MONDAY

REU:

- Gave week 3 status presentation
    - Completed in week 2:
        - Introduction to SDN Concepts
        - Introduction to AODV application
        - Mapping requests and handling them (buggy)
        - Researching and noting the OpenFlow protocol
    - Scheduled for week 3:
        - Implement a Service in the AODV Multihop application which will communicate with the separate control application
        - Begin coding the control application
        - Modify the AODV service to:
            - Retrieve information needed by the control program
            - Distribute information (settings, etc) sent by the control program
            - Appropriately handle recieved settings requests (flow table mods)

Research:

- Generic AIDL and Service tutorials, references

Implementation:

- Created skeleton of the ControllerInterface, a Service hosted in the AODV application which will communicate with the SDN control application
- Created skeleton of the SDN Control Application
- Learned basic AIDL: Android Interface Definition Language, for inter-application communications
- Implemented a rough communication interface between AODV and the control application using AIDL
    - *.aidl file and related function calls/implementations must be kept up-to-date in both the wifi application and the control application individually
        - This was easily done in Linux using two symbolic links to the same folder -- Eclipse generally does not allow using the same source folder in two different projects. Whether this workaround will later cause problems is yet to be seen.
- Successfully started and communicated with the ControllerInterface Service in the AODV application through the SDN Control application (a basic hello method).

TUESDAY

Considerations:

- Should there be two services?
    - ControllerInterface - for speaking to the SDN Control Application
    - The actual wifi management service - for speaking to any generic data applicaiton
        - Can one service start another?

Implementation:

- Ported mapping code from the single hop wifi application to the new wifi service application (which is based on the multihop application).
    - Will require modification of multihop's ContactManager to get only adjacent nodes
    - Needs to be improved for reliability; update contacts at more appropriate times
- Added meaningful methods in AIDL:

    - void addFlowEntry(int nodeId, …..flow entry fields.....)
    - void removeFlowEntry(int nodeId, ….flow entry fields.....)
    - Map<Integer, List<Integer>> getNetworkMap()
        - not currently working, null pointer exception. May actually be an issue with passing Objects in AIDL, rather than the mapping code. StackTrace isn't very helpful.
        - If it is that latter, it may be easier to just return a specifically formatted string, or use the next method...
    - String getNetworkAdjacencyList()
    - void startWifiService(int contactId, String displayName)
        - initiates the basic wifi structure
    - void startAdHocNetwork()
        - initiates the routing protocol
- Completed porting all significant code from the wifi application's main Activity to the new Service ControllerInterface.
    - In terms of how the application functions, ControllerInterface has replaced Connect (the original main Activity)
    - The new Service can be started and bound by any application carrying the correct AIDL interface, and run persistently in the background.
    - Implemented a new Activity which will serve as the default Activity for the wifi service application (allowing the user to run the service without having a control application installed). Unlike the Connect Activity which it replaced, this Activity does not do any significant work; it only makes the appropriate connections and method calls on the Service.
        - This Activity is buggy, but not currently of major importance. Most use will be with the SDN Control Application, which starts/binds the Service more reliably, and sends SDN commands to the Service. This activity will be improved if time allows.
- Began implementation of the NetworkManagement Activity (second Activity in the SDN Control Application)
    - Queries the Wifi Service for a network map
    - Presents the map at a text field near the top of the screen
    - Lists all known nodes (not strictly adjacent) in a listview below the network map
        - Selecting a node should (in the future) present a list of SDN options:
            - View flow table
                - Will require a new AIDL method
            - Add flow entry
            - Remove flow entry

Future Plans:

- Look into how AODV handles its flow tables
    - Determine how to edit them
- Look into how the ContactManager finds distant contacts
- Stabilize the Service/interactions with it

WEDNESDAY

Implementation:

- Cleaned up and fixed a lot of unstable code related to AIDL and connection to the service, as well as checking the wifi status (this took a long time).
- Improved network mapping code, finished AIDL communications related to mapping data
- Continued development of SDN Control Program

    - Pane to show current network map (received from AdHocService via AIDL)
    - ListView beneath to show all known contacts (likewise received from a separate AIDL method)
    - Began defining Parcelable classes for storing FlowTable entries (which can be sent via AIDL between Service and Controller)
    - Defined a Parcelable Contact class.
- Renamed ControllerInterface to AdHocService to better reflect its major purpose
    - More importantly, this means that end user applications (data sharing apps) and control applications will communicate directly with the same Service.

