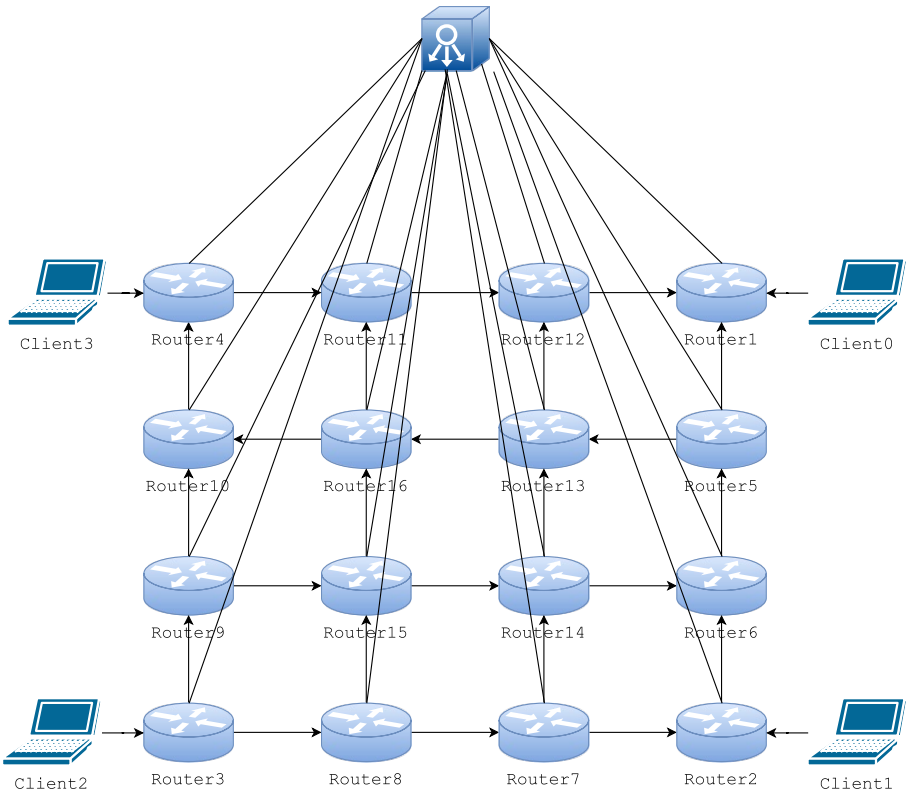
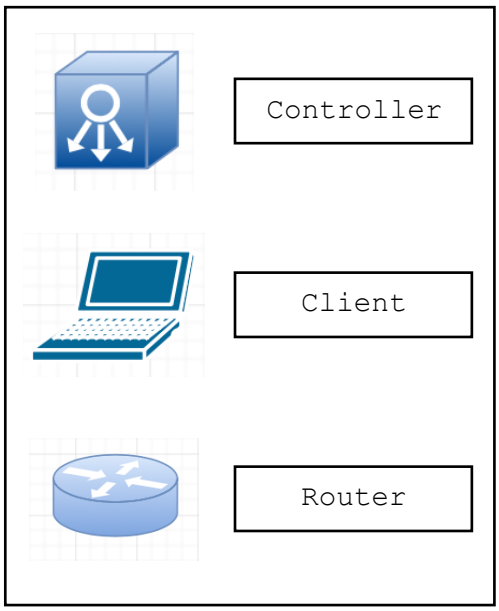
Name: Leong Kai Ler
StudentNumber: 15334636
Lecturer: Dr.Stefan Weber
Course: CS2031 Telecommunication II
Title: Telecommunication II Assignment2 Report
Topic: Alternative to Traditional Routing
Date: 15 December 2017

Introduction to design

This design is a open network that can allow up to 4 end users or clients to engage in. There are a total of 16 routers in it, each with at least 3 to 4 nodes. Each node has its own magnitude of weight or generally known as distance. Only 4 specific routers have one of their nodes connected to one unique client. In other words, one client is only connected to one router in the open network. All paths are preconfigured and calculated by the controller using the Dijkstra's algorithm by Robert Sedgewick and Kevin Wayne from Princeton. The required path information are then forwarded to each router that requires it. The controller is basically rendered obsolete and will no longer need to perform any transmission if all paths have been sent.





