ST AUGUSTINE'S
COLLEGE - SYDNEY

Nodepaths
Design Documentation - Task 2

By Alex Greig

May 8, 2021

Contents

1 Project Portfolio

1.1 Problem Definition

The problem with today's internet is the control large corporations have over information transmission, enabling data to be lost, monetized, or given to the government, all without users knowing. Another problem of the internet is the inherent security flaws associated with centralised control; that is, its ability to be hacked. The final problem is the inequality of the internet as large populations throughout the world are unable to access the internet, with access they would be able to receive better educational resources, progressing humanity forward.

The application that I am creating is called Nodepaths, and it is important for the consumer market as it will provide high security to user's data and a network that is robust and resistant to problems allowing for secure data transfer between devices globally, solving the problems above. The application will have a significant effect on the way we use the internet and allow users to share, communicate and complete complex computational tasks efficiently and securely. The application will be a decentralised network that uses peer-to-peer architecture and mesh topology, allowing any machine capable of connecting to a network to join and become a "node". The application will also include the feature of Wi-Fi peer to peer transmission allowing for nodes to be connected by Wi-Fi radio waves eliminating costs of telecommunication services, however, if users cannot be connected by the mesh then cellular internet is available. This information will be encrypted, end to end. The mesh network will relay data and messages using a wireless ad-hoc network, where data is propagated along a path by hopping from node to node until it reaches its destination. The paths will be calculated based on the Ad-Hoc On Demand Vector Routing protocol (AODV), a routing protocol that determines and maintains routes whenever they are required.

The application that I am creating, Nodepaths, will have three main functions. The first is creating a user, to do this it will need to create identification. The application will create a public and private key using the Curve25519 elliptic curve in conjunction with the Diffie–Hellman key agreement scheme and the Advanced Encryption Standard, then a psuedo-random IPv4 Address and finally validate a username that has been entered by the user. The second function of Nodepaths is to transfer data, normally messages, over a wireless ad-hoc network to another node. The final main function of the application will be to give users the ability to add new nodes (friends) to the application allowing them to select friends that they want to text to. These three functions when integrated will provide the foundation for a user-friendly text messaging application that runs on a decentralised, distributed network. Although the functions may seem simple or basic the underlying backend behind a network of this caliber is complex; to function, innovative solutions are needed.

Future improvements and upgrades to the application will include the implementation of Blockchain technology to provide digital transactions and a cryptographic wallet on the application, eliminating the need for banks and other financial institutions. In the future, the application may also provide assistance to disadvantaged communities as it would give them communication without the cost of telecommunication services.

