Dynamic Path-Spectrum Route Identification using Crowd-sensed Smartphone Data

Introduction:

With increased adoption of handheld wireless devices combined with spectrum scarcity, it is beyond doubt that future wireless networks must effectively improve spectrum utilization. TV White Spectrum (TVWS) approach has gained a lot of popularity in the recent times as a promising approach in solving the spectrum scarcity problem. For TVWS, standards bodies like the FCC strongly recommend the querying of geo-location databases to obtain spectrum availability information. LTE is now widely deployed and it outperforms Wi-Fi 40% of the time, and 75% of the time, the difference between LTE and Wi-Fi throughput is higher than 1 Mbits/s [27].

In that context, for a mobile smartphone user, when moving from one location to another farther location, it is advantageous to plan ahead of time, a physical route utilizing the Crowd-sensed Smartphone Data, that best trades-off travel time for improved signal quality. Also, this approach naturally accommodates the mobile user select and switch intelligently between the best available networks: Wi-Fi, LTE. On the other hand, we will avoid a congested road traffic route despite its high spectrum availability. We will term such routes as **path-spectrum routes**.

Since this problem is quite generic in terms of physical routes to start with, we constrain the problem to that of bus routes. Bus routes are pre-determined and this makes suggesting path corrections/route deviations (switch from one bus route to another) precise and relevant in time. In contrast, driving routes (say, by car) are not pre-determined, implying a potentially large set of candidate routes to choose from, and so, unless such a set is somehow further constrained, the relevance of suggesting a route deviation quite ahead of time is not well defined. However, solving this problem for Metro Boston MBTA bus connections will establish the proof-of-concept.

It's likely the radio environment changes enroute, and so, we should allow for dynamic path correction. The number of path corrections can be limited based on user preferences.

At this point, we can identify some potential applications for path-spectrum routes:
1. A patient requiring an emergency transport to a nearby hospital. When seamless data connectivity is guaranteed in the ambulance, the doctors can attend remotely to the patient in getting immediate medical attention which can be critical.

2. Mobile users (1 Tx and *N* Downloaders) in close proximity may form a mobile P2P network to offload partial chunks of downloaded data among themselves effectively bypassing the cellular network.

Problem statement:

1. How good or bad are the routes, in terms of cellular/Open Hot-Spot Wi-Fi signal quality, recommended by Google Maps? How can we identify path-spectrum routes that enjoy good signal quality and satisfies the mobile user's data connection requirements?

In Figure 1, the heat map is indicative of the network coverage from AT&T. Clearly, apart from Downtown area, there are a lot of locations that do not have any coverage. Since Google Maps does not optimize for network coverage, a preferred route is one that lets the mobile user move along the hotter regions of the heat map locations. We can see that there will be a trade-off when trying to achieve a minimum commute time and good signal quality.

Further, there is also the user's preference in choosing a route over another for matters of convenience. Let's say there are two routes R1 and R2 for a mobile user to get from location A to location B. Even though the distance associated with R1 might be more than distance associated with R2, the commute time along route R2 can be shorter compared to that of R1 possibly because of traffic conditions, road conditions, frequent toll booth stops. And, the user might still prefer the longer or slower route for it might be a convenient route. i.e. fewer number of bus transfers or less walking in a route. For example, a user might prefer to switch buses to get to their home rather than commute by train for the latter involves more than 15 mins of walking (refer to Figure 2).