Questions:

- How do we ultimately want end user applications to send data?
    - A high level android ContentProvider Activity?
        - Catch data sharing broadcasts, start an activity (within the AdHocService application) to handle getting the URI/sending over network.
    - Using AIDL?
        - very complicated and overkill for most applications which only need to send some data
    - Other options?

Remaining Issues:

- The wifi service crashes occasionally, usually when not in use. It is then difficult to restart
    - The issue of not being able to restart wifi after it stops has been around since before any code was modified...not sure how to approach.
    - Maybe finding a way to kill the service process (low level) will help here.

THURSDAY

Notable Readings:

- SDN and OpenFlow: A Tutorial by ip infusion
    - http://www.imsaa.org/tutorial_4.pdf
    - Discusses the Network OS and Nypervisor
    - Network OS is a distributed system, while the nypervisor sits on a single device.

Implementation:

- Discussed with Paul how to work around the fact that the ContactManager no longer contains only adjacent Nodes (which made mapping convenient in the SingleHop application)
    - At a lower level, the Node's FlowTableManager can be accessed. This contains a method to return adjacent nodes (which may work). If not, one is implementable at this level of abstraction.
    - Need to query and pass this information back to the map manager, then back to the service, then to the control program via AIDL.
- Added a new Thread to the main Activity of the SDN Control application to poll the service every half second, and display the status on the main activity's layout (should make debugging the Service and connection stability much easier).
    - Service not bound.
    - Bound, but wifi not running.
    - Bound and wifi running.
- Finalized the Parcelable Contact class
    - Contact objects (currently containing just an id and name) can be freely passed between the service and control application.
    - Modified the networkMap AIDL method to return a HashMap<Contact, List<int>> (rather than int to int). This allows the Control application to receive all relevant

node information with a single AIDL call. The network map is undirected (I am assuming, unless an error occurs), and and thus every contact id (effectively ip address) that occurs in any of the list values should exist in a Contact key.

Future Plans:

- Finish fixing network mapping tomorrow to remain on schedule
- Research network OS and nypervisor
- Look into flow table modification

Simplified summary of what is working (typical use case):

- Control application is started:
    - Status polling thread starts (in onResume()) and sets status to “Service not bound”
    - On button click, a new thread is started which binds to the WifiService in the second application.
        - Status updates to “Service bound, but no wifi”
        - The thread waits till Service is bound, then calls the AIDL command to initiateWifi. Then this thread dies.
        - Status updates to “Service bound and wifi running”
    - On a different button click, the AIDL command to start the routing protocol is called, which tells the service to call node.startNode() in AODV.
        - TO-DO: Find a way to detect this, then allow status thread to update it.
        - ContactsView Activity in the service application is called and appears
            - This is not desirable or useful behavior, but it is a remnant of the activity from which the service was made. When the useful (non-gui) code of this Activity can be ported to another class, the intent launching this Activity will be replaced with an AIDL call, if needed Probably, it will just be removed.
    - The third button clicked brings the user to the ManageNetworkActivity.
        - Here, the first button pressed updates map data. It sends the AIDL method which tells the Service to tell the MapManager to clear its stored data and broadcast a new map request.
        - Because map requests do not currently work in multihop (tomorrow’s goal now that I have Paul’s insight on using FlowTableManager), the MapManager responds with meaningless data (a randomly created HashMap of Contacts).
        - The Map is successfully passed on to the ManageNetworkActivity via AIDL, where an adjacency list (of contact ids) is displayed at the top, and all known nodes are displayed in a ListView at the bottom (contact ids and display names).

FRIDAY

Research:

- Read a few articles relating to the Nypervisor and network operating systems
    - Network OS is a distributed system, which operates in part on individual devices.
    - Each part of the network OS communicates with others, sharing state information.
    - Nypervisors act independently on individual devices; virtualize the network interface before reaching the network os.
    - Read NOX: Towards an Operating System for Networks
    - http://www.cs.yale.edu/homes/jf/nox.pdf

- Studied AODV's flow table/forwarding architecure.
    - Application initiates and starts a Node
    - Node declares a FlowTableManager
    - FlowTableManager declares multiple flow tables
    - FlowTableManager adds relevant flow table entry types
    - Hop counts are recorded in flow table entries, and used to determine forwarding schemes.

Implementation:

- Added code to the application control, map manager, and flow table manager to correctly aggregate adjacent contacts.
    - This collection does not appear accurate, and needs to be debugged, but is on the right track.
- Now able to monitor if node is active.