1.2 Context and Data-flow Diagrams

Figure 1: Context Diagram

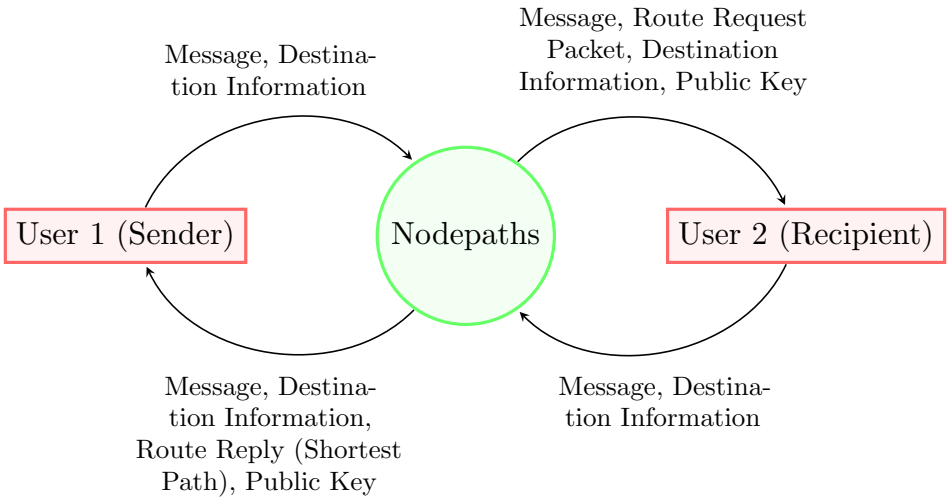
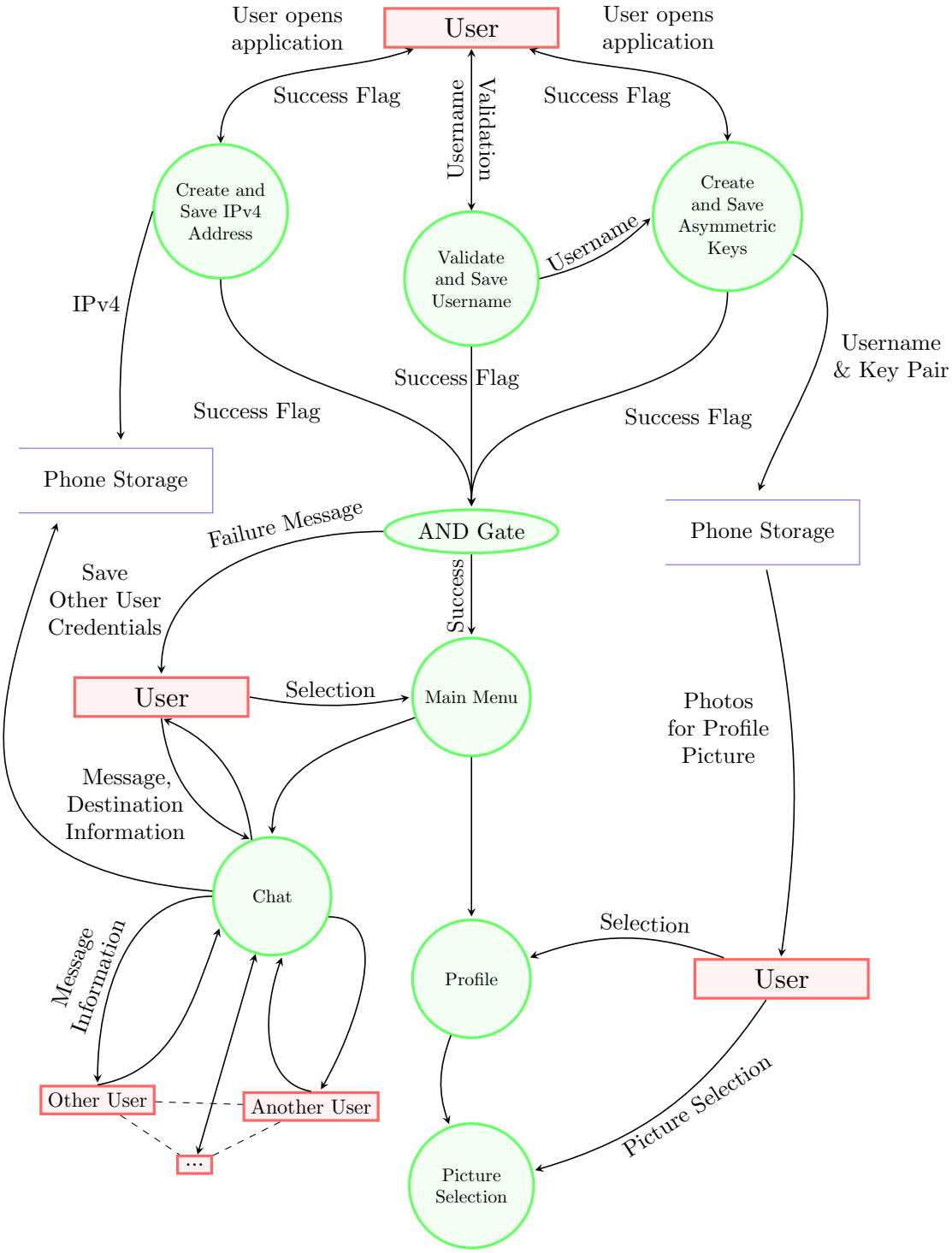


Figure 2: Data Flow Diagram



1.3 Algorithms

The core algorithms that will be used in the application are displayed below, written in the pseudocode syntax:

Algorithm 1: Main Program

```
1  BEGIN MAINPROGRAM
2     NodeCreation()
3     DISPLAY defaultUI() //Buttons and backgrounds that are common
          to all screens
4     conversationList as Array of Conversations //This includes
          node, recent messages, user info
5     OPEN "Phone Storage" for OUTPUT
6        WHILE <> EOF
7           line = READ line from "Phone Storage/
                PreviousConversations.txt"
8           APPEND line to conversationList
9        END WHILE
10    CLOSE "Phone Storage"
11    DISPLAY conversationButtons(conversationList) // Creates the
          buttons with previous information
12    userPressQuit = False
13    currentPage = messages
14    WHILE userPressQuit = False DO
15       IF UserPressedButton() == True:
16          IF userInput.button() == Quit
17             userPressQuit = True
18          ENDIF
19          ELSE IF userInput.button() == Plus \\Top right-hand
                corner of screen
20             DISPLAY screen(NewNode)
21             IF userInput == "Search Network" THEN
22                nodes = SendData(RREQ, allUsers)
23                DISPLAY text "You have found these [nodes] around
                      you" //listing the paths (nodes, friends) that
                       returned from the sub-routine.
24             ENDIF
25          ELSE IF userInput.button() == MenuBar
26             DISPLAY screen(userInput.menuBarType()) \\ Will
                   Display the other screens that are part of the
                   application when they are selected on the menu
                   bar.
27             IF userInput.menuBarType() == Settings AND
                   userSelection == Profile Picture THEN
28                userPicture = selectPicture()
29             ENDIF
30          ELSE IF userInput.button == Conversation
31             DISPLAY screen(NodeMessage(UserID))
32             IF UserSendsMessage == True THEN
33                INPUT message
34                SendData(message, userID)
35             ENDIF
36          ELSE
37             DISPLAY Error("Couldn't identitfy the user input")
```

```
38          ENDIF
39       ELSE
40          PASS
41       ENDIF
42    END WHILE
43 END MAINPROGRAM
```

The main program above displays different processes, inputs and outputs that would be in the real program, however, this code isn't a full replication of the intended end product. If I was to create this application in full I would have more functions that separate the program into smaller pieces, making it easier to read and write. The first difference from the code above is I would separate all the screens into different functions however as previously mentioned this is impractical for the intention of the task.

Algorithm 2: Node Creation

```
1  BEGIN SUBPROGRAM NodeCreation
2
3     success = False
4     DISPLAY landing_page() // Displays the landing page with
          username input field (First UI Screen)
5     WHILE success == False DO
6        success = True
7        INPUT username
8        //Validates Length
9        IF username.len() >= 5 OR username.len() <= 20 THEN
10          success = True
11       ELSE
12          success = False
13       ENDIF
14       // Checks for special characters within username
15       IF username.contains_special_chars() == False THEN
16          success = True
17       ELSE
18          success = False
19       ENDIF
20    END WHILE
21    ipv4 = IPv4Gen()
22    EccKey = EccKeyGen()
23
24    OPEN "Phone Storage" for INPUT
25       WRITE username, ipv4, EccKey
26    CLOSE "Phone Storage"
27 END SUBPROGRAM NodeCreation
```

Algorithm 3: Elliptic-Curve Cryptographic (ECC) Key Generation

```
1  BEGIN SUBPROGRAM EccKeyGen
2
3     curve = get_curve('secp256k1')
4     k = rand_num(0, 627710173538668) //Parameters is the starting
          and ending range of the random number
```

```
5     privKey = k
6     pubKey = privKey * curve.g
7
8     RETURN privKey, pubKey
9  END SUBPROGRAM EccKeyGen
```

The Elliptic-Curve Cryptographic Algorithm is efficient and as fast as simply creating a random number, however, it allows for a private and public key to be created. The algorithm itself is based on an elliptic curve which are created through the equation:

$$y^2 = x^3 + ax + b$$

Due to this equation given a curve which has infinite points, we will limit the curve to a finite field. To do this we use modular arithmetic which transforms the equation into the form:

$$y^2 = x^3 + ax + b(mod\ p)$$

where p is a prime number between 3 and 2^{256}. This is a general equation for an elliptic graph however certain specific equations have been named such as secp256k1 (used in Bitcoin), which is

$$y^2 = x^3 + 7(mod\ 17)$$

To check whether a point, for example $P(3, 4)$, lies on this curve you substitute values in to x and y in the equation. A point G over an elliptic curve can be multiplied by an integer k and the result is another EC point P on the same curve and this operation is fast. This is the basis behind the key generation, we multiply a fixed EC point G (the generator point) by certain integer k (k can be considered as private key), we obtain an EC point P (its corresponding public key). It is very fast to calculate $P = k * G$ and extremely slow (considered infeasible for large k) to calculate $k = \frac{P}{G}$.

