NFC-aided Map Routing

CS634: Mobile Computing: Project

Benjamin Schiller (benjams) - Hrishikesh Terdalkar (hrishirt) - Dibya Ranjan Sahoo (drsahoo)

Aim: To create an Android App for Map routing in IITK Campus with use of NFC tags.

Concept:

Routing in a small-scale map (IITK Campus) is performed with the help of NFC tags to tackle scenarios with bad GPS/Network reception. NFC Tags are used for storing the info about current location as well as nearest targets of interest, which when read via phone plots the points on the map as well as adjusting the start location. Routing to destination is performed using iterations of A* search algorithm.

Due to restrictions of Google Map API and unavailability of map-tile-data for most countries including ours, we decided to go with Open Street Map. (OSM Wiki: http://wiki.openstreetmap.org/wiki/Main_Page). OSM provides XML data which is parsed and interpreted in form required for A* algorithm.

The App has offline map data of the IITK campus, and capability to fetch map-tiles data through mobile data / wifi.

NFC Technology

"Near-Field Communication" (NFC) is a set of ideas and technology that enables smartphones and other devices to establish radio communication with each other by touching them together or bringing them into proximity, typically a distance of 10 cm or less.
Each full NFC device can work in 3 modes: NFC target (acting like a credential), NFC initiator (acting as a reader) and NFC (peer to peer).
We are using NFC tags - which work as "targets" which contain manually stored information. Phone (with the help of our app) works as NFC reader, which interprets and utilizes this information.
Specifically, our NFC tags contain geo-coorindates data in format
"Name_1,X_coordinate_1,Y_coordinate_1;Name_2,X_coordinate_2,Y_coordinate_2;"
Each tag is supposed to be installed in the proper location which contains a special coordinate *"S,X_coordinate,Y_coordinate;"* which holds data about the Geo-coordinates of the tag. "**S**" marker is interpreted as "Start" marker and is set as current location when read by app. Similar to "S", there's also a "**T**" (Target) string that can be interpreted as target.
These are used for the periods of disconnectivity from GPS/Mobile Network.

A* Search Algorithm:

A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way.
It uses a knowledge-plus-heuristic cost function of node *x* (usually denoted *f(x)*) to determine the order in which the search visits nodes in the tree. The cost function is a sum of two functions:

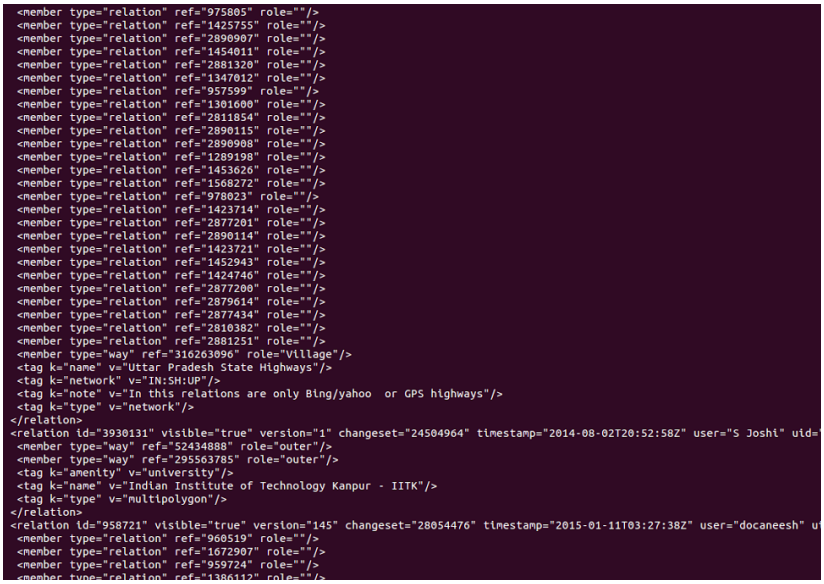
- the past path-cost function, which is the known distance from the starting node to the current node *x* (usually denoted *g(x)*)
- a future path-cost function, which is an admissible "heuristic estimate" of the distance from *x* to the goal (usually denoted *h(x)*).

