Server, Gateway, Client Communication

*Python Implementation of a flow control protocol involving a gateway, server and client*

This documentation includes an in-depth description of the Flow Control Protocol, the code implementation, strength and weaknesses of my implementation and a n analysis of my program including screenshots of the traffic being exchanged by my programs

Sharon Olorunniwo - 16323766

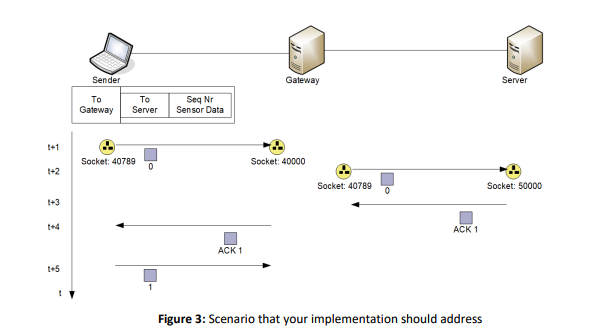
11/10/2017

Server, Gateway, Client Communication

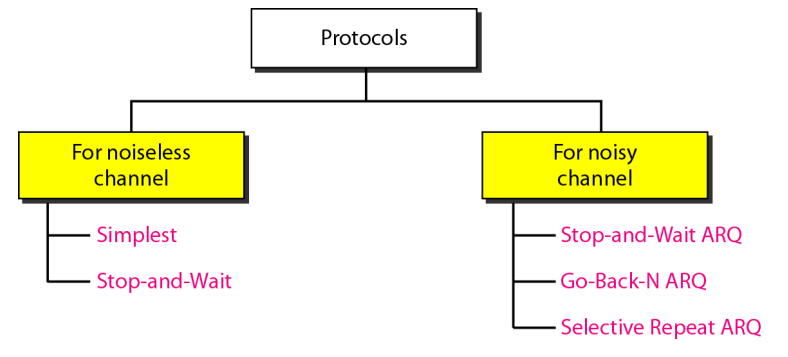
Python Implementation of a flow control protocol involving a gateway, server and client

#### Table of contents

1. What is Flow Control?
2. Stop & Wait ARQ
3. Code Implementation and Analysis
4. Sliding Window Protocol
5. References and Reflection



##### What is Flow Control?



In telecommunications, flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node. Flow control mechanisms can be classified by whether the receiving node sends feedback to the sending node.

Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process it. This can happen if the receiving computers have a heavy traffic load in comparison to the sending computer, or if the receiving computer has less processing power than the sending computer.

Flow control is introduced in Data Link Layer. It also works on several higher layers. The main concept of Flow Control is to introduce efficiency in Computer Networks. There are two approaches to Flow Control: **Feedback based Flow Control and Rate based Flow Control.**

For this assignment the most simplistic approach is to implement a Feedback based Flow Control protocol. The way this works is, until the sender receives feedback from the receiver, it will not be able to send next packet of data.

**Types of Feedback based Flow Control**

**A. Stop-and-Wait Protocol**

**B. Sliding Window Protocol**

1. A One-Bit Sliding Window Protocol
2. A Protocol Using Go Back N
3. A Protocol Using Selective Repeat

Stop-and-Wait Protocol

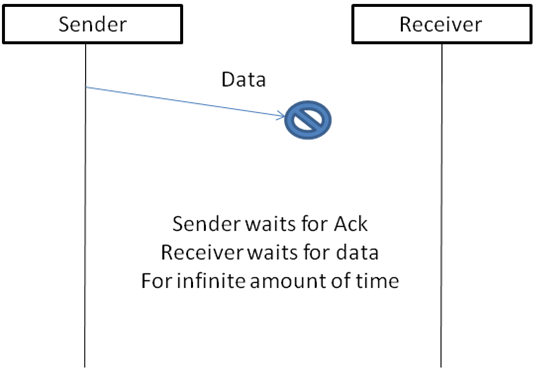
Stop-and-wait flow control is the simplest form of flow control. In this method, the receiver indicates its readiness to receive data for each frame, the message is broken into multiple frames. The sender waits for an ACK (acknowledgement) after every frame for specified time (called time out). It is sent to ensure that the receiver has received the frame correctly. It will then send the next frame only after the ACK has been received, hence why it is a feedback based protocol.