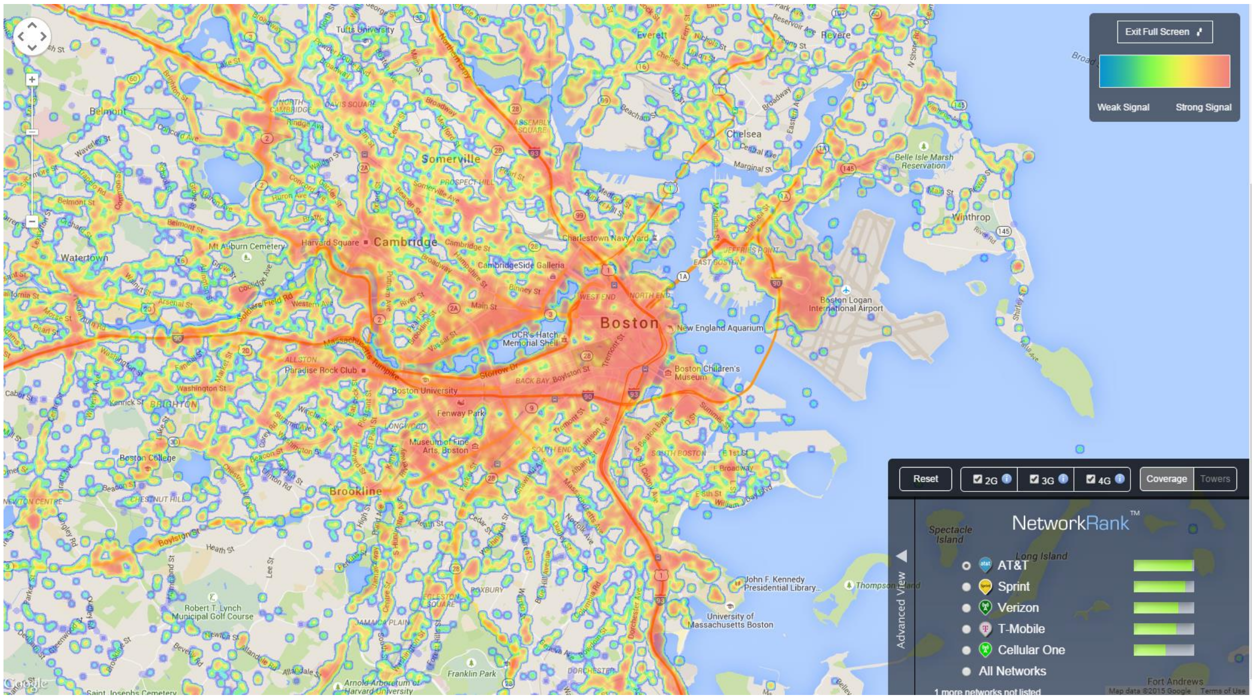

Figure 1: 2G/3G/4G coverage of AT&T obtained from OpenSignal

We observe the coverage is more or less patchy like in Figure 1 with other Cellular carriers too.

