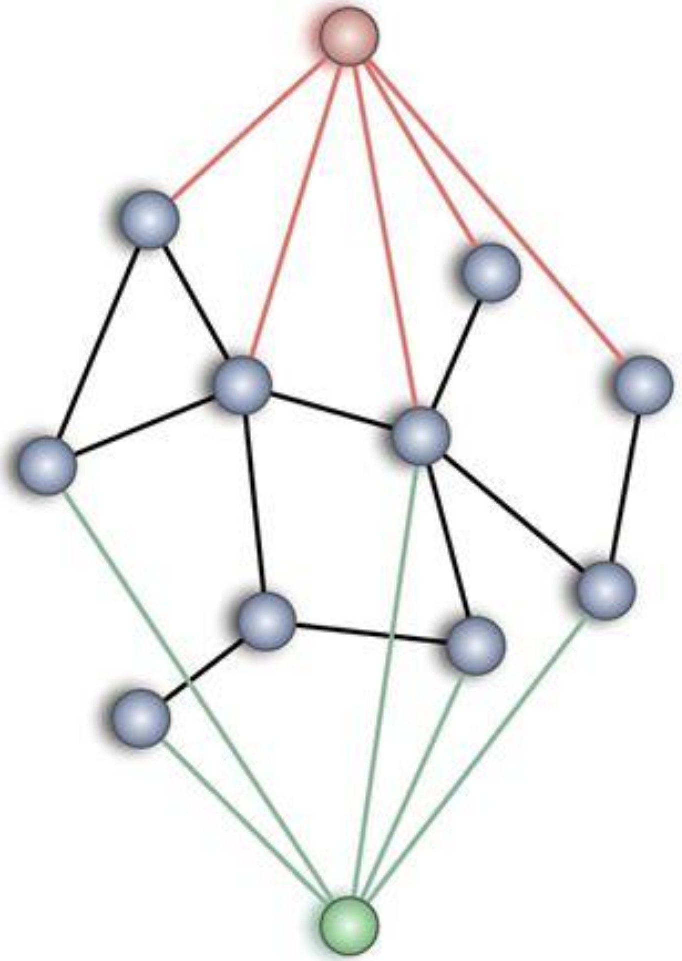
Ad-hoc Messaging

Michael Alvarez
Luke Beukelman
Ben Breisch

Outline

- Project Motivation & Background
- System Architecture
- Design + Implementation
- Technical Challenges
- Demo
- Testing and Performance
- Challenges
- Security and Privacy
- Future Work

