Ανάπτυξη Εφαρμογής Chat και VoIP

Δίκτυα Υπολογιστών II

Ομάδα W

Κοντογιάννης Χρήστος, Κάτης Σταύρος

1. Αποστολή Μηνυμάτων (Instant Messaging / Chat)

Α. Κώδικας

Για το πρώτο τμήμα της εργασίας υλοποιήθηκε p2p επικοινωνία με ανταλλαγή μηνυμάτων κειμένου με τη χρήση του UDP και ενός αλγόριθμου κρυπτογράφησης σε τοπικό δίκτυο(LAN).

Αρχικά προστέθηκαν οι παρακάτω βιβλιοθήκες στις ήδη υπάρχουσες για να υποστηρίξουν τον αλγόριθμο κρυπτογράφησης.

```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
```

Έπειτα αρχικοποιήθηκαν οι επιπλέον μεταβλητές:

```
InetAddress receiverAddress;
static String KEY = "12345678901234567890123456789012";
static String ALGORITHM = "AES";
static SecretKey SECRET_KEY = new SecretKeySpec(KEY.getBytes(), ALGORITHM);
```

Η receiverAddress θα περιέχει τη διεύθυνση IP του παραλήπτη του μηνύματος που θα ξέρουμε εκ των προτέρων, το KEY το κλειδί κρυπτογράφησης 32 bytes για τον αλγόριθμο(ALGORITHM) AES-256 και η SECRET_KEY τύπου SecretKey που χρησιμοποιείται στους αλγορίθμους κρυπτογράφησης και αποκρυπτογράφησης της κλάσης cipher.

```
DatagramSocket textSocket = null;
try {
	textSocket = new DatagramSocket(5000);
	byte[] buffer = new byte[1024];
	DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
	final DatagramSocket finalTextSocket = textSocket;
	new Thread(() -> {
		while (true) {
			try {
				finalTextSocket.receive(packet);
				byte[] encryptedBytes = new byte[packet.getLength()];
				System.arraycopy(packet.getData(), 0, encryptedBytes, 0,
packet.getLength());
				String decryptedMessage = decrypt(encryptedBytes);
				textArea.append("Received: " + decryptedMessage + "\n");
			}catch (Exception e) {
				e.printStackTrace();
				textArea.append("Error processing incoming message.\n");
			}
		}
    }).start();
} catch (Exception ex) {
	ex.printStackTrace();
	textArea.append("Error initializing textSocket.\n");
}
```

Δημιουργείται ένα DatagramSocket για τη λήψη UDP πακέτων στη θύρα 5000 και χρησιμοποιείται ένα Thread για να εκτελεστούν οι λειτουργίες λήψης πακέτων, αποκρυπτογράφης και εμφάνισης κειμένου παράλληλα. Το textSocket γίνεται τελικό (final), ώστε να μπορεί να χρησιμοποιηθεί μέσα στο ανώνυμο Thread. Αυτό είναι απαραίτητο επειδή οι ανώνυμες κλάσεις απαιτούν οι εξωτερικές μεταβλητές να είναι τελικές.

