DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

1222 · 2022
800 ANNI

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SIGNET
signet.dei.unipd.it

An ns-3 Implementation of a Bursty Traffic Framework for Virtual Reality Sources

Mattia Lecci, Andrea Zanella, Michele Zorzi

Workshop on ns-3 (WNS3)
Virtual Event, US, June 2021

Introduction

- eXtended Reality (XR)
  - Growing interest in consumer market
- Demanding requirements
  - Resolution 5073x5707 per eye
  - 120 FPS refresh rate
  - 5-9 ms network latency
- No available traffic model

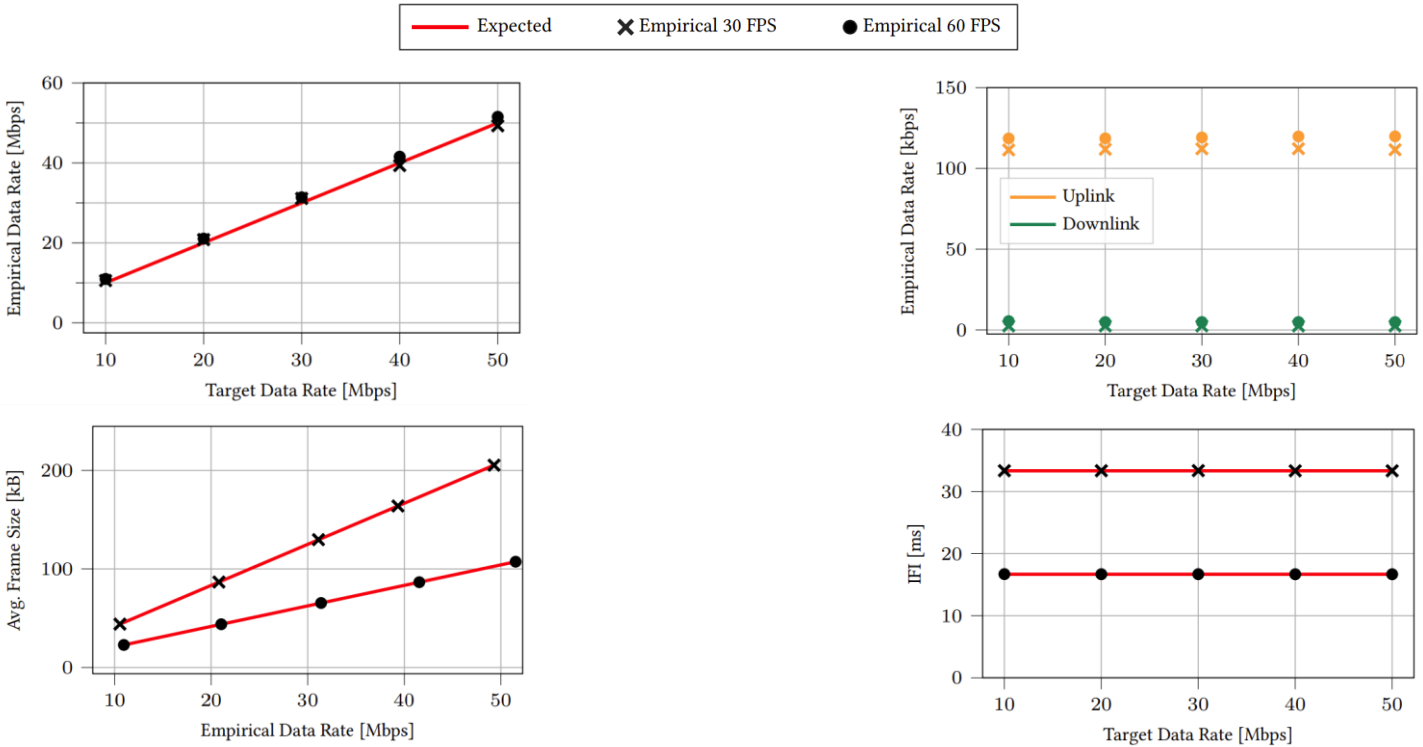
Contributions

1. First generative model for XR traffic
   - Based on >90 min of acquired VR traces
   - Parameterized frame rate and data rate
2. Ns-3 implementation for bursty traffic (available)
   - Flexible interface for burst generators
3. Ns-3 implementation of XR traffic model (available)
4. Several acquisitions of VR traffic (available)
   - Can be imported directly into ns-3 simulations