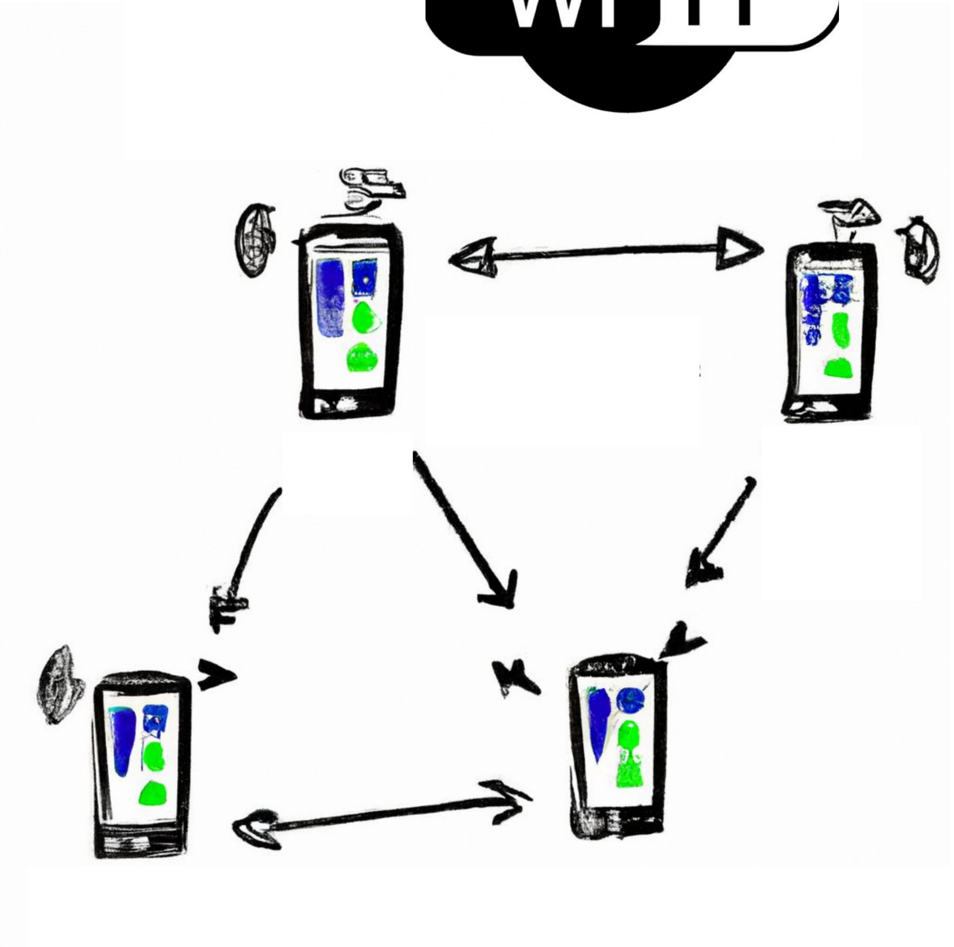


Project Motivation

- Most modern messaging apps require existing network service.
- In disaster areas, network infrastructure may not exist or be damaged.
- Crowded events often lack bandwidth for communication
- Existing technologies for Ad-hoc networking:
  - AirDrop (Apple)
  - WiFi Direct (Everything except Apple)
- These have not been utilized in popular messaging applications.

What is WiFi Direct?

- Method for two WiFi devices to communicate directly without connecting to an existing network or AP
- Android's WiFi P2P API
  - Discover peer devices
  - Connect by forming 'groups'
  - Network (TCP) sockets to communicate



Related Work

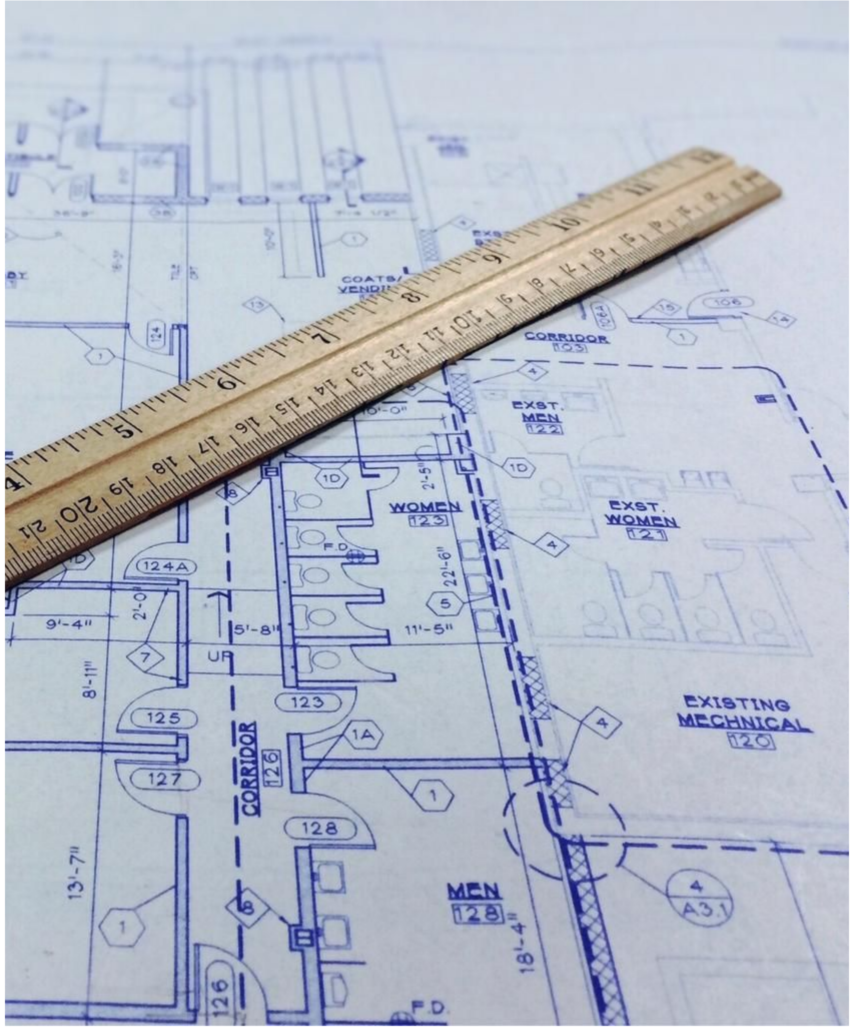
- Existing implementations of ad-hoc “messaging” using:
    - Bluetooth (Bridgefy)
        - Broadcasts messages up to 100 meters.
        - Works on Android and IOS.
        - Direct messages are encrypted.
        - Uses mesh networking, so messages can send across multiple devices.
    - Multi-protocol - AP Mode (Briar)
        - On Android.
        - Uses Bluetooth, and uses Tor style networking.
        - Messages are stored, and transmit to new neighbors when a node travels.
    - Multi-protocol (Airdrop)
        - Designed for file transfer, not for messaging.
        - Uses Bluetooth for discovery and connection creation.
        - Transmission of data uses Wi-Fi.