Note that the lines between routers or clients and routers are bidirectional. The arrow line is just for distinguishing the connection between routers and connection between routers and controllers. Moreover, these lines may look equal in length, but that does not portrait the idea that the distance between nodes are the same.

Client Class

The client class act as the end points of the network. This idea of of this implementation can actually support as many socket as possible depending on how many available nodes are there in the network. Each client has one port for itself, 3 ports for 3 other users for communication and in addition, 1 port of the router that it is connected to. The connection between the client and its corresponding router is bidirectional. Packets sent from the clients must contain at least 3 of the following information: The data being sent, the port number of the client and destination of the packet. The port numbers are crucial to identify the path of routers that the packet will be taking to be delivered to its destination.

The client only has information of where its target client is, but not how to get there. The router and controller does the navigating job together. Generally speaking, the client class only send and receive packets delivered from other clients as validation of the results.

Advantage:

1. The client has the least functionality effects on the design, which is akin to how our networks works in real life. As physical applications have little impact on delivering information and it is mostly handled in the network layer.
2. This design can handled as many end users as possible, provided that there are enough routers and ports for the additional clients.
3. Able to send unlimited amount of data properly.

Disadvantage:

1. This design does not implement checking procedures on the packet, such as sequence numbers and acknowledgements.
2. Improper management of terminal display makes validation part very messy.

Router and RouterNodes Class

A router is a networking device that forwards data packets between networks. It performs the traffic directing functions on the networking layer. A data packet is typically forwarded from one router to another router through the networks that constitute an internetwork until it reaches its destination node. In this context, the destination node can be any the client in the network.

As mentioned before, each client in the network can only be connected to one router in this design, while the others are merely connecting themselves with other routers and also all of them to the controller. In this design, the routers each consist of 3 or 4 nodes. The approach taken in this assignment is that the nodes in the router share the same port and act as switches or portal entrance. The router is responsible checking the incoming packet and decides how to deal with it. There are mainly three cases:

1. If the packet is not from the controller and at the same time its source and destination requires a path not yet recorded in the router, then the controller has to put the packet transmission on hold, and create a new packet to be send to the controller in request of a new path. This new packet will contain the source and destination of the original packet, so the controller can check and decide which path that it should tell the routers to take.
2. When the packet comes back from the controller, the content of the packet is stored in a HashMap with the source and destination of the packet to be send as its key. At this point, the content is the next router that the current router should assign its node to send the packet to. This allow the subsequent transmission with the same source and destination to propers without asking the controller. It then send the packet to its next router using the right node. This step is only done in one router. This is to prevent the router from instructing the previous packet to the next node and also, to prevent all nodes from sending a packet at the same time, causing duplicates of packet to be sent at the same time around the network path. Eventually, a few copies of packets will reach the destination if this happens. To avoid this, only the router hat has the packet, which is the one that requested a new path from the controller can send the packet. The first two cases are only executed once for each source and destination combination.
3. This is the normal case, which only happens when the first two has. At this point, the router already have the requisite information to direct the packet to which node to be send to the subsequent node. All that the router has to do is to know the source and destination of the packet, look for the corresponding next node and direct the packet to it.

For this implementation of routers, it was proven that using HashMaps within HashMaps to store the next node is impossible, if the key being used is the source and destination of the packet. As one source can send to multiple destinations, or in other words, one client can send to multiple clients. So eventually, the keys will overlap with each other and the HashMaps will be storing the wrong next node for each source and destination. This flaw can be substituted by using a HashMap to store the next node and use the combination of source and destination as string as the key.

Each node in the router can only point to one direction or one router. So its only job is sending packets and it is up to the router to decide which node to use. Each node is represented by a code made from the number of router it is pointing to. This is to assist job of router to check which node to use to send the packet to its next node or router.

One interesting aspect is that only the router connected with the client will be contacting the router.