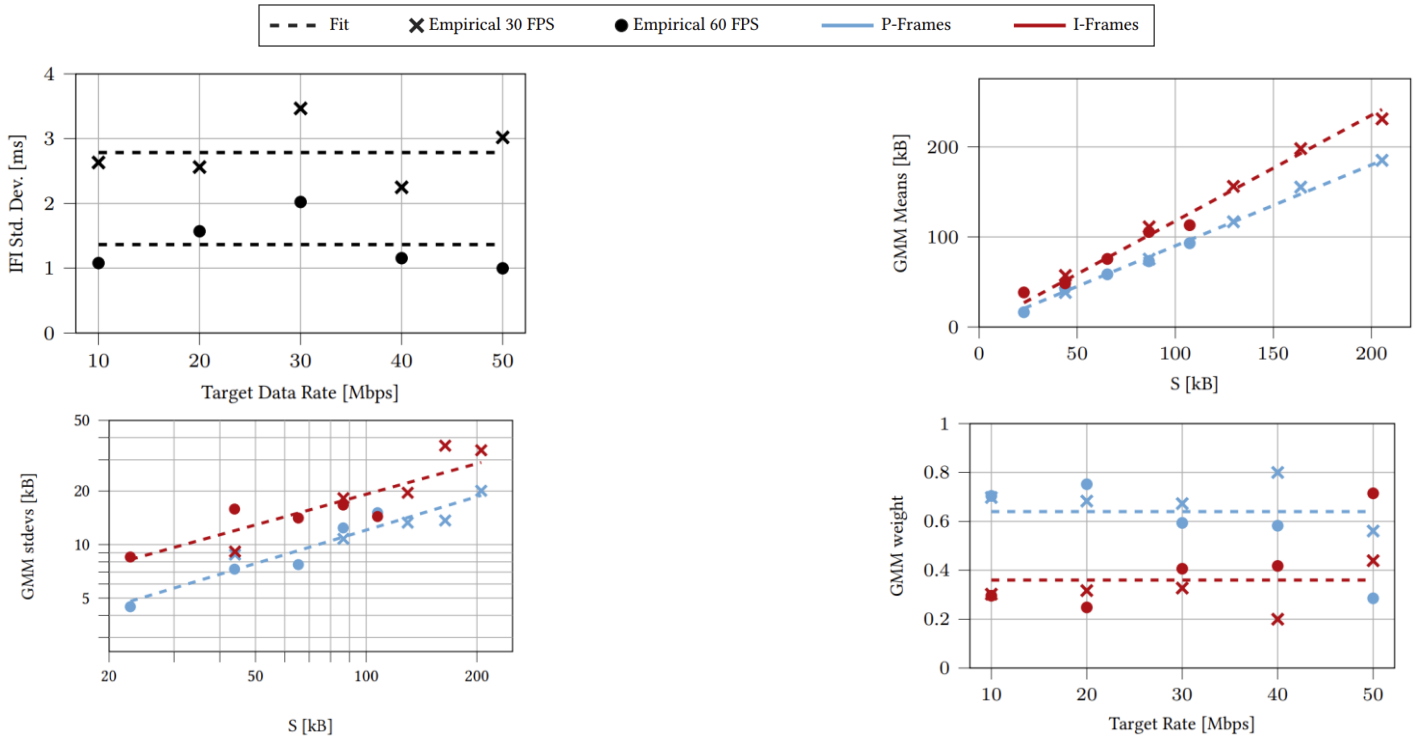
VR Traffic: Acquisition

- Rendering server
  - Desktop PC with GPU
- VR headset
  - Phone connected to rendering server via USB tethering
- About 10 min long traces
  - {30, 60} FPS, {10, 20, 30, 40, 50} Mbps
- Static SteamVR waiting room
  - Almost no movements from phone

VR Traffic: Analysis



Traffic Model



Ns-3 Implementation

- Novel customizable traffic model
- Based on ns-3.33[1]
  - Compatible with prior versions, but `CsvReader` is needed
- Large packets are split into multiple fragments from `BurstyApplication`
  - `BurstSink` tries to re-aggregate them, if possible
  - `BurstGenerator` provides an interface to customize the traffic statistics
  - `SeqTsSizeFragHeader` includes information necessary for `BurstSink`

