

Project 1 Technical Specification

CS621: Network Programming – Spring 2019

Updated: Feb 13th, 2019

Project Outcomes

1. Enable network compression for point-to-point links in ns-3.
2. Implement the network application that detects the presence of network compression by end-hosts.
3. Verify and validate your simulated compression link and compression detection application.

Overview

For this project you will implement a network level compression link. You will then implement a network application to detect whether network compression is present to validate your simulated compression link. It is inspired by the work, [End-to-End Detection of Compression of Traffic Flows by Intermediaries](#), which is recommended that you read in detail up to Section VI.

Crucial to success in this project will be a deep and detailed reading and understanding of the ns-3 documentation, where it is relevant to your project. Among the [ns-3 documentation](#) are tutorials, a reference manual, a model library, and a full API reference. You should explore and make use of all of these resources.

Components

(1) Compression Link

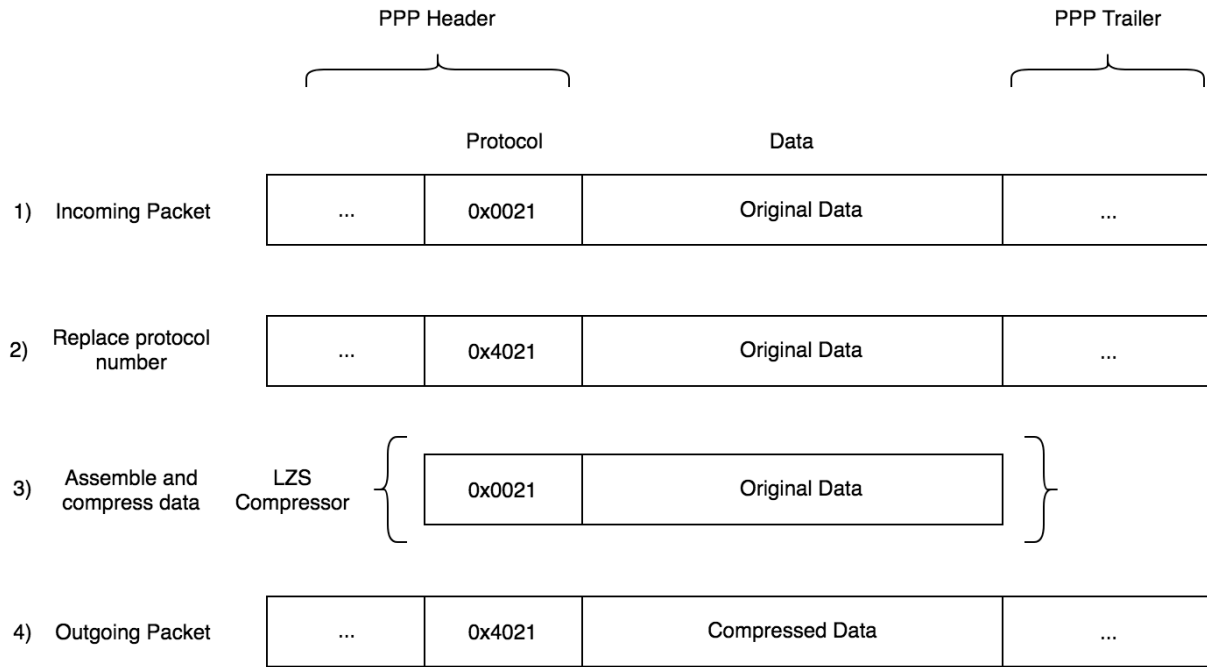
Your compression link application must be built using ns-3. It will be responsible for compression and decompression of incoming and outgoing packets. You will use the [Lempel-Ziv-Stac algorithm](#) for compression and decompression, for which you may find and use a library. Your compression link should follow some requirements in the [RFC 1974: PPP Stac LZS Compression Protocol](#).

Specifically, you should implement functionality that takes a PPP packet, and first checks the protocol number in the header. You should also have a configuration file that specifies packet types to compress. If you determine that the checked packet type matches one in the configuration file, you should pre-process then compress it. For the purposes of this project, your configuration file will only have one protocol entered, IP, protocol number 0x0021.

Pre-processing should first take the matched packet then replace the original protocol number with the LZS protocol number, 0x4021. It should then take the original protocol number and append it to the original data, and compress that whole bitstring and replace it with the original data section in the original packet, as illustrated in Figure 1. Decompressor, at the other side of the compression link, should then reverse all the pre-processing steps performed at the compressor, to retrieve the original incoming packet ((1) in Figure 1), before pushing it to the next interface.