System Architecture

- Using Wi-Fi Direct (802.11)
- Implemented in Kotlin on Android Phones

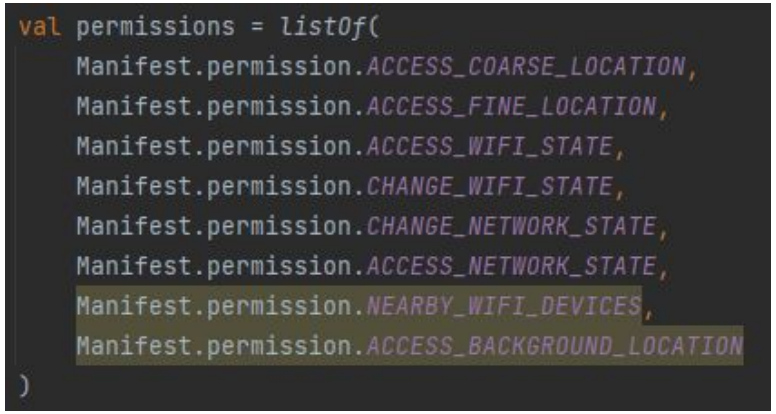
—

- Every device advertises a name and MAC address.
- When an attempt to connect is made, both devices must consent to messaging.
- Messages are stored locally