## **Characteristics**

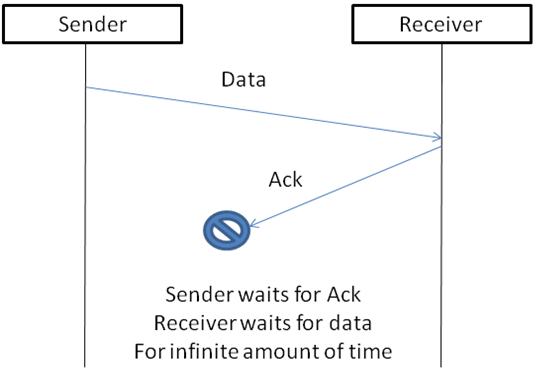
* Used in Connection-oriented communication.
* It offers error and flow control
* It is used in Data Link and Transport Layers
* Stop and Wait ARQ mainly implements Sliding Window Protocol concept with Window Size 1

However, nothing is ever perfect in the world and the Stop-and-Wait Protocol is no exception. Here are three different scenarios that could arise during the data packet transmission.

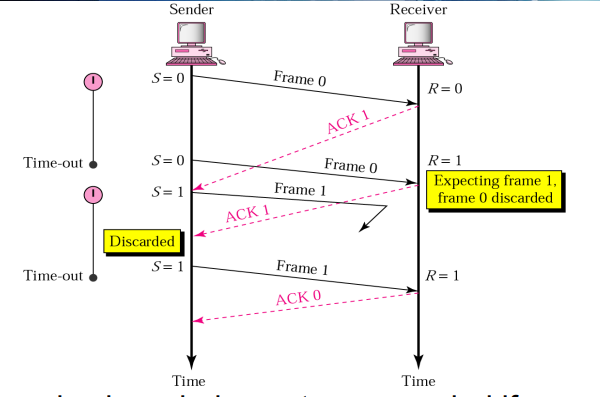
### Scenario 1: Lost Data



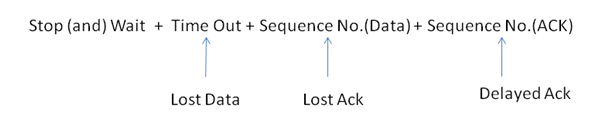
### Scenario 2: Lost Acknowledgement



### Scenario 3: Delayed Acknowledgement/Data:



In order to combat those possible issues, we implement a Stop-and-Wait ARQ ***(Automatic Repeat Request),*** which handles both error and flow control.

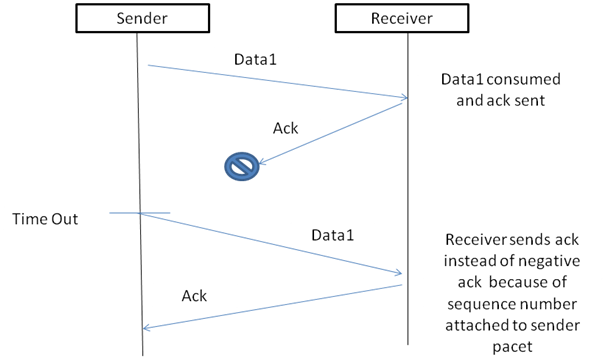


Below are the same three scenarios but in a Stop-and-Wait ARQ rather than just a Stop-and-Wait Protocol.