You should not worry about implementing Compression Control Protocol and its corresponding control packets, as you may assume the two routers have already reached an Opened state and LZS has been negotiated as the primary compression algorithm. Similarly, you should also not implement anything from the negotiation phase, and not worry about using the exact compression data format, specified in X3.241-1994.

You may only need to modify [PointToPointNetDevice](#) to enable compression/decompression on point-to-point links. A good starting point to get familiar with, in addition to [PointToPointNetDevice](#) is the [ns-3 point-to-point model overview](#).



(Note: 0x0021 is the PPP IP protocol number, and 0x4021 is the PPP LZS protocol number)

Figure 1: The specified method to implement for assembling compressed packets, as specified in [RFC 1974](#).

A good starting point for building ns-3 applications might be this ns-3 wiki article, [How To Make and Use A New Application](#).

(2) Compression Detection Application

You will implement the network compression detection *only in the cooperative environment* as described [here](#) (Section IV). In summary, your network application is a client/server application where the sender sends two sets of 6000 UDP packets back-to-back (called packet train), and the receiver records the arrival time between the first and last packet in the train. The first packet train consists of all packets of size 1100 bytes in payload, filled with all 0's, while the second packet train contains random sequence of bits. You can generate random sequence of bits using `/dev/random`. If the difference in arrival time between the first and last packets of the two trains ($\Delta t_H - \Delta t_L$) is more than a fixed threshold $\tau = 100\text{ ms}$, the application reports **Compression detected!**, whereas when the time difference is less than τ there was probably no compression link on the path and the application should display **No compression was detected**.

Your application is required to take in at least one command-line argument, Compression Link Capacity, which specifies the maximum bandwidth across the link between the two routers.

(3) Simulation Verification and Validation

Create a 4-node topology in ns-3 as illustrated in Figure 2. Nodes *S* and *R* are the end-hosts running the network application. Nodes *R1* and *R2* are the intermediate routers where the link between them is compression-enabled. Your simulations should also be built using ns-3. You should include four logically separate simulations, each doing one of the following:

- Transmit low entropy data over a network topology without a compression link.

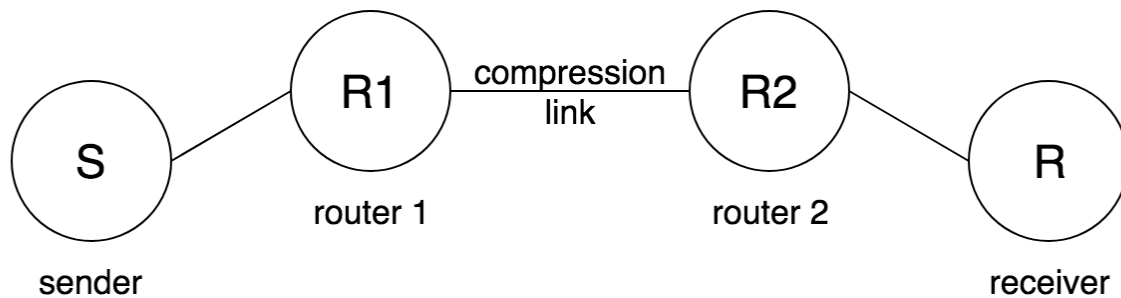


Figure 2: A 4-node topology with a compression link.

- Transmit high entropy data over a network topology without a compression link.
- Transmit low entropy data over a network topology with a compression link.
- Transmit high entropy data over a network topology with a compression link.

You will then vary the middle link ($R1R2$) capacity parameter from 1,2,3,...,10 Mbps, reporting each $\Delta t_H - \Delta t_L$. The link capacity of the two outer (non compression) links on your four node topology should be set persistently to 8 Mbps.

Make sure you are careful to control for confounding variables across your four simulations, as you will ultimately be comparing time between these four simulation types so as to try and detect the compression link.

Submission Requirements and Deadlines

Your final submissions will be graded on design, implementation, code quality, coding style, and documentation quality. You will submit two documents: (1) Technical usage manual, and (2) final report. You should follow the [ns-3 coding style](#). While you are not required to submit a design document, you are strongly encouraged to maintain one in order to deliver your project successfully.

There will be an in-class project checkpoint on **February 14th**, where each group will discuss their project progress with the professor.

All code and the required usage manual, should be pushed to your repositories before **Thursday March 7, 11:59 PM**. Commits made after the due date will not be reviewed.

Keep in mind that at the end of the semester, you will present and demo your project **May 9th** in class, and submit a final report that is due **May 15th, 11:59PM**.