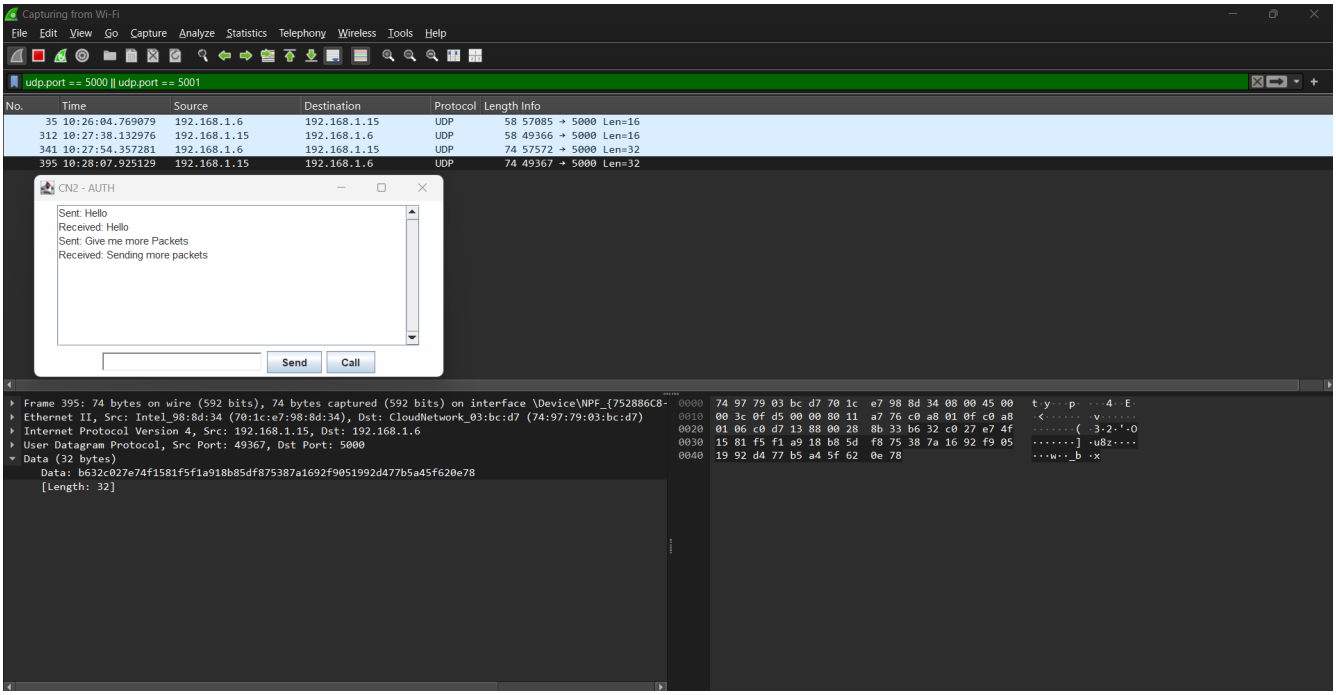
```
if (e.getSource() == sendButton){
	new Thread(() -> {
	// The "Send" button was clicked
		try(DatagramSocket socket = new DatagramSocket()){
			String message = inputTextField.getText();
			byte[] buffer = encrypt(message);
			InetAddress receiverAddress = InetAddress.getByName("192.168.1.15");
			DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
receiverAddress, 5000);
			socket.send(packet);
			textArea.append("Sent: " + inputTextField.getText() + "\n");
		}catch (Exception ex) {
			ex.printStackTrace();
			textArea.append("Error during text transmission.\n");
		}
	}).start();
```

Όταν πατάμε το κουμπί send εκτελούνται οι παραπάνω εντολές. Πρώτα δημιουργείται DatagramSocket με χρήση try-with-resources, που κλείνει αυτόματα, μετά λαμβάνεται το string του κειμένου από το πλαίσιο κειμένου inputTextField, έπειτα κρυπτογραφείται και στέλνεται σαν πακέτο στη διεύθυνση που ορίστηκε και στη θύρα 5000. Τέλος εμφανίζεται το μήνυμα που στείλαμε στην οθόνη μας, για έχουμε πρόσβαση σε όλη την ιστορία του chat.

```
public static byte[] encrypt(String data) throws Exception {
		Cipher cipher = Cipher.getInstance(ALGORITHM);
		cipher.init(Cipher.ENCRYPT_MODE, SECRET_KEY);
		return cipher.doFinal(data.getBytes());
}
public static String decrypt(byte[] data) throws Exception {
		Cipher cipher = Cipher.getInstance(ALGORITHM);
		cipher.init(Cipher.DECRYPT_MODE, SECRET_KEY);
		byte[] decryptedBytes = cipher.doFinal(data);
		return new String(decryptedBytes);
}
```

Η πρώτη μέθοδος κρυπτογράφησης δημιουργεί αντικείμενο cipher για αλγόριθμο AES, μετά το αρχικοποιεί σε λειτουργία κρυπτογράφησης και μετατρέπει το String Data σε byte array. Η δεύτερη μέθοδος αποκρυπτογράφησης αποκρυπτογραφεί το byte array στο αρχικό μήνυμα που στάλθηκε από τον άλλο χρήστη.

