ECE428 MP0

Introduction

The objective of this MP0 is to create a simple network application.
It contains two tasks described as follows.

Task1 is to implement an event logger that receives messages
from multiple network nodes and stores them in a centralized log. The logger should
be able to receive messages from the network nodes using the Python script generator.py,
which generates messages for random events. The rate of generated events/messages
can be controlled using two input arguments. The logger should also allow other
network nodes to send their messages to it using TCP protocol on a specified port.
The logger should then write the messages and node information into a log file,
including the node name, connection time, messages, and disconnection time.

Task2 is to do an evaluation for this network. The log file produced by the network
system can be evaluated to track two metrics: (1) the delay between event generation
and recording in the log file, and (2) the network bandwidth used by the logger.
The logger can use the current time when writing messages to the log file and count
the number of bytes received from all network nodes over time. The metric values can
be stored directly by the logger into a separate file or sent to standard output and
redirected into a file by the shell.

Design Choices

To complete this machine problem, I developed two functions: *logger.py* and *node.py*.
The *logger.py* function acts as a web server, collecting messages sent by the *node.py*
function, which serves as a web client. The messages are generated by *generator.py*,
including a timestamp and a random message. The communication between *logger.py* and
node.py
functions is established through a TCP connection, allowing the transmission of messages
between the two functions.

Prerequisites

Python 3.0+
socket, sys, time, threading, matplotlib. pyplot and numpy module in Python3

Usage

To do task1:

Start one terminal and run:

`python3 logger.py`

Start any number of terminals (up to 24), and run:

`python3 -u generator.py 1 100 | python node.py (#Nodename) 127.0.0.1 1234`

Start To do task2:

Three Nodes run parallel:

Logger: `python logger.py 1234 2`

Nodes Command:

```
python3 -u generator.py RA(from 0.1-1) 100 | python3 node.py A 127.0.0.1 1234 &
python3 -u generator.py 1 100 | python3 node.py B 127.0.0.1 1234 &
python3 -u generator.py 2 100 | python3 node.py C 127.0.0.1 1234
```

Data Analysis

In Task2, we define the delay and bandwidth as follows:

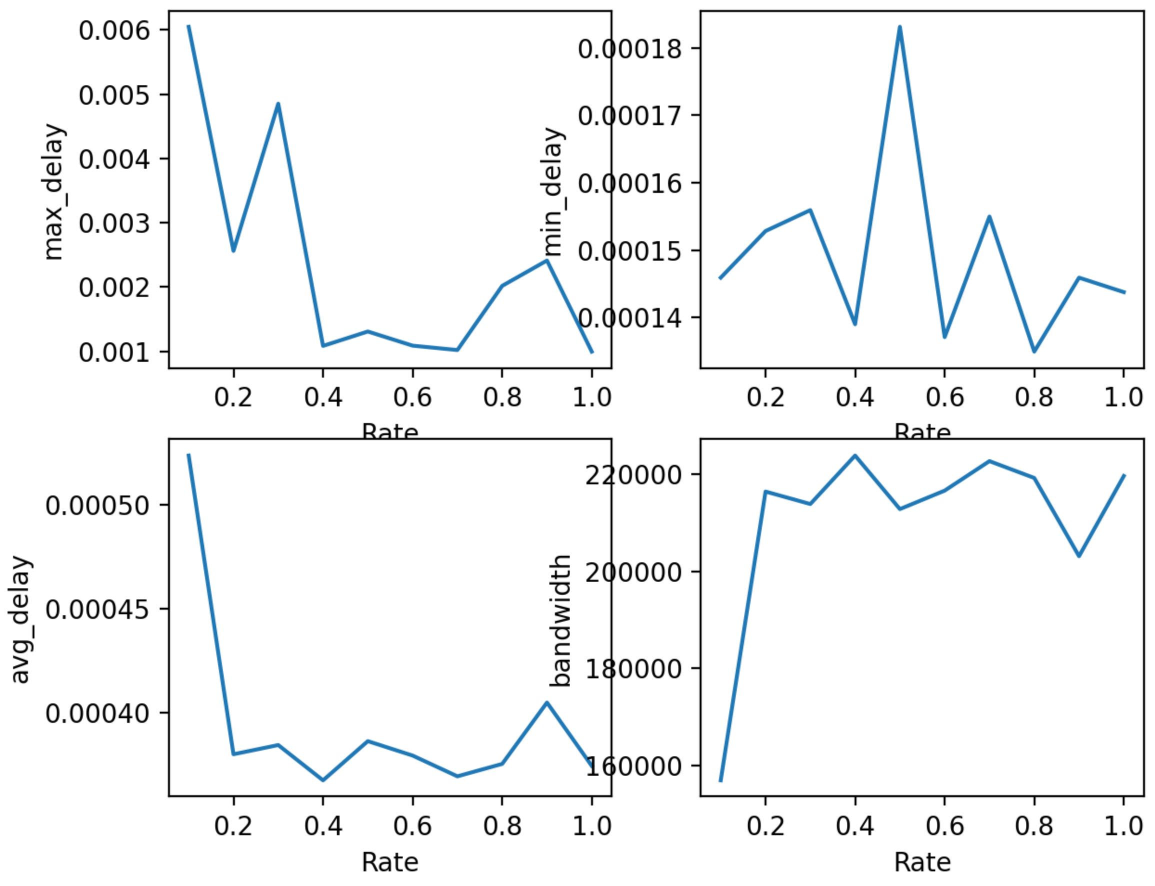
Delay:

$$Delay = time(written) - time(generated)[sec]$$

Bandwidth:

$$Bandwidth = \frac{n_bytes(received)}{Delay}[bytes/sec]$$

Plots and Conclusion



As the sending rate of Node A increases, the delay decreases on the whole, while bandwidth is on the rise.

Bugs Encountered

Bug 1: sticky packet problem

- **Symptom:** Sometimes, the logger can not split *node_name* with the first *message*, so the *node_name* is not correct, and the first *message* is missing.
- **Cause:** The node send *node_name* and first *message* together, a typical **TCP sticky packet problem** (although TCP do not send packet).

- **Solution:** Add a time.sleep() function after sending *node_name*

Authors

Zhanhao He 3190110713 Zhanhao.19@intl.zju.edu.cn
Bo Pang 3190110669 Bo.19@intl.zju.edu.cn