Algorithm 4: IPv4 Creation

```
1  BEGIN SUBPROGRAM IPv4Gen
2
3  base_ip = "192.168"
4  FOR i = 0 to 1 DO
5     APPEND random_int(0, 255) to base_ip
6  NEXT i
7  END FOR
8  RETURN base_ip
9  END SUBPROGRAM IPv4Gen
```

Algorithm 5: Send Data Packets over Network

```
1  BEGIN SUBPROGRAM SendData(Message, Users)
2     transmitter = WifiPeertoPeerApi()
3     OPEN "Phone Storage" for OUTPUT
4     WHILE <> EOF
5        READ line from "Phone Storage/User Data"
6        IF line.starts_with() == (ipv4 OR EccKey OR Username)
7           Append line to UserInformation
8        ENDIF
9     END WHILE
```

```
10      transmitter.perform_handshake(Users)
11      transmitter.send_udp_packet(RREQ, Users, UserInformation) //
            finding the shortest path to the designated user
12      transmitter.send_udp_packet(Message, User, UserInformation)
13      If transmitter.recieves_reply()
14          RETURN transmitter.reply()
15      ENDIF
16  END SUBPROGRAM SendData
```

Algorithm 6: Select Profile Picture

```
1  BEGIN SUBPROGRAM selectPicture
2      OPEN DefaultPhotoViewingApp on DeviceScreen
3      picture = DefaultPhotoViewingApp.selection()
4      CLOSE DefaultPhotoViewingApp on DeviceScreen
5      RETURN picture
6  END SUBPROGRAM selectPicture
```

1.4 Test data and expected outputs

Input	Processing	Output
The user opens the application for the first time, and they are asked to input a username.	The application calculates a pseudo-random number with a uniform distribution to create a IPv4 address in the range of 192.168.1.0 to 192.168.254.255. It then creates an asymmetric key pair, a private and public key that will be used for identification of nodes and end-to-end encryption of data transmissions. It then takes the username that was inputed and validates it, checking that the length is between 5 and 20 characters and doesn't contain any special characters. The username, the IPv4 address and the key pair are stored securely in the phone storage	If a username was inputted that doesn't meet the required validation then the user will be redirected to enter another username. A loading screen is displayed with text saying, "Setting up Profile and Configuration." This happens while waiting for operations to be complete, then welcome text appears when done and a main menu appears.

The user selects the option on the main menu, "Search for users on the network."	Using the mobile phone's API for peer-to-peer Wi-Fi connections it sends out a Wi-Fi signal and also listens for incoming Wi-Fi connections. It sends a data packet on the network containing the username and searches for users with the same name. If there are multiple users with the same username then the user can select the profile they want. The two nodes establish a connection and communication is available	The user is prompted with the different users that have the username they searched, it then displays the ability to add them. If they add a node as a friend, the text, "Successfully added [username] as a friend" will be displayed.
The user scans a QR code of another user.	The QR code is a representation of the public key, IP address and username of a user and when it is scanned, it is saved onto the mobile phone's storage	Display text, "Successfully added [username] as a friend"
When in the chat section of the application the user selects a user to message, they then send a text to them.	The application, utilising the ad-hoc on-demand vector routing protocol will flood the network with Route Request Packets via UDP, if the route is found then a Route Reply packet will be sent, this contains the shortest path between users. When the path is created using TCP, the public keys will be exchanged. The message will be encrypted with the other user's public key and transmitted across the network. The message will be validated to make sure the packet size isn't larger than 100 MB.	Once the message has been sent, the message on the user's screen goes blue, displaying to the user that the message has been transmitted.

Within the profile section of the application the user can select to add a photo to their profile.	The application will open the default picture viewing application on the phone and allow the user to select a picture. The user can scale the photo to fit inside a circle with diameter of 1000 pixels.	The default photo viewing application will be displayed, then the text, "Successfully added the profile picture," will be displayed once the picture is scaled and fitted to the profile picture template.