Advantage:

1. The approach of combining the source and destination of the packet as a string is efficient and every one of them formed is unique.
2. The implementation of router with nodes all sharing the same ports is compatible with the use of Dijkstra algorithm, which will be explained later.
3. Simple router nodes design as a switch, as all the complicated processing are handled by the routers.
4. Use of HashMap to store packets and next nodes

Disadvantage:

1. When requesting a path from controller, the packet can be stored in the request packet that will be sent to the controller as a Point-to-Point Protocol(PPP) approach.

Controller Class

The controller handles the network trafficking by assigning shortest paths to the node. The design used here utilises the Dijkstra's algorithm to figure out the shortest way to transmit packets from different source and destinations between nodes and sends them to the routers. Before the controller is contacted, it has initially formed the a path table of its own using the DijkstraSP, a class imported. This class takes in all the clients (included as it is part of the connection) and router numbers as vertex and the connection of nodes between them as edges. It then formed the shortest paths for every router numbers and clients. As the distances between routers are different weighted, it makes sense to use this class and algorithm.

Dijkstra's algorithm initializes dist[s] to 0 and all other distTo[] entries to positive infinity. Then, it repeatedly relaxes and adds to the tree a non-tree vertex with the lowest distTo[] value, continuing until all vertices are on the tree or no non-tree vertex has a finite distTo[] value. This algorithm solves the single-source shortest-paths problem in edge-weighted digraphs with non-negative weights using extra space proportional to V and time proportional to E log V (in the worst case).The class, DijkstraSP, in addition is an efficient implementation of Dijkstra's algorithm because it uses IndexMinPQ for the priority queue, where an element with high priority is served before an element with low priority.

The text file used in DijkstraSP, EWD.txt contains consists of 5 elements:

1. number of routers as vertices (1st line)
2. number of connections between routers as edges (2nd line)
3. start point of a connection
4. end point of a connection
5. weighted distance of each connection

The connections are represented as one direction instead of bidirectional in this text.

After calculating the shortest paths, the controller will preconfigured them according to the possible source and destination of the client's packet. This is then stored in a simple array of classes called MyKey that can assist in storing the paths formed along with its corresponding source and destination. This class is formed in replace of the functionality of HashMaps. As the only way to access the correct paths is with the correct combination of source and destination as its key. This implementation is quite similar to having a link state table, since one combination of source and destination only offers one path. The paths are in the form of integer arrays comprising of the node numbers in the path.

After pre-configuring the shortest route for each combination of source and destination, the controller is ready to accept requests from the routers. When the controller receives a request from a router, it will check the source and destination in the request packet (not the), and finds the compatible, preconfigured path for it. Then, what it needs to do is just by separating the contents of the path array, and send individual node numbers contained in it as the next router(router[n+1]) for the router[n] to send the packet to. All these individual node numbers are send in packets to the routers separately as direction to the next router. As the packets reach the routers, the node number will be used as reference to find out which node the router should direct the client packet to, in order to pass it to the next router.

After sending all 12 possible paths the controller will no longer be contacted. The controller will also not be contacted by the router asking for the same path again.