B. Wireshark



Εδώ βλέπουμε τα πακέτα κειμένου που στάλθηκαν μεταξύ των δύο υπολογιστών καθώς το payload σε δεκαεξαδική και μορφή κωδικοποιημένου κειμένου.

2. Αποστολή Ηχητικών Μηνυμάτων(VoIP)

Α. Κώδικας

```
new Thread(() -> {
	DatagramSocket audioSocket = null;
	try {
		audioSocket = new DatagramSocket(5001);
		AudioFormat format = new AudioFormat(8000.0f, 8, 1, true, true);
		SourceDataLine speaker = AudioSystem.getSourceDataLine(format);
		speaker.open(format);
		speaker.start();
		byte[] buffer = new byte[1024];
		DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
		while (true) {
			audioSocket.receive(packet);
			speaker.write(packet.getData(), 0, packet.getLength());
		}
	}catch (Exception ex) {
		ex.printStackTrace();
	} finally {
		if (audioSocket != null && !audioSocket.isClosed()) {
		audioSocket.close();
		}
	}
}).start();
```

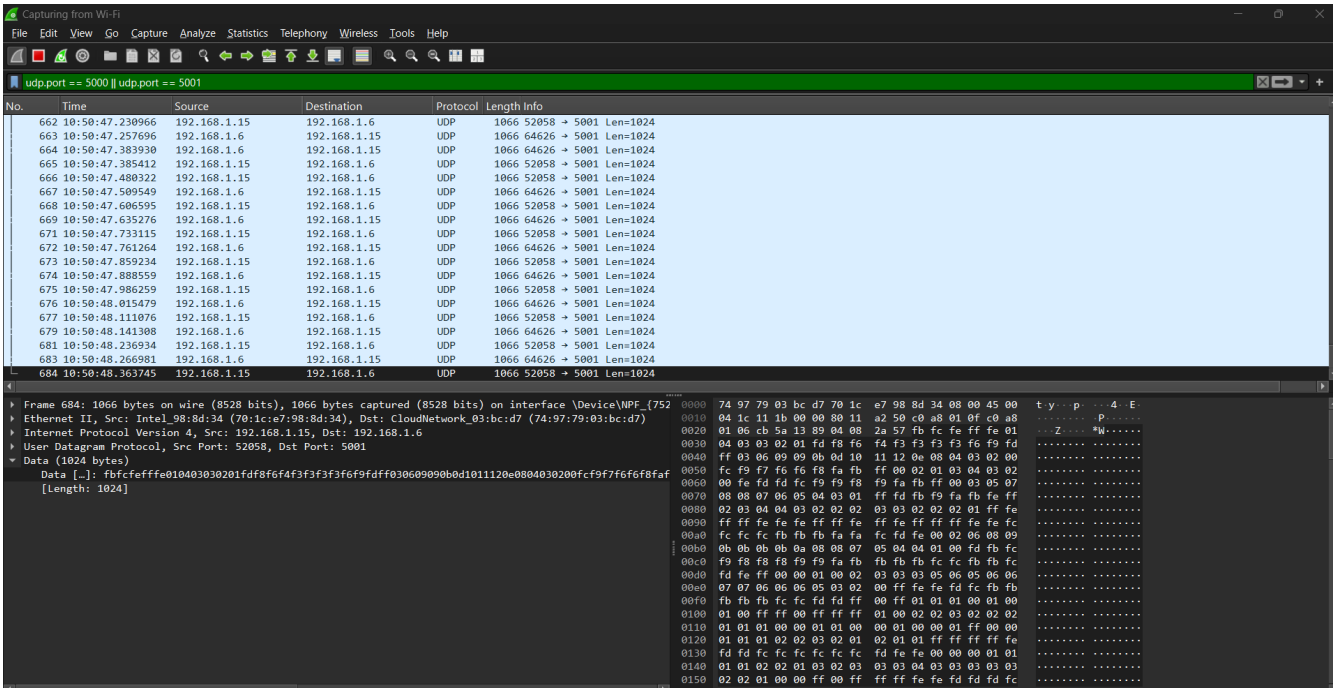
Δημιουργείται ένα αντικείμενο DatagramSocket που ακούει στη θύρα 5001 σε ένα ανώνυμο νήμα. Το AudioFormat ορίζει τη μορφή του ήχου που θα αναπαραχθεί. Υλοποιείται μονοφωνική παλμοκωδική διαμόρφωση ήχου PCM (pulse code modulation) με χαρακτηριστικά της εκφώνησης. Το αντικείμενο SourceDataLine συνδέεται με τα ηχεία. Ο βρόγχος while επιτρέπει τη συνεχή λήψη πακέτων. Ο βρόγχος finally εκτελείται ανεξάρτητα αν πεταχτεί exception.

```
}else if(e.getSource() == callButton){
	X = true;
	if (Call != null && Call.isAlive()) {
		Call.interrupt();
		X = false;
		textArea.append("Call stopped.\n");
		return;
	}else{
		X = true;
	}
	Call = new Thread(() -> {
		textArea.append("Starting call...\n");
		try(DatagramSocket socket = new DatagramSocket()) {
			AudioFormat format = new AudioFormat(8000.0f, 8, 1, true, true);
			TargetDataLine microphone = AudioSystem.getTargetDataLine(format);
			microphone.open(format);
			microphone.start();