The *h(x)* part of the *f(x)* function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, *h(x)* might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

If the heuristic *h* satisfies the additional condition $h(x) \leq d(x,y) + h(y)$ for every edge (*x, y*) of the graph (where *d* denotes the length of that edge), then *h* is called monotone, or consistent. In such a case, A* can be implemented more efficiently—roughly speaking, no node needs to be processed more than once (see *closed set* below)—and A* is equivalent to running Dijkstra's algorithm with the reduced cost *d'(x, y) := d(x, y) + h(y) - h(x)*.

OSM Handling

OSM gives map data in form of XML strings, in which intersections, important places etc exist as node. This needed to be parsed properly, which is done in class ParseOSMXMLFile.java



It parses the .osm Files from Openstreetmap.org and creates a graph of nodes and ways in form of a linked list. Each node is given a tag to decide whether its a part of a street and also some location information if available.

Shown besides is an example of OSM file which is handled by the above class.

How App Works:

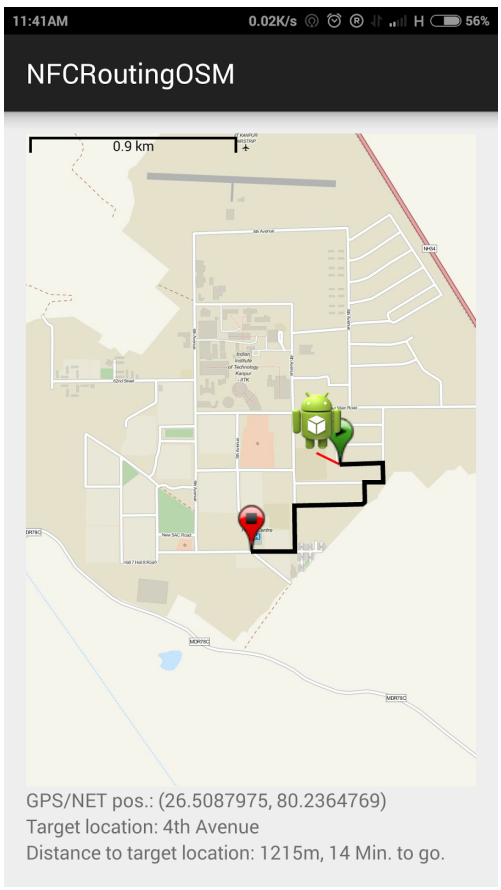
As mentioned above, first map data is fetched and parsed. A* is then applied on these nodes in the underlying graph, to compute nearest route. Once a route is calculated, it's plotted on the map and every 10 seconds user's location is re-checked to confirm that user is still on the correct route. If user deviates from the route, then from the new location, A* is run again to compute the new route.

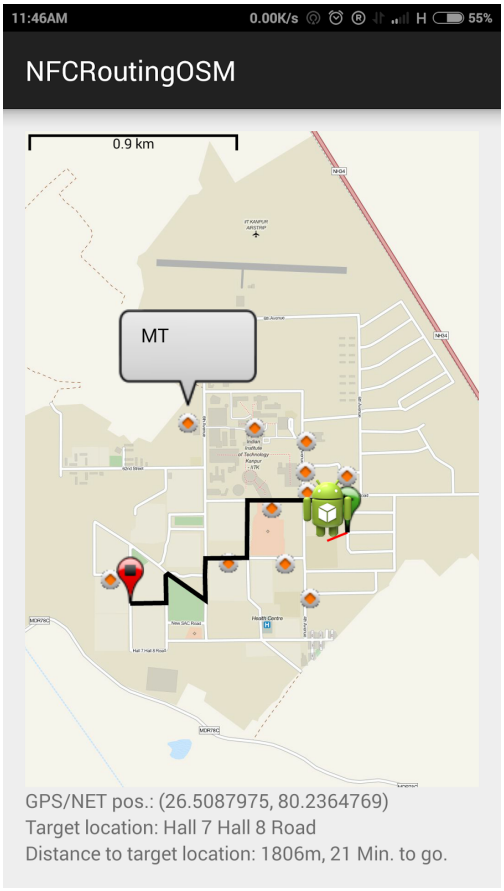
If it happens that user has lost network / GPS connection, NFC tags (which are expected to be deployed all over the campus) will have information about the current location at numerous places.