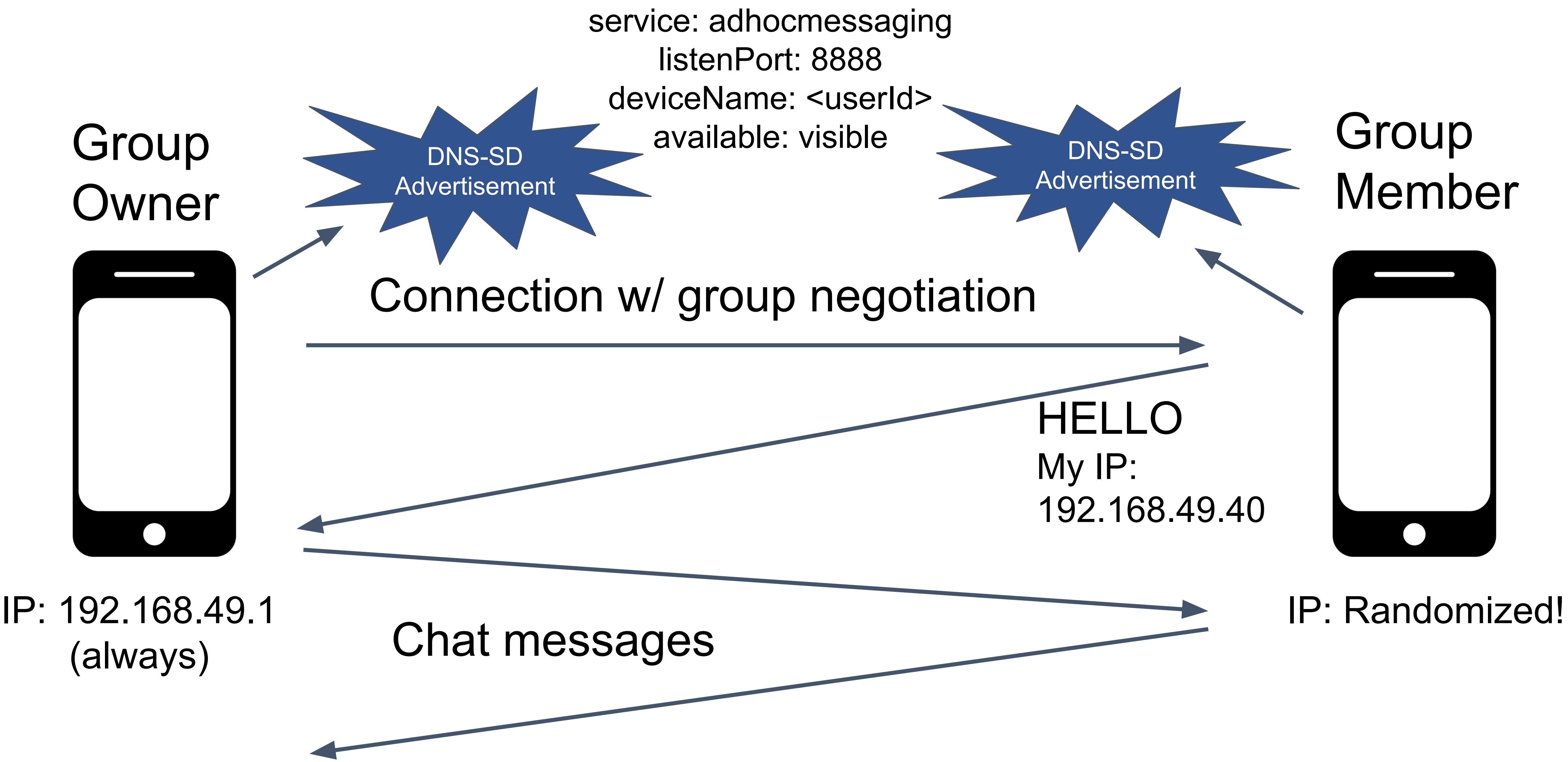


Android App Development!

- App developed in Kotlin w/ Android Studio
  - Access to Android emulator, debugger
- Composables - reusable units/functions
  - Screens, cards, buttons, text boxes, etc.
- Many asynchronous function calls, multithreading
  - Separate threads for main UI, Wi-Fi Direct, Server, Client sockets
- Permissions - Wi-Fi P2P API requires most of them



```
val permissions = listOf(
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.ACCESS_FINE_LOCATION,
    Manifest.permission.ACCESS_WIFI_STATE,
    Manifest.permission.CHANGE_WIFI_STATE,
    Manifest.permission.CHANGE_NETWORK_STATE,
    Manifest.permission.ACCESS_NETWORK_STATE,
    Manifest.permission.NEARBY_WIFI_DEVICES,
    Manifest.permission.ACCESS_BACKGROUND_LOCATION
)
```



Group Owner

service: adhocmessaging
listenPort: 8888
deviceName: <userId>
available: visible

Group Member

Connection w/ group negotiation

HELLO
My IP:
192.168.49.40

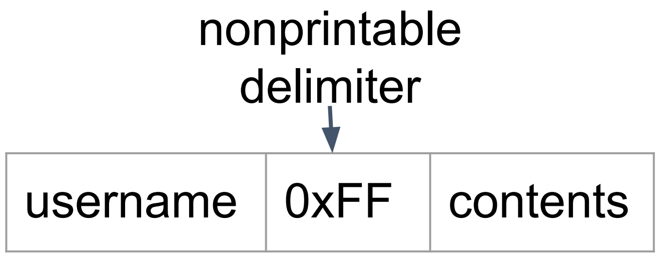
IP: 192.168.49.1
(always)

Chat messages

IP: Randomized!

Implementation Details

- Chat Message Structure
  - 2 fields: username & contents
  - Extendable to files, photos, etc.