			byte[] buffer = new byte[1024];
			DatagramPacket packet;
			InetAddress receiverAddress = InetAddress.getByName("192.168.1.15");
			while (X) {
				int bytesRead = microphone.read(buffer, 0, buffer.length);
				boolean detectsSound = false;
```

```
                        for (int i = 0; i < bytesRead; i++) {
                              if (buffer[i] != 0) {
                                    detectsSound = true;
                                    break;
                              }
                        }
                        if (detectsSound) {
                              packet = new DatagramPacket(buffer, bytesRead,
receiverAddress, 5001);
                              socket.send(packet);
                        }
                  }
                  microphone.stop();
                  microphone.close();
            }catch (Exception ex) {
                  ex.printStackTrace();
                  textArea.append("Error during audio transmission.\n");
            }
      });
      Call.start();
}
```

Εδώ υλοποιείται η διαχείριση του κουμπιού Call για την έναρξη και τη διακοπή μιας κλήσης. Χρησιμοποιείται global νήμα Call για να στέλνει συνεχώς δεδομένα ήχου μέσω UDP, ενώ διαχειρίζεται τη διακοπή του νήματος όταν το κουμπί πατηθεί ξανά. Ο βρόγχος if σταματάει το νήμα Call αν αυτό τρέχει ήδη και θέλουμε να σταματήσουμε να στέλνουμε πακέτα πατώντας το κουμπί για δεύτερη φορά. Έπειτα δημιουργείται νέο νήμα για την κλήση και ρυθμίζεται το μικρόφωνο. Στη συνέχεια λειτουργεί βρόγχος while μέχρι να τερματιστεί η call. Όταν γίνει interrupt του νήματος εκτελούνται οι microsoft.stop και .close, που σταματούν το μικρόφωνο και ελευθερώνουν τους πόρους.

B. Wireshark



Τα ηχητικά πακέτα που στάλθηκαν μεταξύ των δύο υπολογιστών καθώς το payload σε δεκαεξαδική και μορφή κωδικοποιημένου κειμένου.