### Scenario 1: Lost Data

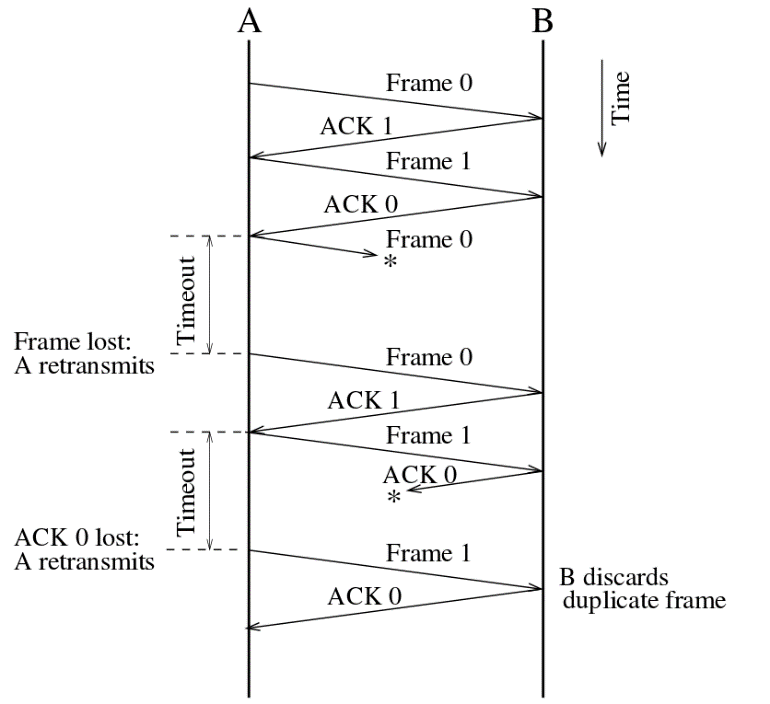
### cn6

### Scenario 2: Sequence Number (Data)



### A working Example

1. Sender A sends a data frame or packet with sequence number 0.
2. Receiver B, after receiving data frame, sends and acknowledgement with sequence number 1 (sequence number of next expected data frame or packet). There is only one-bit sequence number that implies that both sender and receiver have buffer for one frame or packet only.



In this scheme we take Communication Channel error free, but if the Channel has some errors than receiver is not able to get the correct data from sender, so it will not possible for sender to send the next data (because it will not get acknowledge from receiver). So, it will end the communication, to solve this problem there are two new concepts were introduced.

**TIMER -** if the sender was not able to get acknowledgment in the time than, it sends the buffered data once again to receiver. When the sender starts to send the data, it starts timer.

**SEQUENCE NUMBER -** from this the sender sends the data with the specific sequence number so after receiving the data, receiver sends the data with that sequence number, and here at sender side it also expects the acknowledgment of the same sequence number.

### **Advantages and Disadvantages of Stop-and-Wait Protocol**

**Advantages**

The only advantage of this method of flow control is its simplicity.

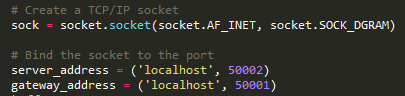
**Disadvantages**

1. The sender needs to wait for the ACK after every frame it transmits. This is a source of inefficiency, and is particularly bad when the propagation delay is much longer than the transmission delay.
2. Stop and wait can also create inefficiencies when sending longer transmissions. When longer transmissions are sent there is more likely chance for error in this protocol. If the messages are short the errors are more likely to be detected early. More inefficiency is created when single messages are broken into separate frames because it makes the transmission longer.

Code Implementation and Analysis

I decided to use this opportunity to learn and perfect my Python Programming Skills. So, I implemented a Stop-and-Wait ARQ in the python programming language. In my own opinion, it made the assignment a lot less complicated and straight-forward.

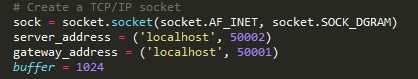
The code below creates a socket. The server port is a fixed value of 5002 and the gateway port is fixed value of 50001.



In order to keep the whole process running continuously I decided to use a while loop. So, while the condition is True, there is constant communication between the sever, the gateway and the client.

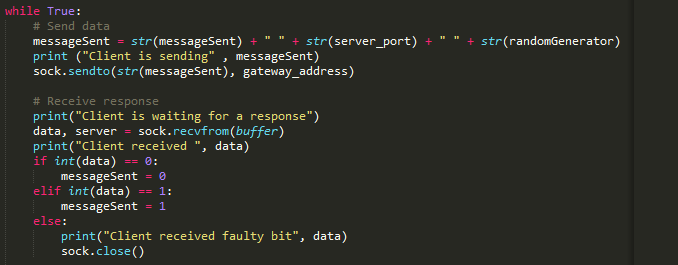
### **The Gateway**

The gateway starts off with a socket creation. It then given an address using the following code.



