Pusan National University

네트워크 텀 프로젝트

ns-3를 이용한 TCP Congestion Control Protocol 성능 비교

2021. 11.

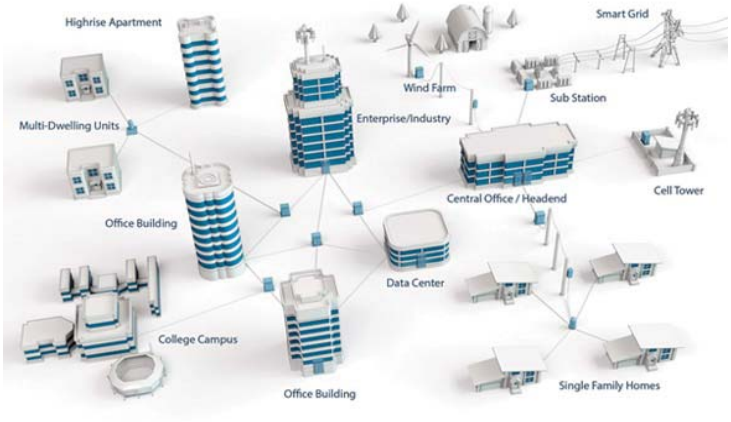
정보컴퓨터공학부 유영환

FNC
FUTURE NETWORK & COMMUNICATION

INDEX.

1 Starting ns-3

- ns-3는?
  - Discrete event network simulator
  - 실제 환경과 유사한 가상의 네트워크를 구축
  - 교육과 연구 목적으로 제공되는 오픈 소스 프로그램
  - 계층 구조를 가지며 패킷 기반의 통신 프로토콜



- 실행 환경
  - Virtual machine : VirtualBox, Vmware
  - ubuntu
  - macOS

- ns-3 설치
  - 홈페이지(https://www.nsnam.org/)
  - 최신 버전(ns-3.35) download 후에 압축풀기
  - 빌드

```
parksj@parksj-H110M-DS2V:~$ cd ns3/ns-allinone-3.29/
parksj@parksj-H110M-DS2V:~/ns3/ns-allinone-3.29$ ls
README  build.py      netanim-3.108  pybindgen-0.17.0.post58+ngcf00cc0
bake    constants.py  ns-3.29        util.py
parksj@parksj-H110M-DS2V:~/ns3/ns-allinone-3.29$ ./build.py
```

- ns-3 설치
  - 테스트

```
parksj@parksj-H110M-DS2V:~/ns3/ns-allinone-3.29/ns-3.29$ ./test.py -c core
Waf: Entering directory `/home/parksj/ns3/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/parksj/ns3/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.847s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      energy
fd-net-device            flow-monitor             internet
internet-apps            lr-wpan                  lte
mesh                     mobility                 mpi
netanim (no Python)      network                  nix-vector-routing
olsr                     point-to-point           point-to-point-layout
propagation              sixlowpan                spectrum
stats                    tap-bridge               test (no Python)
topology-read            traffic-control          uan
virtual-net-device       wave                     wifi
wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    openflow
visualizer

0 of 0 tests passed (0 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors

***  Note: ns-3 tests are currently disabled. Enable them by adding
***  "--enable-tests" to ./waf configure or modifying your .ns3rc file.

***  Note: ns-3 examples are currently disabled. Enable them by adding
***  "--enable-examples" to ./waf configure or modifying your .ns3rc file.
```


- ns-3 설치
  - 스크래치 파일 실행

```
parksj@parksj-H110M-DS2V:~/ns3/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/scratch-simulator
Waf: Entering directory `/home/parksj/ns3/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/parksj/ns3/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.226s)
Scratch Simulator
```

  - 설치 참조 사이트
    - https://www.nsnam.org/wiki/Installation

2 Key Abstraction

- Node
  - Computing device, host, end system
  - Represented in C++ by the class Node
- Application
  - ns-3 applications run on Nodes to drive simulations
  - Represented in C++ by the class Application
  - ex) UdpEchoClientApplication, UdpEchoServerApplication