NFC tags contain small amount of information (512 Bytes), which are used to store coordinates of the place as well as coordinates of few nearby important places. Holding the phone near to an NFC tag enables phone to read the data from NFC tag, and plot the points on the map as well as updating its current location, from where again A* can run to find the route. The added **markers** are selectable and can be chosen as target locations.

App can function in completely offline mode (tested in Aeroplane mode with GPS off), with help of getting NFC data from nearest tags from time to time. We have created 4 sample tags, which contain location data about nearby places of interest. Following are some screenshots of the app. First one, with normal routing. Second one

after an NFC tag has been read and a new target location has been chosen. *(start location is also modified by the tag)*





Achieved: Android app that routes using A* algorithm..

- ❏ Map Routing both GPS based and offline (NFC based location)
- ❏ NFC Reading / Interpreting
- ❏ "Important" places nearby - stored on NFC tags
- ❏ Clickable target location markers on reading tags
- ❏ Map of IITK campus available offline
- ❏ Ability to fetch map on mobile data / wifi
- ❏ OSM Parser (Required for converting available OSM data in usable format)
- ❏ Route re-calculation on deviating from path
- ❏ Distance / Time estimates to target location

Issues Faced:

- ❏ Initial plan was to use Google Maps API and map data, but their terms of use don't allow usage of own algorithm with their map data. Also, the tiles aren't available for a lot of countries including our own. (Mostly available for African countries).
- ❏ Getting the .osm file, parsing it into a self made osm-parser was some time consuming part.
- ❏ General Android-debugging difficulties
    - ❏ e.g. Markers don't have a suitable way to react on user interaction, so had to extend the class and write own method.

Content:

1] Entire Source Code

```
./NFCRoutingOSM_v1.0
    ./NFCRoutingOSM_v1.0/appcompat_v7
    ./NFCRoutingOSM_v1.0/NFCRoutingOSM
```

2] OSM Map Data for IITK Campus

```
./NFCRoutingOSM_v1.0/NFCRoutingOSM/assets/campusFull.osm
```

3] Compiled Android App

```
./NFCRoutingOSM.apk
```

3] App Icon

```
./NFC_Icon.png
```

4] NFC Tags Content

```
./NFC_Tag_Content.txt
```

5] README.txt

```
./README.txt
```

6] Project Report

```
./Report.pdf
```

7] Screenshots

```
./screenshots/
   ./screenshots/nfc_based_routing.png
   ./screenshots/app_on_homescreen.png
   ./screenshots/routing.png
```

8] Map Tiles Data (.osm and .jpg)

```
./NFCRoutingOSM_v1.0/NFCRoutingOSM/assets/
```

Future Improvements:

- ➢ OSM data about the campus does not contain too many nodes.
- ➢ Currently NFC tags contain coordinate data about nearby locations. The tags can contain only 500 bytes information. (500 characters). Stored data on NFC tags can be made into compressed format to support storing more information.
- ➢ Multiple suggested paths at same time.

References:

1. NFC: http://en.wikipedia.org/wiki/Near_field_communication
2. A* Algorithm: http://en.wikipedia.org/wiki/A*_search_algorithm
3. OSM Wiki: https://wiki.openstreetmap.org/wiki/Main_Page
4. Android NFC Reference: https://developer.android.com/guide/topics/connectivity/nfc/index.html
5. Multiple stackoverflow references