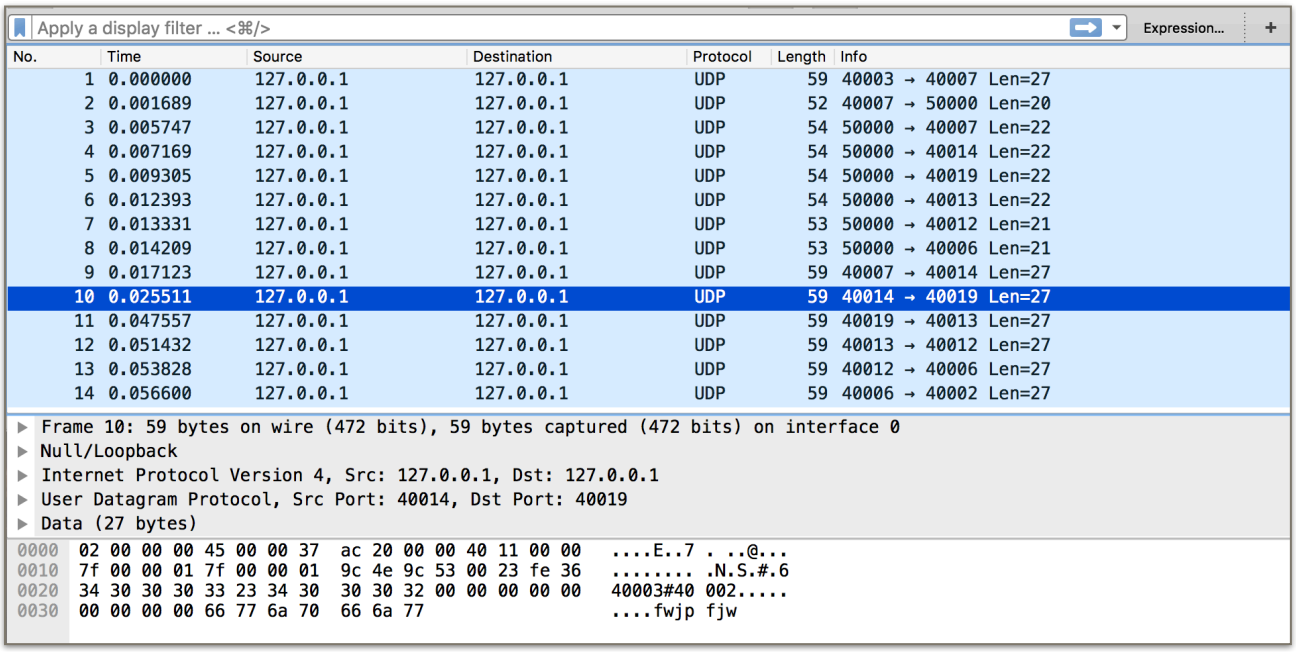
Advantage:

1. The use of Dijkstra Algorithm solves the problem of calculating and configuring the shortest paths between two clients in a router network. This algorithm is efficient even when used in a network of one million vertices and 15,172,126 edges. This means just a controller is capable implementing it can manage a large scale of network. The algorithm is used to preconfigure paths in this design, but in reality, it can also be used to configure paths on the spot, provided that the router numbers are inside the text file used.
2. Use of MyKey in substitute to HashMap to ensure the paths stored is not overlap which is a weakness in using HashMaps for this implementation.

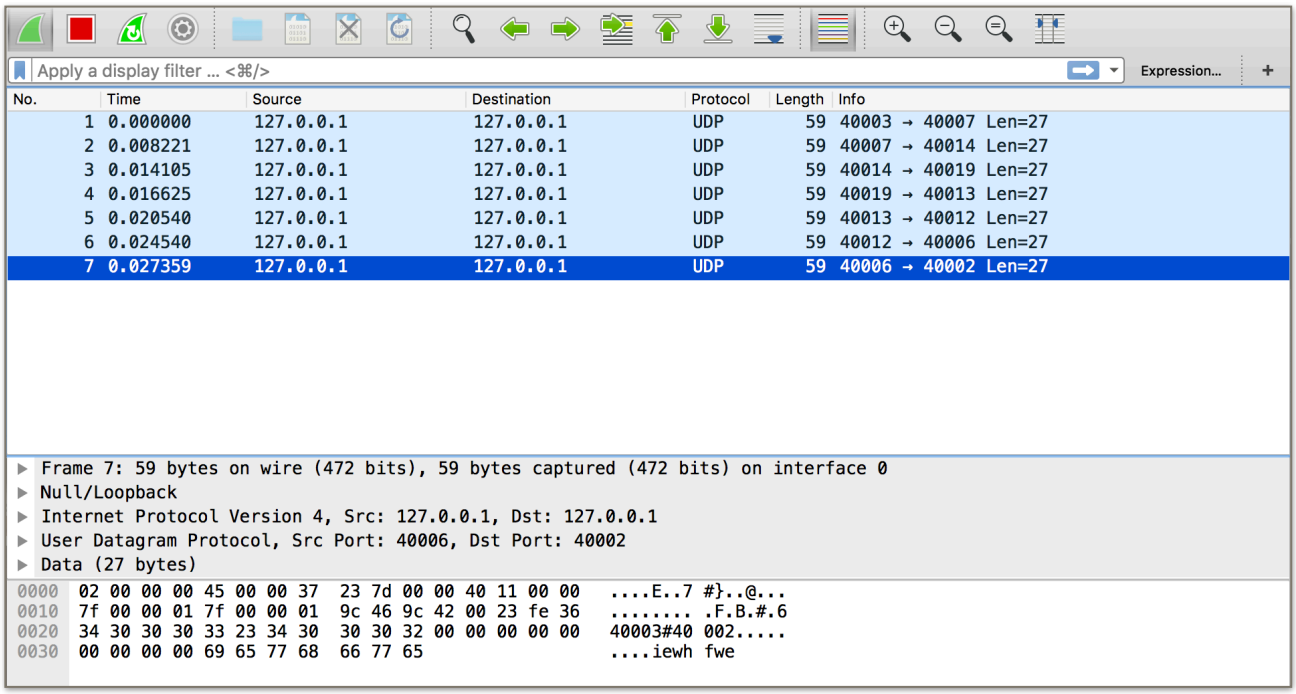
Disadvantage:

1. Undergo too many steps to configure the path and put them into a suitable input for the router to access.

Test and Results1

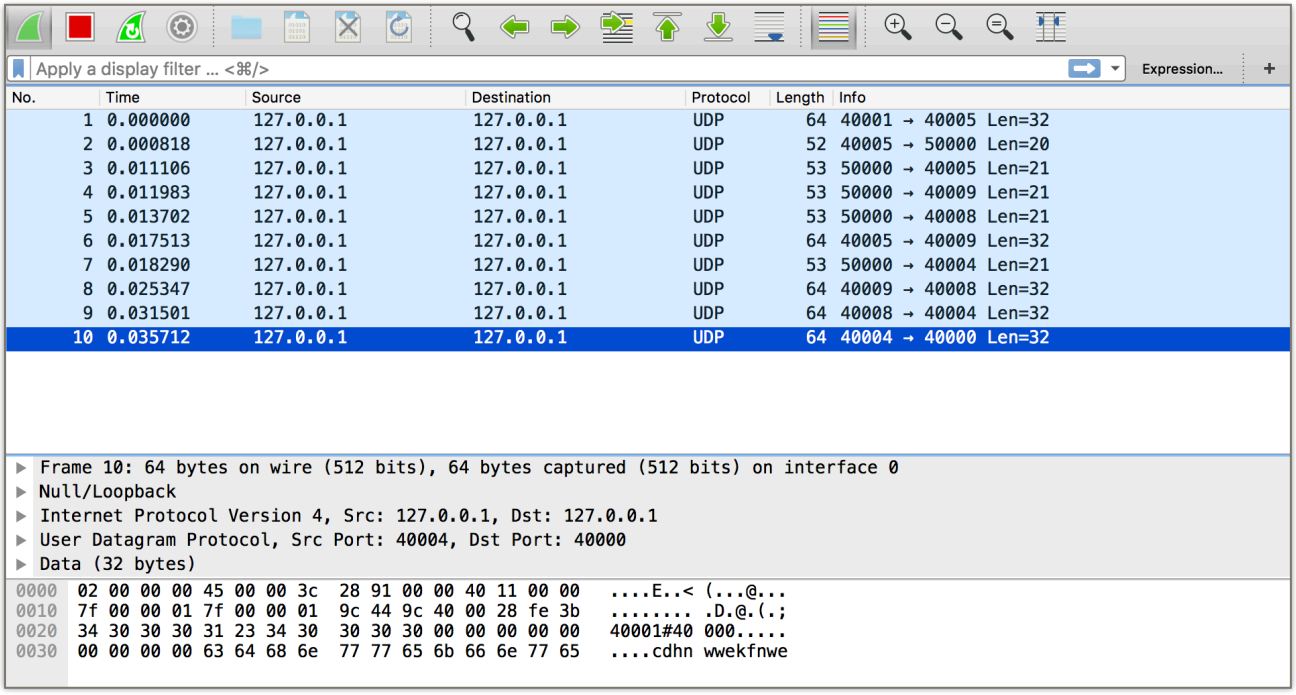


Sending packets from client 3 to client 2 for the first time. As the packets reach the first router, a different packet is being send to the controller with a port of 50000, judging the size of the packet. Then, the controller sends packets back to 6 routers. This forms a path from client3 to client2, which is 40003→40007→40014→40019→40013→40012→40006→40002.

