SICE-Networks / **Net-Fall22** Public

Code Pull requests Actions **Wiki** Security Insights

07_gbn

Jump to bottom

Alex Shroyer edited this page on Nov 10, 2022 · 3 revisions

Reliable UDP (Part 2)

Instructions

The purpose of this assignment is to complement your existing Reliable UDP (RUDP) stop-and-wait protocol with a go-back-N protocol. As we have learned, stop-and-wait is horribly slow to transmit lots of data, especially as latency and loss in the network increases. The previous assignment shows this slow file transfer speed on the lossy network.

In this assignment, we will improve the protocol and get faster file transfer speeds on the same lossy networks!

Task 1 - Implement go-back-N protocol

For go-back-N, you will need at a minimum:

- An allocated buffer that stores your client's window of data. You may use a statically-sized buffer (e.g. 128 "slots").
- The ability to buffer and send data while keeping track of which "slots" the receiver acknowledges.
- A timer to indicate when to **go-back-N** and retransmit.

In the stop-and-wait protocol, the client has a "buffer" of size 1, and can only send the next message after the current message is acknowledged by the server. Go-back-N allows you to use a larger buffer, and send up to N messages before waiting for acknowledgment or timeout.

Setup

```
cd netster # NOTE: or netster_py if you're using Python!
make part4
```

In order to build your code, run `make` in the `src` directory. You may also clean your directory automatically with `make clean`.

Instructions

Extend your `gbn_client` and `gbn_server` functions to use go-back-N to transfer the files. File transfer should complete without error even if you connect to the “lossy” ports on `solar` and `lunar`.

First start a server:

```
ashroyer@solar:~/netster$ ./netster -i 0.0.0.0 -p $PORT -f destinationfile.txt -r 2
```

Then start a client:

```
ashroyer@lunar:~/netster$ ./netster solar.open.sice.indiana.edu -p $PORT -f sourcefile.txt -r
```

The client then transfers the file to the server. When the transfer is complete, both the client and server should exit.

Header Format

While stop-and-wait only required a single bit header to properly distinguish messages, go-back-N is more complex. We do not require your code to use a specific header format, but your client and server **must** be consistent with each other.

> (BEGIN NOTE)
>
> To be (optionally) compatible with `netster-ref`, your code must use the same header format.
>
> `netster-ref` header is 1 byte (8 bits) representing the sequence number. Because it is 8 bits, the range of possible sequence numbers is 0-255 inclusive. Sequence numbering starts at 0.

> It is not required for your code to be compatible with `netster-ref` , so you are free to ignore this (note).
>
> (END NOTE)

Task 2 (Graduate Sections) - Add simple congestion control

You must implement a congestion control "window" that begins small and grows as data is successfully acknowledged. Your strategy for acknowledgments (including negative acknowledgment) is up to you.

The goal is to prevent the go-back-N protocol from blasting the network with a full window of data at once, but start slow and then "probe" the network to determine how large the window may grow to avoid loss.

Clone this wiki locally

```
https://github.iu.edu/SICE-Networks/Net-Fall22.wiki.git
```