W
in=65535) ns3::TricklesHeader (CONTINUATION Trickle=3 Parent=1 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=40 tsecr=40 RTT=200) ns3::TricklesShiehHe
ader (TCPBase=0 startCwnd=2 ssthresh=65535) Payload (size=1000)
+ 0.200672 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0
 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.1 > 10.0.0.2) ns3::TcpHeader (49000 > 49153 Seq=0 Ack=0 Win
=65535) ns3::TricklesHeader (CONTINUATION Trickle=4 Parent=1 FirstLoss=0 SACKS=[1:3] Request=0 Recovery=n tsval=40 tsecr=40 RTT=200) ns3::TricklesShiehHeader
(TCPBase=0 startCwnd=2 ssthresh=65535)
r 0.201344 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Defa
ult ECN Not-ECT ttl 64 id 1 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.2 > 10.0.0.1) ns3::TcpHeader (49153 > 49000 Seq=0 Ack=0 Win=65535) ns
3::TricklesHeader (REQUEST Trickle=2 Parent=0 FirstLoss=0 SACKS=[1:3] Request=1000 Recovery=n tsval=0 tsecr=0 RTT=200) ns3::TricklesShiehHeader (TCPBase=0 sta
rtCwnd=2 ssthresh=65535)
+ 0.201344 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0
 DSCP Default ECN Not-ECT ttl 64 id 2 protocol 144 offset (bytes) 0 flags [none] length: 1082 10.0.0.1 > 10.0.0.2) ns3::TcpHeader (49000 > 49153 Seq=0 Ack=0 W
in=65535) ns3::TricklesHeader (CONTINUATION Trickle=5 Parent=2 FirstLoss=0 SACKS=[1:3] Request=2000 Recovery=n tsval=40 tsecr=40 RTT=200) ns3::TricklesShiehHe
ader (TCPBase=0 startCwnd=2 ssthresh=65535) Payload (size=1000)
+ 0.201344 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0
 DSCP Default ECN Not-ECT ttl 64 id 3 protocol 144 offset (bytes) 0 flags [none] length: 82 10.0.0.1 > 10.0.0.2) ns3::TcpHeader (49000 > 49153 Seq=0 Ack=0 Win
```

Research pipeline



Research pipeline with ns-3

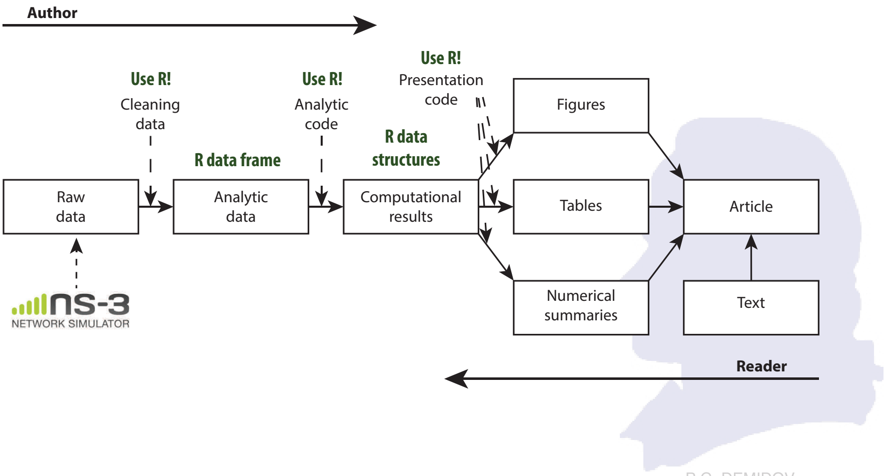


A tool for filling the pipeline

Key insight: use data science methods. Use R!

- Free software for statistical computing
- Tons of packages for different needs (CRAN)
- R-studio is a free IDE for developing
- `http://www.r-project.org`

Research pipeline



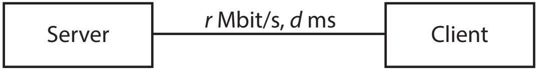
Literate programming

- Introduced by D. Knuth in 1984
- Essentially a programming language and a documentation system
- ns-3 uses C++ and Doxygen, thus it follows the literate programming ideas
- How to introduce the concept of literate programming to experiments?

R+Markdown+knitr+ns-3

- R is a programming language
- Markdown is a documenation language
- knitr is an R package that ties R and Markdown and produces a report
- R executes ns-3 using shell commands and parses stdout and stderr

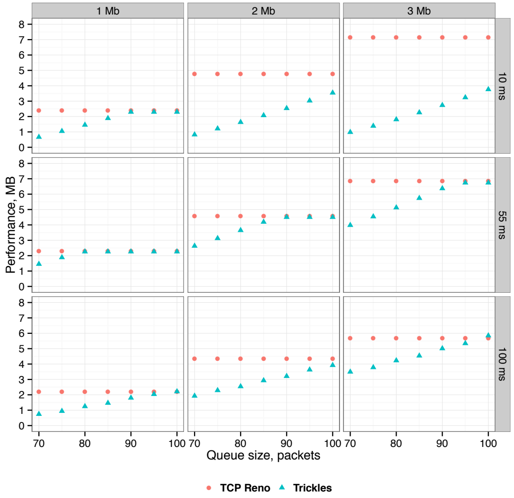
Simple experiment



- $r = \{1.0Mb/s, 2.0Mb/s, 3.0Mb/s\}$;
- $b = \{10ms, 55ms, 100ms\}$;
- queue length varies from 70 to 100 with the step of 5 packets.

R Markdown source code is available at
https://github.com/dchaly/stateless/ns-3.20/ns3-stateless-report.Rmd

Experimental evaluation



Future work

- More model evaluations, testing and debugging
- Develop new stateless protocols
- Performance analysis of stateless protocols in different settings: wired, wireless, SDN etc.