nonprintable delimiter

username	0xFF	contents

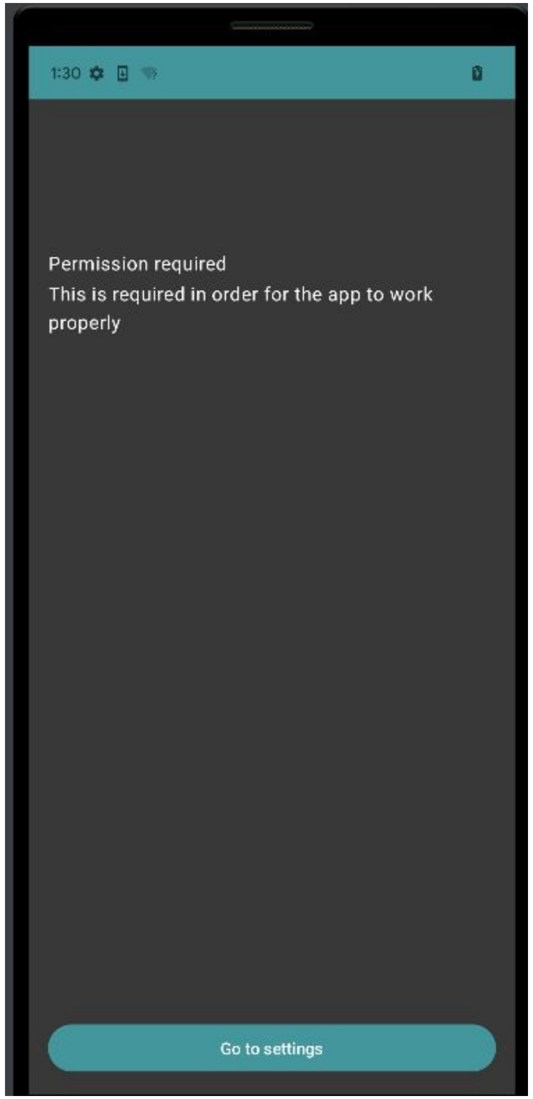
- Messages stored in local SQLite database
  - Indexed on contact_name
  - Android Room API makes this very painless

Field	Datatype
chat_id	int
contact_name	String
source_is_me	Boolean
contents	String

Layout

- Prompts user to add permissions
- Skipped when adequate permissions are given
- Prevents user from using app without perms

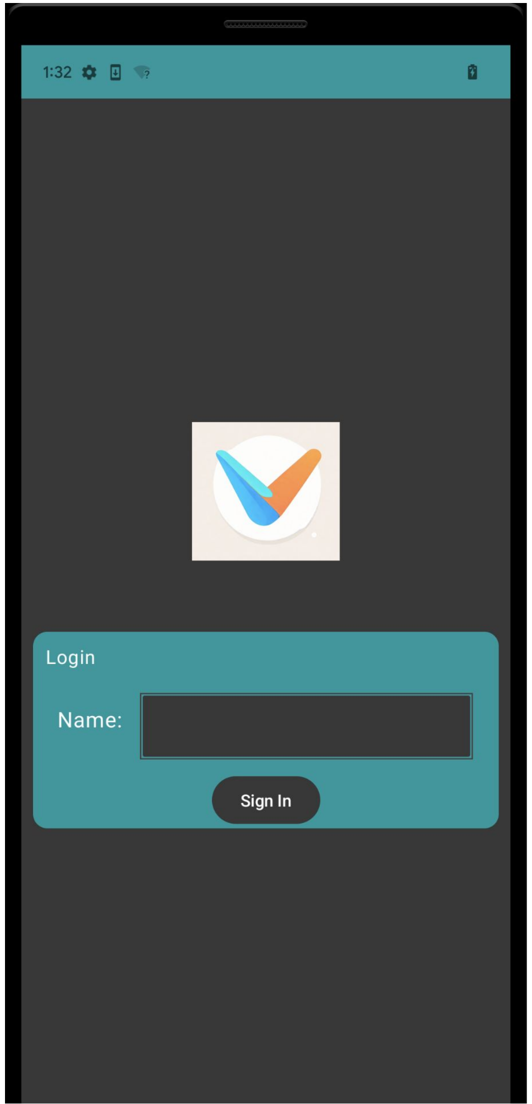
Permissions Page



Permission required

This is required in order for the app to work properly

Login Page



Login

Name:

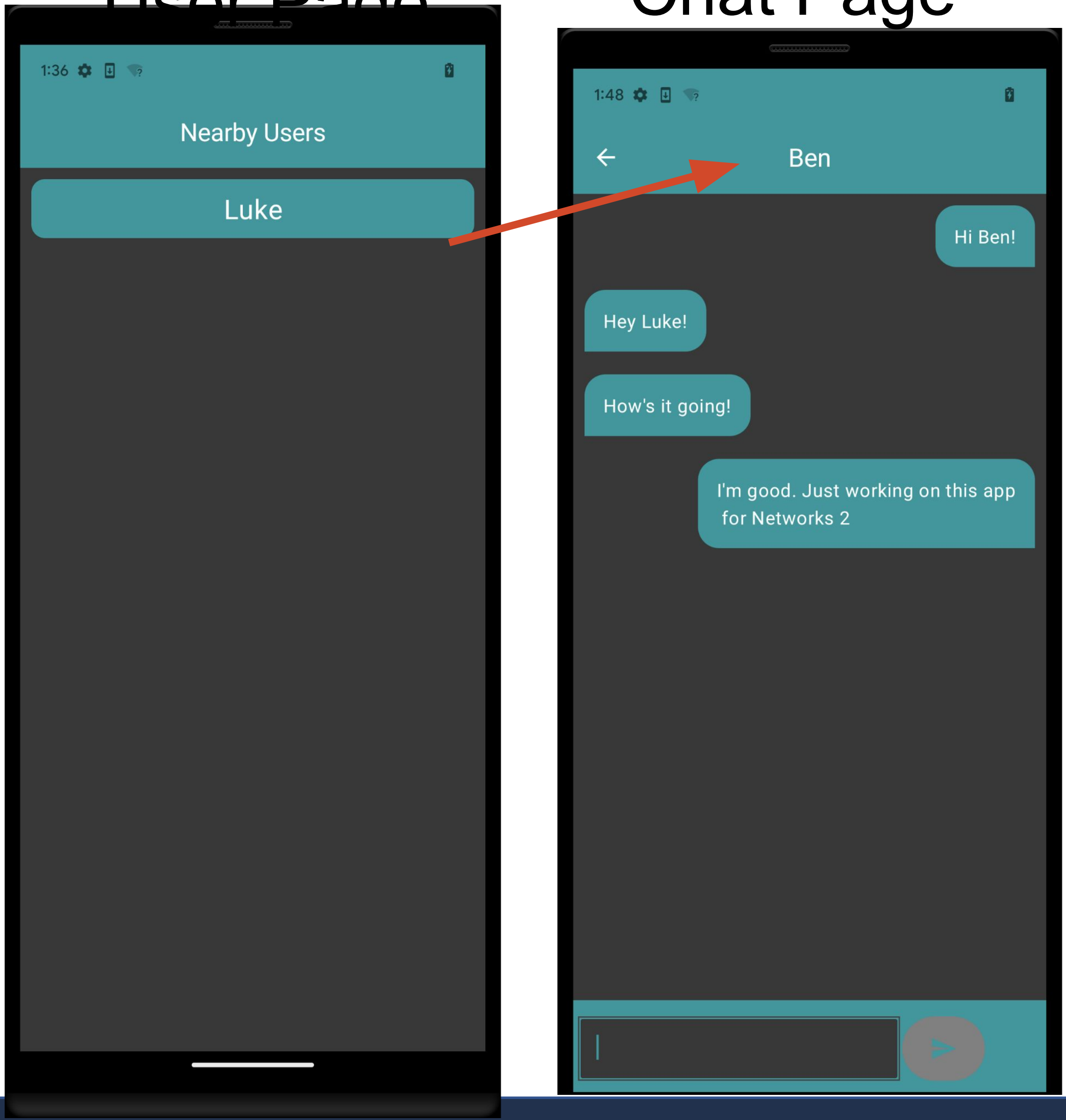
- User sets their display name
- Sign-In button triggers DNS-SD broadcast and starts WiFi-Direct server

Layout

- Shows all nearby users
- Clicking on user opens chat screen
- Users appear as they open and close app

User Page

Chat Page



1:36

Nearby Users

Luke

1:48

Ben

Hi Ben!

Hey Luke!

How's it going!