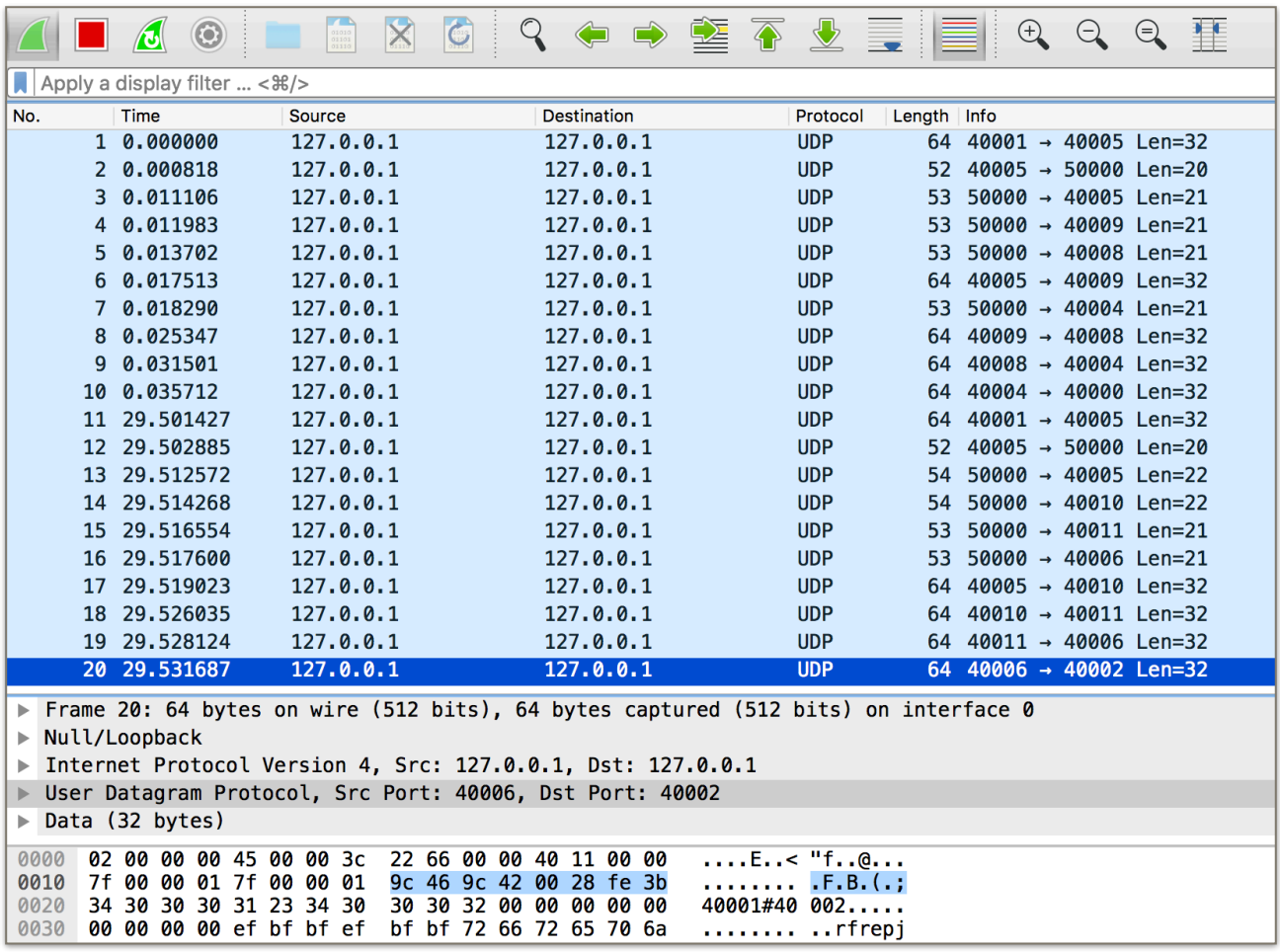


The second time, a packet is sent from client3 to client2, the number of packet transmissions is halved or should be less because there is no need to contact the controller anymore and it does not have to send next nodes to each routers.