[1] Available as ns-3 app: https://apps.nsnam.org/app/bursty-app/

Bursty Application

- Periodically sends bursts of data split into fragments of a given size (attribute `FragmentSize`)
  - Burst size and periodicity are controlled by a `BurstGenerator`
  - `BurstyHelper` simplifies the setup
- Each fragment carries a `SeqTsSizeFragHeader`
- Traces available for burst and fragment transmission

SeqTsSizeFragHeader

- Contains information regarding
  1. Burst sequence number
  2. Burst size
  3. Fragment sequence number
  4. Total number of fragments
  5. Timestamp (for stats)

Burst Generator Interface

- Defines two pure virtual functions:
  - `HasNextBurst`: bool
  - `GenerateBurst`: pair(burst size, time before next burst)
- Users can extend this class to account for
  - Different distributions
  - Correlations
  - Specific attributes

Simple Burst Generator

- Inspired from `OnOffApplication`
- Attributes:
  - `PeriodRv`
  - `BurstSizeRv`
- No correlation

VR Burst Generator

- Direct implementation of the proposed VR traffic model
- Attributes:
  - `FrameRate`
  - `TargetDataRate`
- No correlation
- Also implemented `LogisticRandomVariable` and `MixtureRandomVariable`

Trace File Burst Generator

- Takes advantage of `CsvReader` (ns-3.33)
  - Imports CSV traffic traces
- Useful to import static traces (for testing) or acquired traces
- Attributes:
  - `TraceFile`
  - `StartTime`
- About 90 min of real VR traces are also included

Burst Sink

- Inspired from `PacketSink`
- Expects to receive fragments from `BurstyApplication`(s)
  - Assumes UDP is used
  - Attempts to re-aggregate them into a single packet
  - Handles out-of-order fragments within burst
  - Does not apply any Forward Error Correction (FEC)
- Traces available for burst and fragment reception

Example

From `bursty-application-example.cc`:

```
// Create bursty application helper
BurstyHelper burstyHelper ("ns3::UdpSocketFactory", InetSocketAddress (serverAddress, portNumber));
burstyHelper.SetAttribute ("FragmentSize", UintegerValue (1200));
burstyHelper.SetBurstGenerator ("ns3::SimpleBurstGenerator",
                                "PeriodRv", StringValue ("ns3::ConstantRandomVariable[Constant=100e-3]"),
                                "BurstSizeRv", StringValue ("ns3::ConstantRandomVariable[Constant=10e3]"));

// Install bursty application
ApplicationContainer serverApps = burstyHelper.Install (nodes.Get (1));
Ptr<BurstyApplication> burstyApp = serverApps.Get (0)->GetObject<BurstyApplication> ();

// Example of connecting to the trace sources
burstyApp->TraceConnectWithoutContext ("FragmentTx", MakeCallback (&FragmentTx));
burstyApp->TraceConnectWithoutContext ("BurstTx", MakeCallback (&BurstTx));

// Create burst sink helper
BurstSinkHelper burstSinkHelper ("ns3::UdpSocketFactory",
                                 InetSocketAddress (sinkAddress, portNumber));

// Install burst sink
ApplicationContainer clientApps = burstSinkHelper.Install (nodes.Get (0));
Ptr<BurstSink> burstSink = clientApps.Get (0)->GetObject<BurstSink> ();

// Example of connecting to the trace sources
burstSink->TraceConnectWithoutContext ("BurstRx", MakeCallback (&BurstRx));
burstSink->TraceConnectWithoutContext ("FragmentRx", MakeCallback (&FragmentRx));
```

Example

From `bursty-application-example.cc`:

```
void
BurstRx (Ptr<const Packet> burst, const Address &from, const Address &to,
         const SeqTsSizeFragHeader &header)
{
  NS_LOG_INFO ("Received burst seq="
               << header.GetSeq () << " of header.GetSize ()=" << header.GetSize ()
               << " (burst->GetSize ()=" << burst->GetSize () << ") bytes from "
               << AddressToString (from) << " to " << AddressToString (to) << " at "
               << header.GetTs ().As (Time::S));
}

void
FragmentRx (Ptr<const Packet> fragment, const Address &from, const Address &to,
            const SeqTsSizeFragHeader &header)
{
  NS_LOG_INFO ("Received fragment "
               << header.GetFragSeq () << "/" << header.GetFrags () << " of burst seq="
               << header.GetSeq () << " of header.GetSize ()=" << header.GetSize ()
               << " (fragment->GetSize ()=" << fragment->GetSize () << ") bytes from "
               << AddressToString (from) << " to " << AddressToString (to) << " at "
               << header.GetTs ().As (Time::S));
}
```