I'm good. Just working on this app for Networks 2

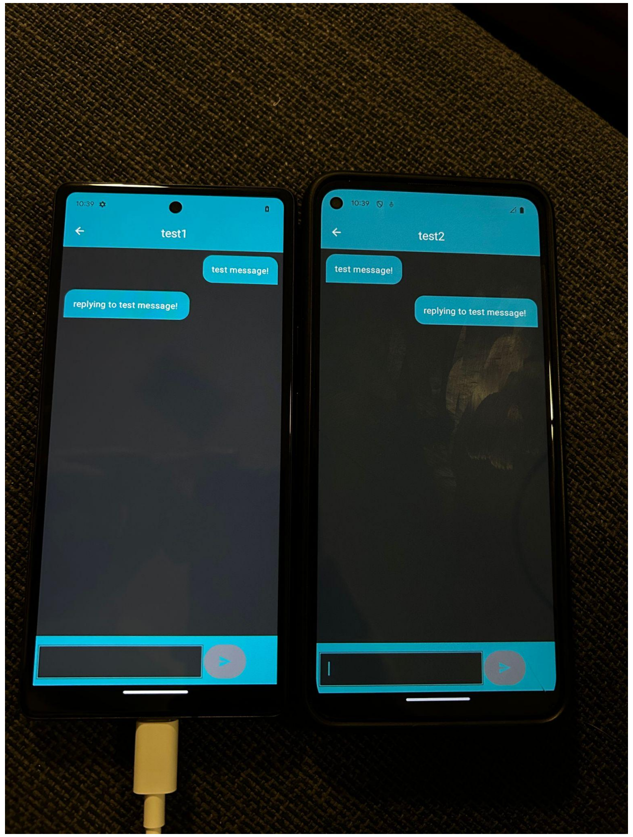
- Retrieves messages from local SQLite DB
- Prompts user to send message
- Automatically updates when new messages are sent/received

Demo

- Set up phone on screen and pass around second device

Testing + Performance

- Real implementation
  - Two Android phones
  - One device logging information
- Current results
  - Can handle high pace usage
- Compared to cellular messaging
  - 1 second in ideal conditions
  - Much longer in unideal
- Future testing
  - Throughput and packet loss
  - Distance impacts on all other factors



test message!

replying to test message!

test message!

replying to test message!

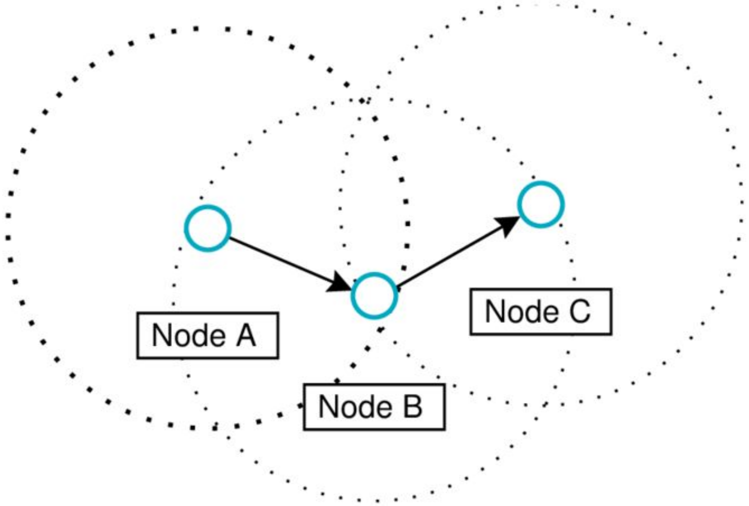
Round-Trip Time (RTT)	**0.594 seconds**
New User Discovery Latency	**6.62 seconds**
Dropping User Latency	**69.55 seconds**
Functional Range	**50 meters (at least)**

Challenges

- New to Android development and Kotlin
- We had two android devices, but one was not compatible with WiFi direct
  - Solution: Buy a cheap Android phone
- Cheap phone claims to support Wi-Fi Direct, but it actually doesn't!
  - Solution: Buy an expensive Android phone and return to Best Buy later
- Limitations of WiFi Direct

Multi-hop Messaging

- Originally a goal in our proposal
- Wifi-Direct technologies not built to support it
  - WiFi Direct is a very opinionated protocol
  - It is designed to be used for nearby, 1-to-1 peer connections
- Would require extensive modification to the protocol
  - Added overhead latency
  - Network complexity



Security + Privacy Considerations

- Security is not a primary concern of our implementation
  - However it could be easily extended
- Encryption scheme (similar to TLS/SSH encryption):
  - Secure key exchange at connection time
  - Encrypt message before transmission and decrypt upon receiving
- Identity verification scheme (CA & Certs):
  - On connection, transmit an identity, public key, and MAC address with a combined signature block from a recognized central authority

Future Work

- More experimentation

- Implementing security
- Integrate into standard messaging apps as an "emergency feature"
- Cross-platform compatibility
  - Android, macOS, Linux, Windows, etc.

Questions?