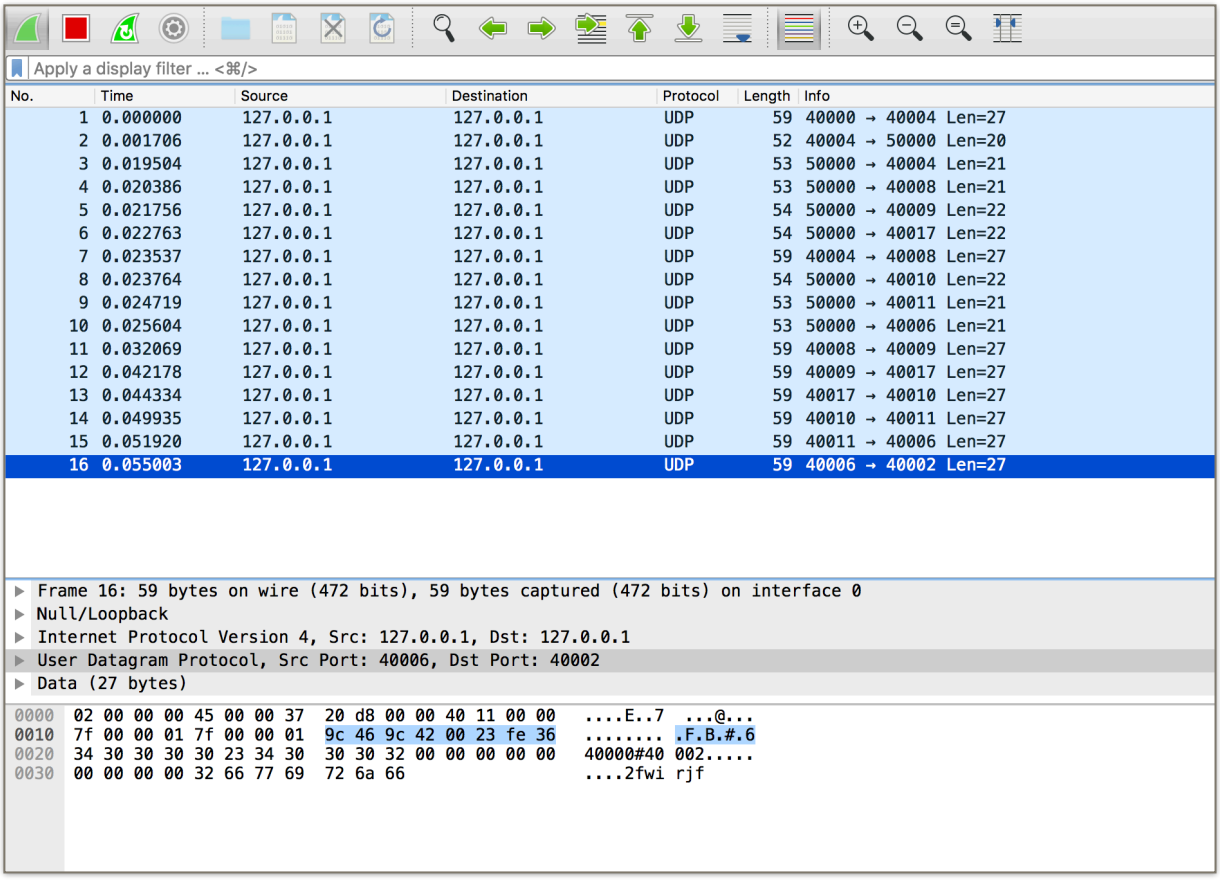
Test and Results2



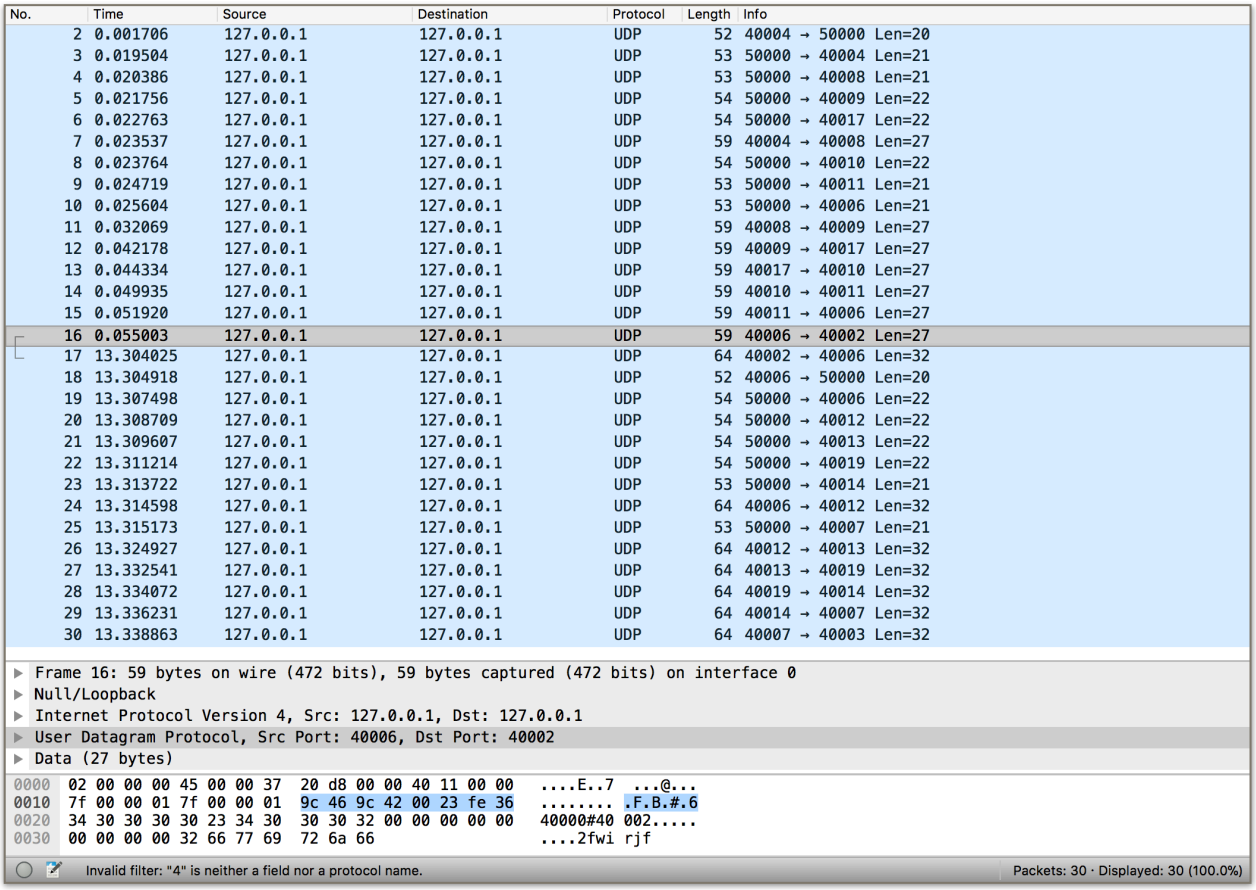
Sending packet from client1 to client0.



Sending packet from client1 to client2.
If there are mistakes in the EWD.text, it will be corrected by the controller.



Sending packets from client0 to client2



Sending packet from client0 to client4.