So once the gateway receives the message from the client it then sends it to the server.

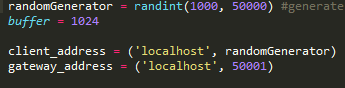
When the client is waiting for a response from the bit has the possibility to be faulty so the following if else if statement takes care of that. If the byte is is faulty the socket is closed immediately.



As part of the assignment as timeOut method was to be implemented to take care of the timing out issue that could possibly arise. This brings order and structure to the Stop-and-Wait protocol. Luckily in Python all that I need to is import the Time library. In my program after every seven seconds the client should have sent a packet to a gateway, the gateway sends the packet to the server, then the server sends an ACK back to client to confirm it received the packet.

### **Multiple Clients**

Another specification is that the program must be able to deal with multiple clients at once. Every client must have its own unique address. The variable randomGenerator generates a random number between 1000 and 50000 and then allocates a random number to the client.



## **Screenshots of the Functioning Program and the Traffic**

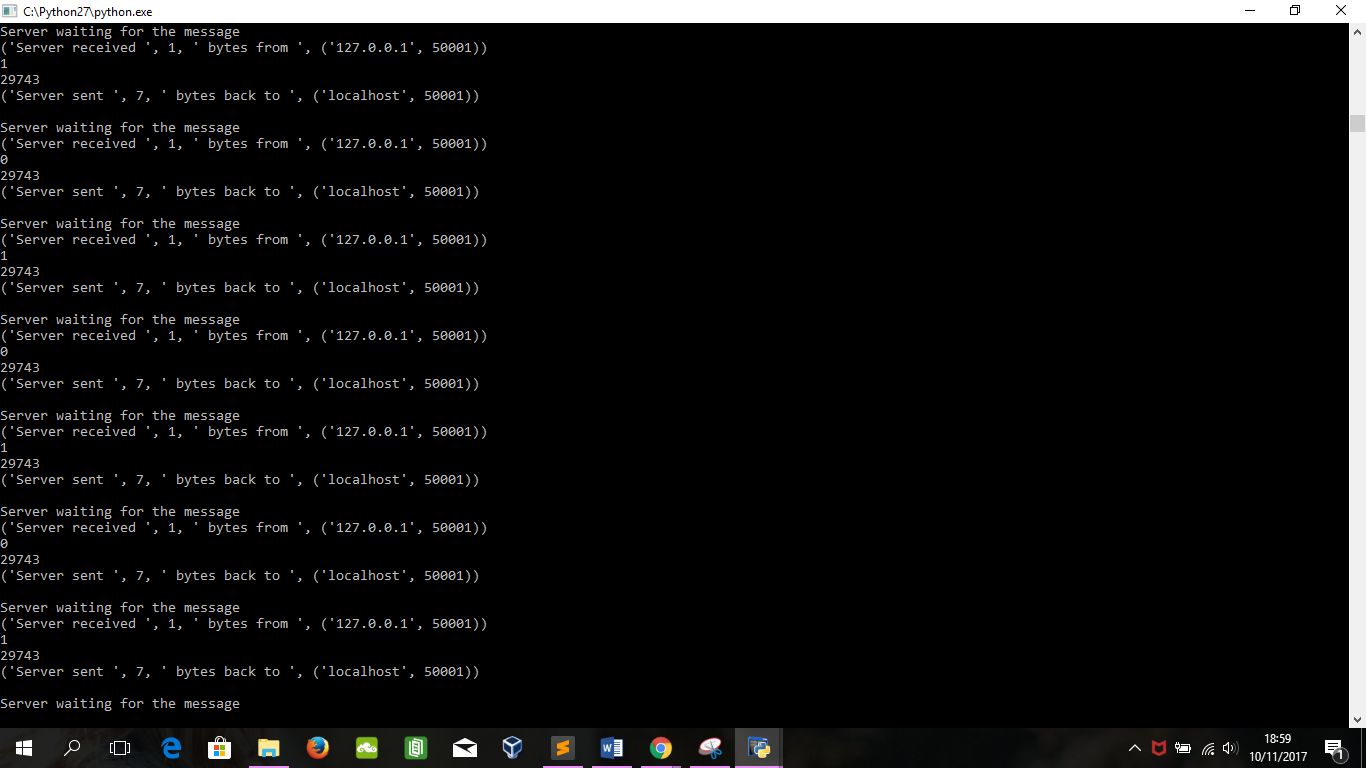
### **Screenshot of Client**

### 

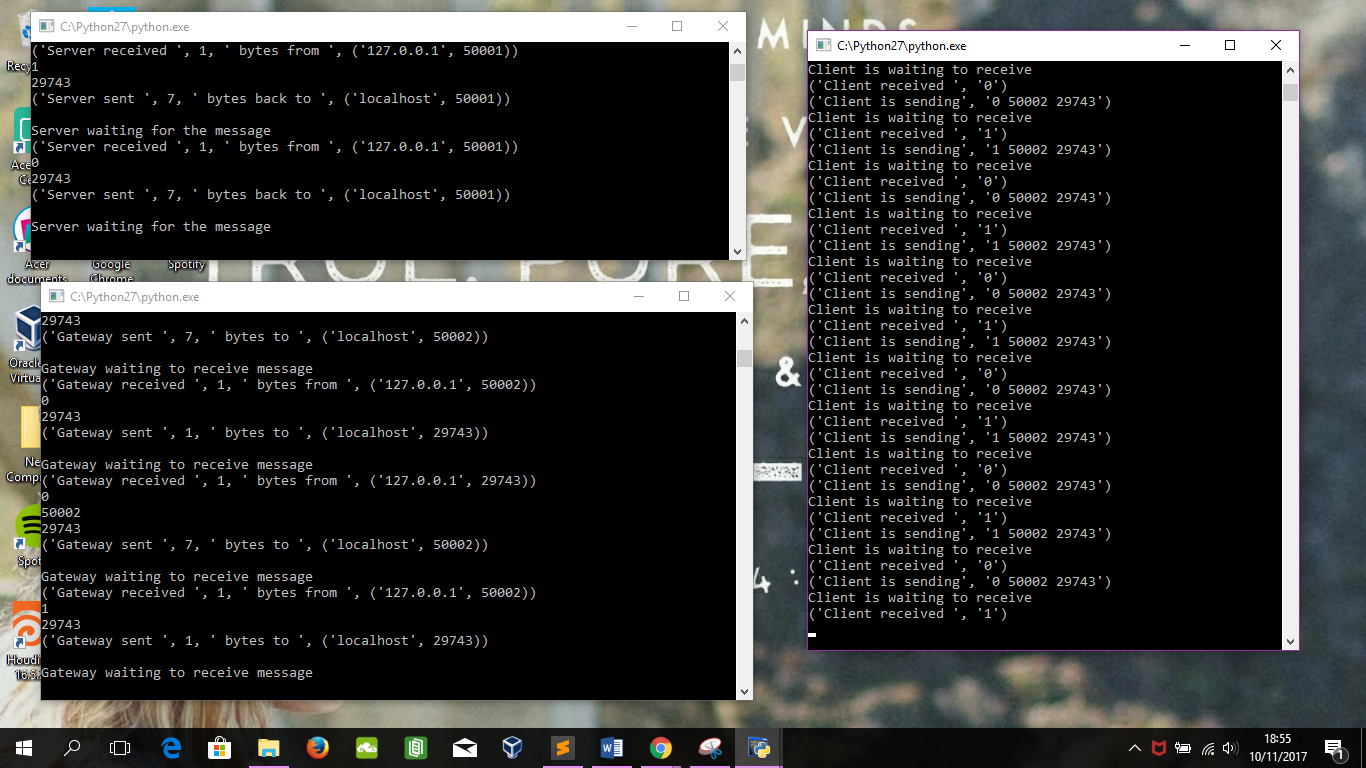
### **Screenshot of Gateway**

### 

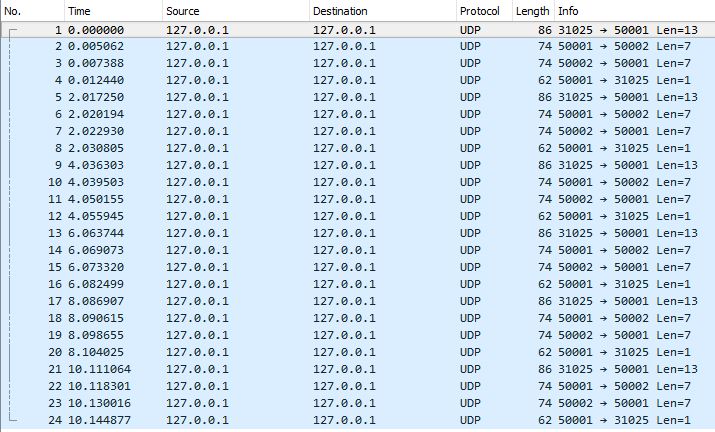
### **Screenshot of Server**

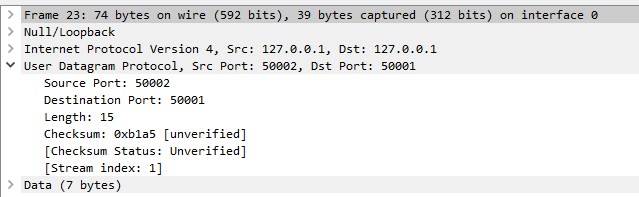


### **Screenshot of only one client, gateway and server**

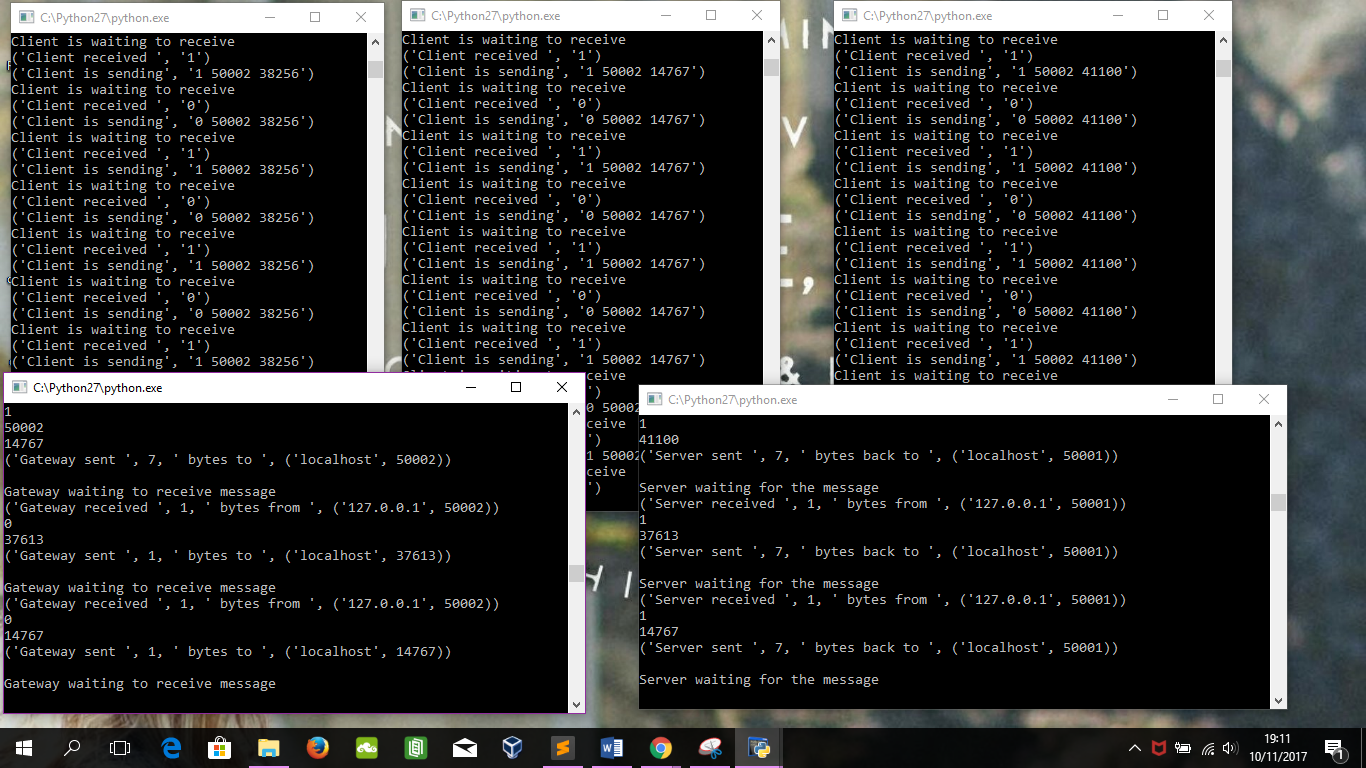


### **Screenshot of traffic of one client, gateway and server**

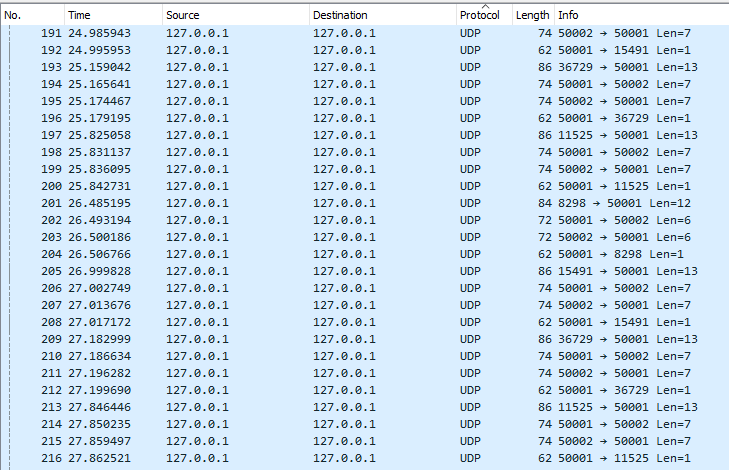