- Channel
  - One connects a Node to an object representing a communication channel
  - Represented in C++ by the class Channel
  - ex) CsmaChannel, PointToPointChannel, WifiChannel
- Net Device
  - NIC(Network Interface Card)
  - Represented in C++ by the class NetDevice
  - a Node may be connected to more than one channel via multiple NetDevices

- Topology Helpers
  - Combine many distinct operations for ease of use
  - Create a NetDevice, add a MAC address
  - Install that net device on a Node, configure the node's protocol stack
  - Connect the NetDevice to a Channel

3 TCP Congestion Control

- TCP Congestion control in ns-3
  - src/internet/model/tcp-l4-protocol.cc

```
TypeId
TcpL4Protocol::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::TcpL4Protocol")
    .SetParent<IpL4Protocol> ()
    .SetGroupName ("Internet")
    .AddConstructor<TcpL4Protocol> ()
    .AddAttribute ("RttEstimatorType",
                   "Type of RttEstimator objects.",
                   TypeIdValue (RttMeanDeviation::GetTypeId ()),
                   MakeTypeIdAccessor (&TcpL4Protocol::m_rttTypeId),
                   MakeTypeIdChecker ())
    .AddAttribute ("SocketType",
                   "Socket type of TCP objects.",
                   TypeIdValue (TcpNewReno::GetTypeId ()),
                   MakeTypeIdAccessor (&TcpL4Protocol::m_congestionTypeId),
                   MakeTypeIdChecker ())
    .AddAttribute ("RecoveryType",
                   "Recovery type of TCP objects.",
                   TypeIdValue (TcpClassicRecovery::GetTypeId ()),
                   MakeTypeIdAccessor (&TcpL4Protocol::m_recoveryTypeId),
                   MakeTypeIdChecker ())
    .AddAttribute ("SocketList", "The list of sockets associated to this protocol.",
                   ObjectVectorValue (),
                   MakeObjectVectorAccessor (&TcpL4Protocol::m_sockets),
                   MakeObjectVectorChecker<TcpSocketBase> ())
  ;
  return tid;
}
```

- TCP NewReno
  - src/internet/model/tcp-congestion-ops.cc
  - Slow start

```
 * As stated, we want to avoid the case when a cumulative ACK increases cWnd more
 * than a segment size, but we keep count of how many segments we have ignored,
 * and return them.
 *
 * \param tcb internal congestion state
 * \param segmentsAcked count of segments acked
 * \return the number of segments not considered for increasing the cWnd
 */
uint32_t
TcpNewReno::SlowStart (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked)
{
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);

  if (segmentsAcked >= 1)
    {
      tcb->m_cWnd += tcb->m_segmentSize;
      NS_LOG_INFO ("In SlowStart, updated to cwnd " << tcb->m_cWnd << " ssthresh " << tcb->m_ssThresh);
      return segmentsAcked - 1;
    }

  return 0;
}
```


- TCP NewReno
  - src/internet/model/tcp-congestion-ops.cc
  - Congestion avoidance