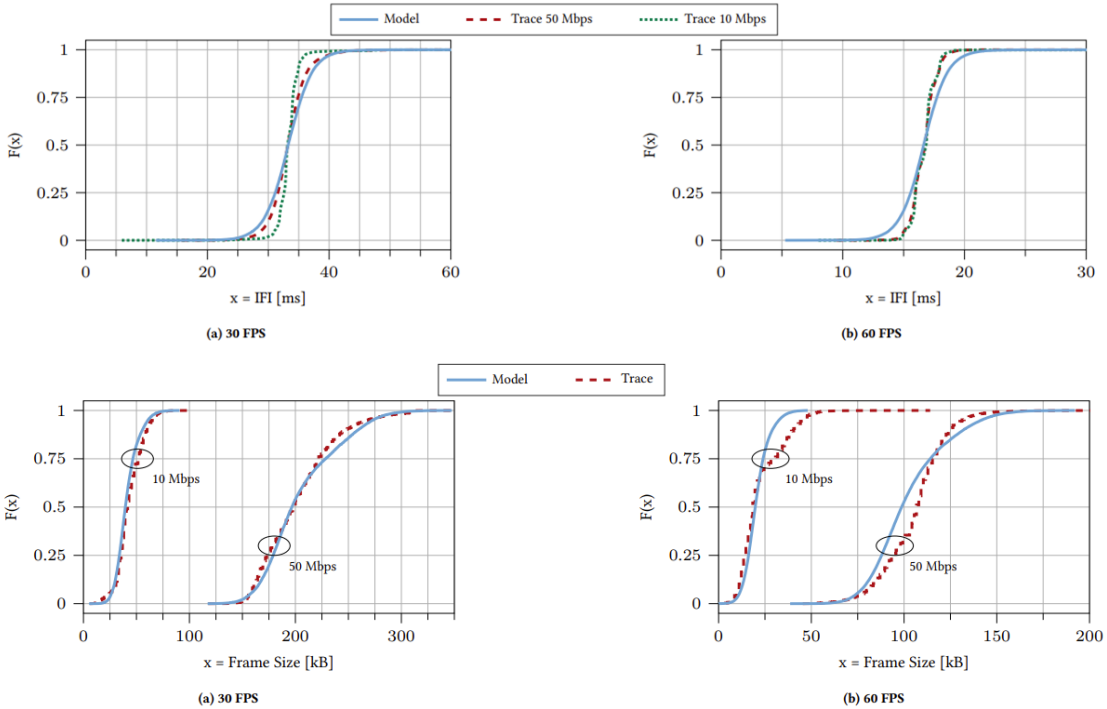
Example

From `bursty-application-example.cc`:

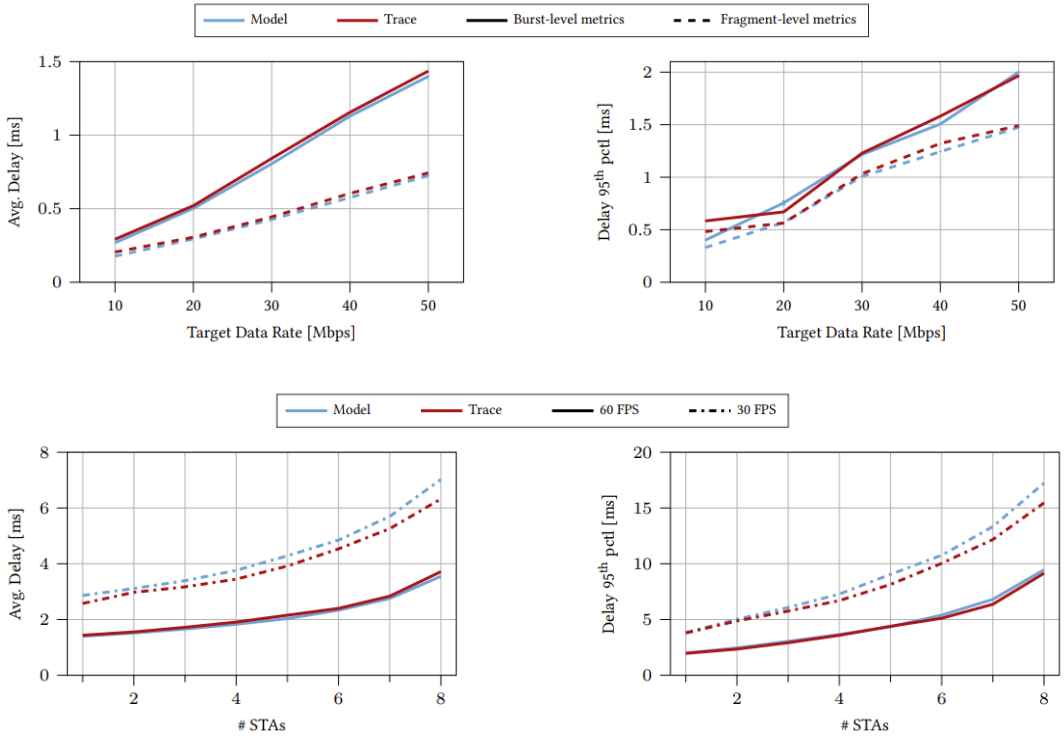
```
                        /ns-3-dev$ ./waf --run bursty-application-example
Waf: Entering directory `                    /ns-3-dev/build'
Waf: Leaving directory `                    /ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.687s)
+0.000000000s Sent burst seq=0 of header.GetSize ()=9784 (burst->GetSize ()=9784) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 0/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 1/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 2/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 3/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 4/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 5/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 6/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 7/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.000000000s Sent fragment 8/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=400) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0s
+0.003968000s Received fragment 0/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.005936000s Received fragment 1/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.007904000s Received fragment 2/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.009872000s Received fragment 3/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.011840000s Received fragment 4/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.013808000s Received fragment 5/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.015776000s Received fragment 6/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.017744000s Received fragment 7/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.018432000s Received fragment 8/9 of burst seq=0 of header.GetSize ()=9784 (fragment->GetSize ()=400) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.018432000s Received burst seq=0 of header.GetSize ()=9784 (burst->GetSize ()=9784) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0s
+0.100000000s Sent burst seq=1 of header.GetSize ()=9784 (burst->GetSize ()=9784) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 0/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 1/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 2/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 3/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 4/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 5/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 6/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 7/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.100000000s Sent fragment 8/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=400) bytes from 0.0.0.0:49153 to 10.1.1.1:50000 at +0.1s
+0.103968000s Received fragment 0/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0.1s
+0.105936000s Received fragment 1/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0.1s
+0.107904000s Received fragment 2/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0.1s
+0.109872000s Received fragment 3/9 of burst seq=1 of header.GetSize ()=9784 (fragment->GetSize ()=1200) bytes from 10.1.1.2:49153 to 0.0.0.0:50000 at +0.1s
```

Repeat...

Model Validation



Example Use Case



Conclusions

- Proposed simple VR traffic model
- Bursty application framework for ns-3
  - Flexible and customizable
  - Can support transmission of generic large packets
  - Implements the proposed VR traffic model
  - Openly available as ns-3 app[1]
- Feedbacks and contributions are welcomed!

[1] Available as ns-3 app: https://apps.nsnam.org/app/bursty-app/

Future Works

- Diversify acquisitions on games and VR applications
  - Realistic head movements will also be included
- Model second-order statistics
  - Temporal correlation of IFI and frame size
  - Cross-correlation between IFI and frame size
- Correlate VR traffic with head rotations
- Acquire different headsets
  - Proprietary encoding and streaming

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

1222·2022 800 ANNI

UNIVERSITÀ DEGLI STUDI DI PADOVA

SIGNET
signet.dei.unipd.it

An ns-3 Implementation of a Bursty Traffic Framework for Virtual Reality Sources

Thank you for your attention!

Ns-3 app: https://apps.nsnam.org/app/bursty-app/