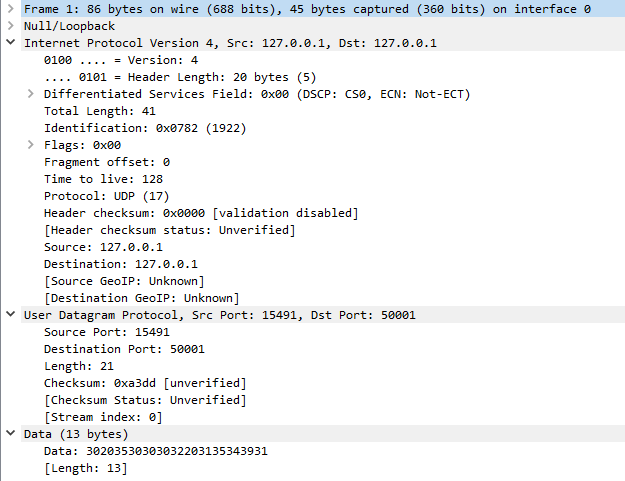


### **Screenshot of only multiple clients, gateway and server**



### **Screenshot of traffic of multiple clients, gateway and server**





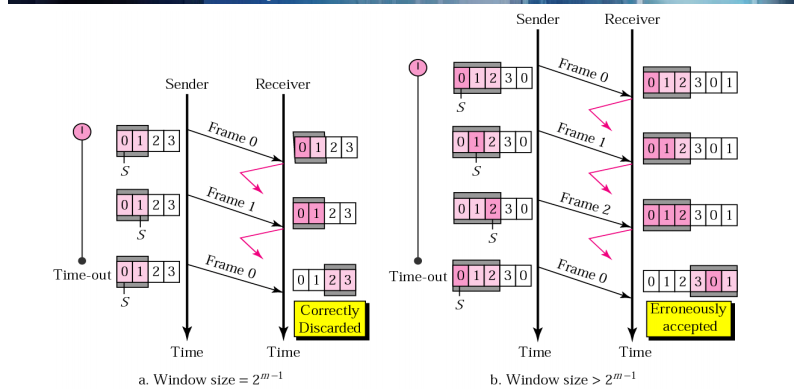
Sliding Window Protocol

Unlike Stop & Wait, Selective Repeat is much more efficient both at filling the pipe (although less so than Go­Back­N) and is particularly useful in a noisy channel as it only requires the sender to resend packets the receiver failed to get.