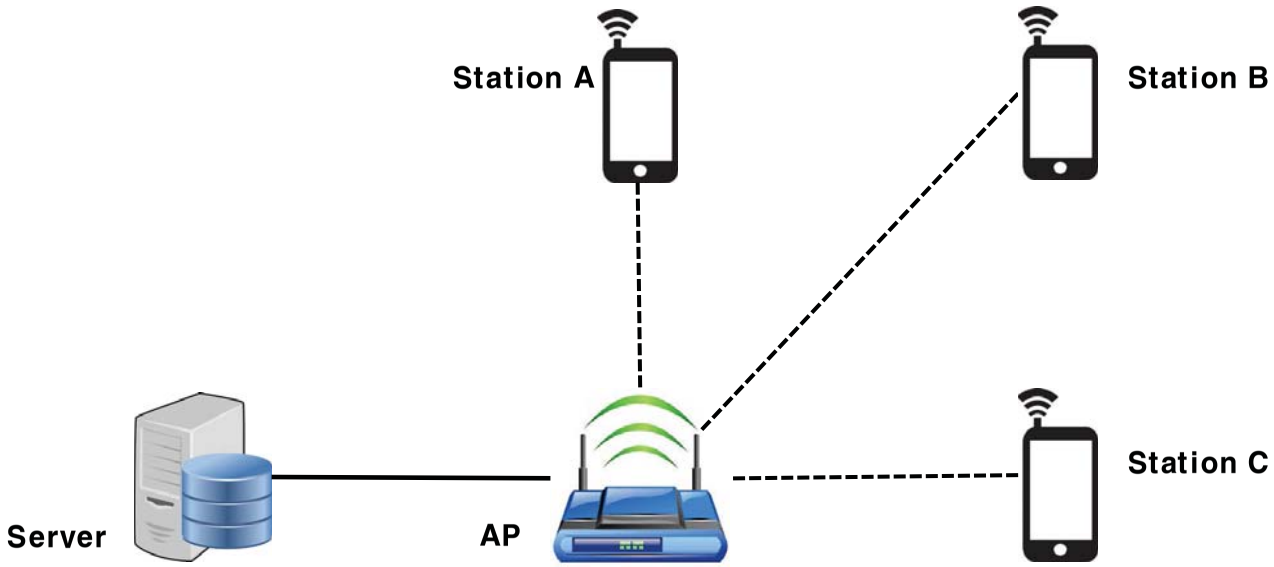
```
/**
 * \brief NewReno congestion avoidance
 *
 * During congestion avoidance, cwnd is incremented by roughly 1 full-sized
 * segment per round-trip time (RTT).
 *
 * \param tcb internal congestion state
 * \param segmentsAcked count of segments acked
 */
void
TcpNewReno::CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked)
{
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);

  if (segmentsAcked > 0)
    {
      double adder = static_cast<double> (tcb->m_segmentSize * tcb->m_segmentSize) / tcb->m_cWnd.Get ();
      adder = std::max (1.0, adder);
      tcb->m_cWnd += static_cast<uint32_t> (adder);
      NS_LOG_INFO ("In CongAvoid, updated to cwnd " << tcb->m_cWnd <<
                   " ssthresh " << tcb->m_ssThresh);
    }
}
```

4 Network Term

- 목적: TCP 혼잡 제어 알고리즘 성능 비교
- 방법
  - 세 개의 알고리즘 선택
    - TCP Vegas, TCP NewReno 포함
    - TcpHybla, TcpHighSpeed, TcpHtcp, TcpScalable, TcpVeno, TcpBic, TcpYeah, TcpIllinois, TcpWestwood, TcpLedbat, TcpLp 중 택 1
  - 사용할 통신 에러율: 0.01, 0.05, 0.1
  - 에러율에 따른 세 개의 알고리즘 성능 비교 및 이유 분석

- 시나리오 설명
  - 시나리오 파일(network-term.cc)을 scratch 폴더로 복사
  - 네트워크 구조
    - 3 Stations – 1 AP – 1 Server (TCP connection)
    - 각각의 Station에서 data 생성
    - Server가 sink node 역할



- network-term.cc
  - 시나리오 설정 값

```
uint32_t payloadSize = 400;                       /* Transport layer payload size in bytes. */
std::string dataRate = "10Mbps";                  /* Application layer datarate. */
std::string tcpVariant = "TcpNewReno";            /* TCP variant type. */
std::string phyRate = "HtMcs7";                   /* Wi-Fi Physical layer bitrate. */
bool pcapTracing = false;                         /* PCAP Tracing is enabled or not. */
double error_p = 0.01;                            /* Packet error rate. */
std::string bandwidth = "10Mbps";                 /* Data rate for P2P links. */
std::string delay = "1ms";                        /* Propagation delay. */
bool sack = true;                                 /* Selective ACK is enabled or not. */
double simulationTime = 100;                      /* Simulation time in seconds. */

std::string queue_disc_type = "ns3::PfifoFastQueueDisc";
std::string recovery = "ns3::TcpClassicRecovery";
```