A method of flow control in which a receiver gives a transmitter permission to transmit data until a window is full. When the window is full, the transmitter must stop transmitting until the receiver advertises a larger window.

The sender can send and the receiver can accept n frames without having to wait for an acknowledgement. A sequence number is assigned to frames in order to help keep track of those frames which did receive an acknowledgement. The receiver acknowledges a frame by sending an acknowledgement that includes the sequence number of the next frame expected. This acknowledgement announces that the receiver is ready to receive n frames, beginning with the number specified. Both the sender and receiver maintain what is called a window. The size of the window is less than or equal to the buffer size.

Sliding window flow control has a far better performance than stop-and-wait flow control. For example, in a wireless environment if data rates are low and noise level is very high, waiting for an acknowledgement for every packet that is transferred is not very feasible. Therefore, transferring data as a bulk would yield a better performance in terms of higher throughput.



### **Selective Repeat Advantages & Disadvantages**

**Advantages**

* Better use of the pipe than Stop & Wait and we can send up to 2^(m­1) packets at a time.
* Much more efficient than Go­Back N in a noisy channel as it only resends a single lost or corrupted frame and not N frames.
* Error detection is easily implemented through redundancy bits added to each frame.

**Disadvantages**

* Most difficult to implement of Stop & Wait and Go­Back N, requires multiple timers, storage of sent frames, both ACK and NAK frames.
* Less efficient use of the pipe than Go­Back N with a send window of (2^m)­1 compared to 2^m for Go­Back N.

References and Reflection

So, the whole assignment was a very interesting journey. At first, I decided to embark on this feat I was planning to implement it in Java. At the time I was learning Python, so I decided to use this as an opportunity to brush up on my Python skills. That made implementation a lot easier. Overall this assignment helped me understand how clients, gateways and servers work and operate thanks to all the extra reading and research. I am aware that my implementation may not be perfect and is definitely not as efficient as it should be but I am satisfied with the outcome.

Here are the links to sites I referenced during this Assignment

* <http://www.arannolan.com/articles/flow-control.html>
* <https://en.wikibooks.org/wiki/Communication_Networks/Error_Control,_Flow_Control,_MAC>
* <https://en.wikipedia.org/wiki/Flow_control_(data)#Stop-and-wait>
* <https://stackoverflow.com/questions/15909064/python-implementation-for-stop-and-wait-algorithm>
* <https://docs.python.org/3/library/asyncio-protocol.html>
* <https://tcd.blackboard.com/bbcswebdav/pid-867490-dt-content-rid-3106353_1/courses/CS2031-A-Y-201718/Slides/CS2031%2002.1%20Flow%20Control.pdf>
* <https://tcd.blackboard.com/bbcswebdav/pid-867469-dt-content-rid-3106319_1/courses/CS2031-A-Y-201718/Assignments/Assignment%201%20Examples/Mark-Collier-Report.pdf>
* <https://en.wikipedia.org/wiki/Gateway_(telecommunications)>
* <https://en.wikipedia.org/wiki/Communications_protocol>
* <http://www.wavelink.com.au/information/gateway-defined.php>
* <http://whatismyipaddress.com/gateway>
* <https://en.wikipedia.org/wiki/Flow_control_(data)>
* <https://en.wikipedia.org/wiki/Stop-and-wait_ARQ>
* <https://en.wikipedia.org/wiki/Sliding_window_protocol>
* <https://en.wikipedia.org/wiki/Go-Back-N_ARQ>
* <https://en.wikipedia.org/wiki/Selective_Repeat_ARQ>
* <https://www.slideshare.net/selvakumar_b1985/flowcontrol-43534944>