- network-term.cc
  - Command line arguments
    - E.g., ./waf --run “scrach/network-term --tcpVariant=TcpHybla ”

```
/* Command line argument parser setup. */
CommandLine cmd;
cmd.AddValue ("payloadSize", "Payload size in bytes", payloadSize);
cmd.AddValue ("dataRate", "Application data ate", dataRate);
cmd.AddValue ("tcpVariant", "Transport protocol to use: TcpNewReno, "
              "TcpHybla, TcpHighSpeed, TcpHtcp, TcpVegas, TcpScalable, TcpVeno, "
              "TcpBic, TcpYeah, TcpIllinois, TcpWestwood, TcpWestwoodPlus, TcpLedbat, "
              "TcpLp", tcpVariant);
cmd.AddValue ("phyRate", "Physical layer bitrate", phyRate);
cmd.AddValue ("simulationTime", "Simulation time in seconds", simulationTime);
cmd.AddValue ("pcap", "Enable/disable PCAP Tracing", pcapTracing);
cmd.AddValue ("error_p", "Packet error rate", error_p);
cmd.AddValue ("bandwidth", "Bottleneck bandwidth", bandwidth);
cmd.AddValue ("delay", "Bottleneck delay", delay);
cmd.AddValue ("queue_disc_type", "Queue disc type for gateway (e.g. ns3::CoDelQueueDisc)", queue_disc_type);
cmd.AddValue ("sack", "Enable or disable SACK option", sack);
cmd.AddValue ("recovery", "Recovery algorithm type to use (e.g., ns3::TcpPrrRecovery", recovery);
cmd.Parse (argc, argv);
```

- network-term.cc
  - Burst error model
  - 참조 파일: src/network/utils/error-model.cc

```
/* Configure the error model */
// Here we use BurstErrorModel with packet error rate.
Ptr<UniformRandomVariable> uv = CreateObject<UniformRandomVariable> ();
uv->SetStream (50);
BurstErrorModel error_model;
error_model.SetRandomVariable (uv);
error_model.SetRandomBurstSize (uv);
error_model.SetBurstRate (error_p);

PointToPointHelper UnReLink;
UnReLink.SetDeviceAttribute ("DataRate", StringValue (bandwidth));
UnReLink.SetChannelAttribute ("Delay", StringValue (delay));
UnReLink.SetDeviceAttribute ("ReceiveErrorModel", PointerValue (&error_model));
```

  - 참고 문서(https://www.nsnam.org/doxygen/)

- network-term.cc
  - 성능 평가

```
// Set up tracing if enabled
if (pcapTracing)
  {
    wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
    wifiPhy.EnablePcap ("AccessPoint", apDevice);
    wifiPhy.EnablePcap ("Station", staDevices);
    UnReLink.EnablePcapAll("server", true);
  }

// Flow monitor
FlowMonitorHelper flowHelper;
flowHelper.InstallAll();

/* Start Simulation */
Simulator::Stop (Seconds (simulationTime + 1));
Simulator::Run ();
flowHelper.SerializeToXmlFile ("network-term.flowmonitor", true, true);
Simulator::Destroy ();
```

- 성능 평가
  - network-term.flowmonitor 내용 일부

```
<Flow flowId="1" timeFirstTxPacket="+1e+09ns"
timeFirstRxPacket="+1.01234e+09ns"
timeLastTxPacket="+1.01e+11ns" timeLastRxPacket="+1.01e+11ns"
delaySum="+4.03266e+12ns" jitterSum="+4.35031e+10ns"
lastDelay="+3.52218e+07ns" txBytes="60831172"
rxBytes="60497596" txPackets="134584" rxPackets="133846"
lostPackets="616" timesForwarded="133846">
```

  - 시뮬레이션 기간 동안 flow 별 throughput 계산
    - 데이터 손실율 = (txBytes – rxBytes) / txBytes * 100
    - 패킷 손실율 = lostPackets / txPackets * 100
    - 전체 전송지연시간 = delaySum

- 마감: 12월 24일 0시
- 팀 없음 → 개인 과제
- 제출 요령: 보고서 업로드
- 보고서 구성
  1. 세 가지 TCP congestion control 프로토콜에 대한 설명
  2. 서로 다른 에러율별 세 가지 프로토콜의 성능 실험 및 결과 분석
     - 시나리오 파일에서 설정 값들을 변경하여 실험 환경 수정 가능
     - 성능 분석 결과(전송량, 손실율, 지연 시간 등) 제시와 결과에 대한 이유 설명