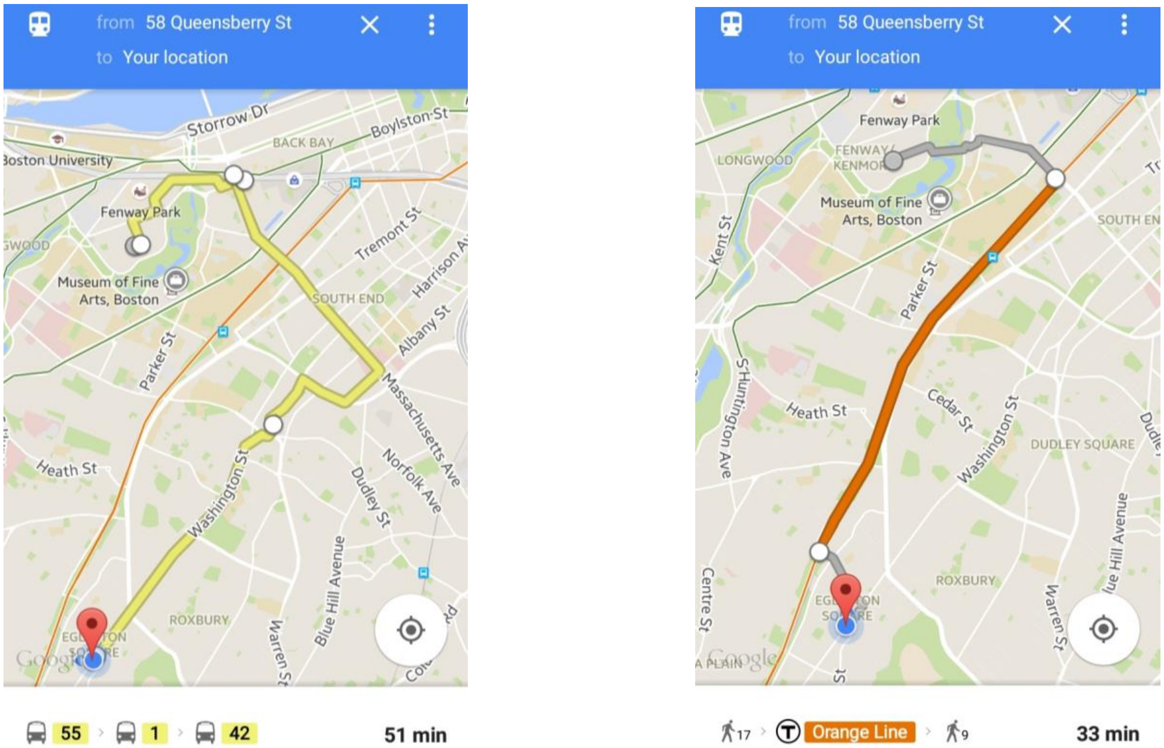

Figure 2: User choosing a slower but convenient route

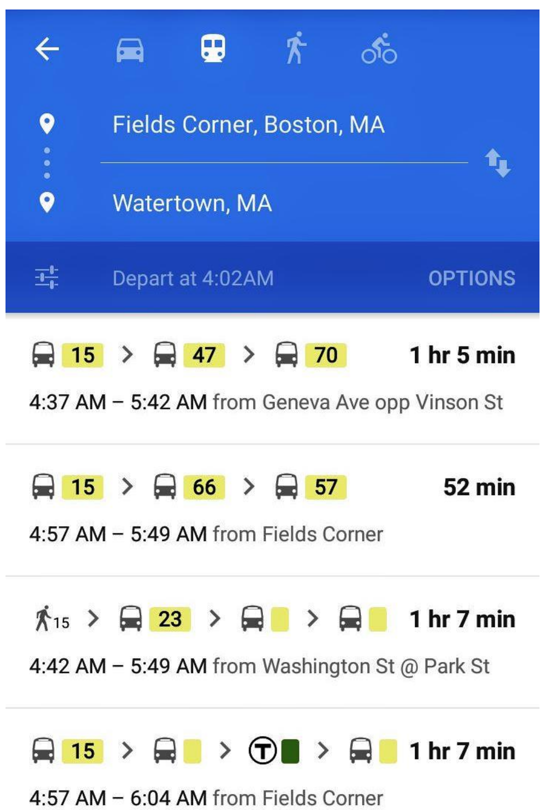

Figure 3: Multiple bus routes options for a pair of locations

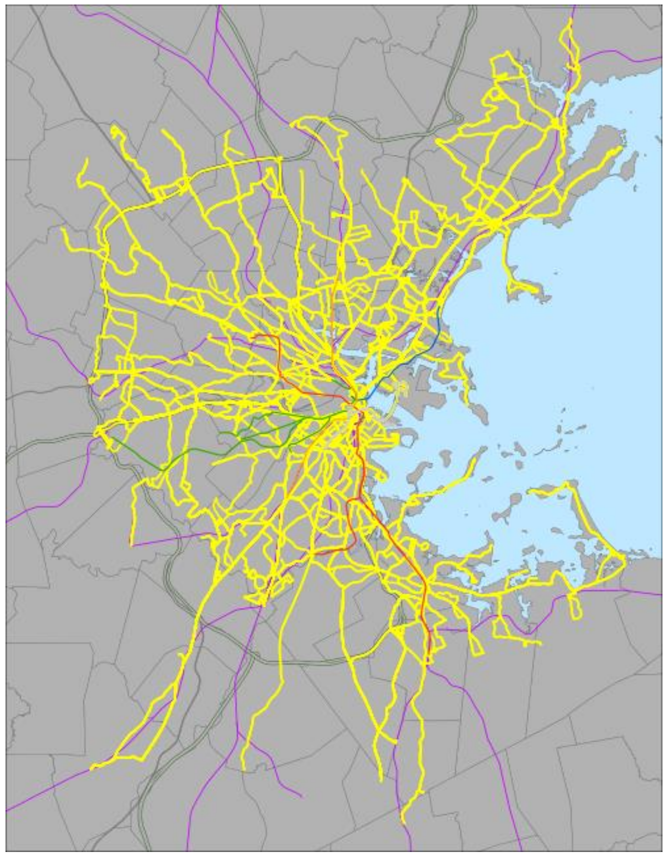

Figure 3: Geographic map of MBTA Bus service [3]

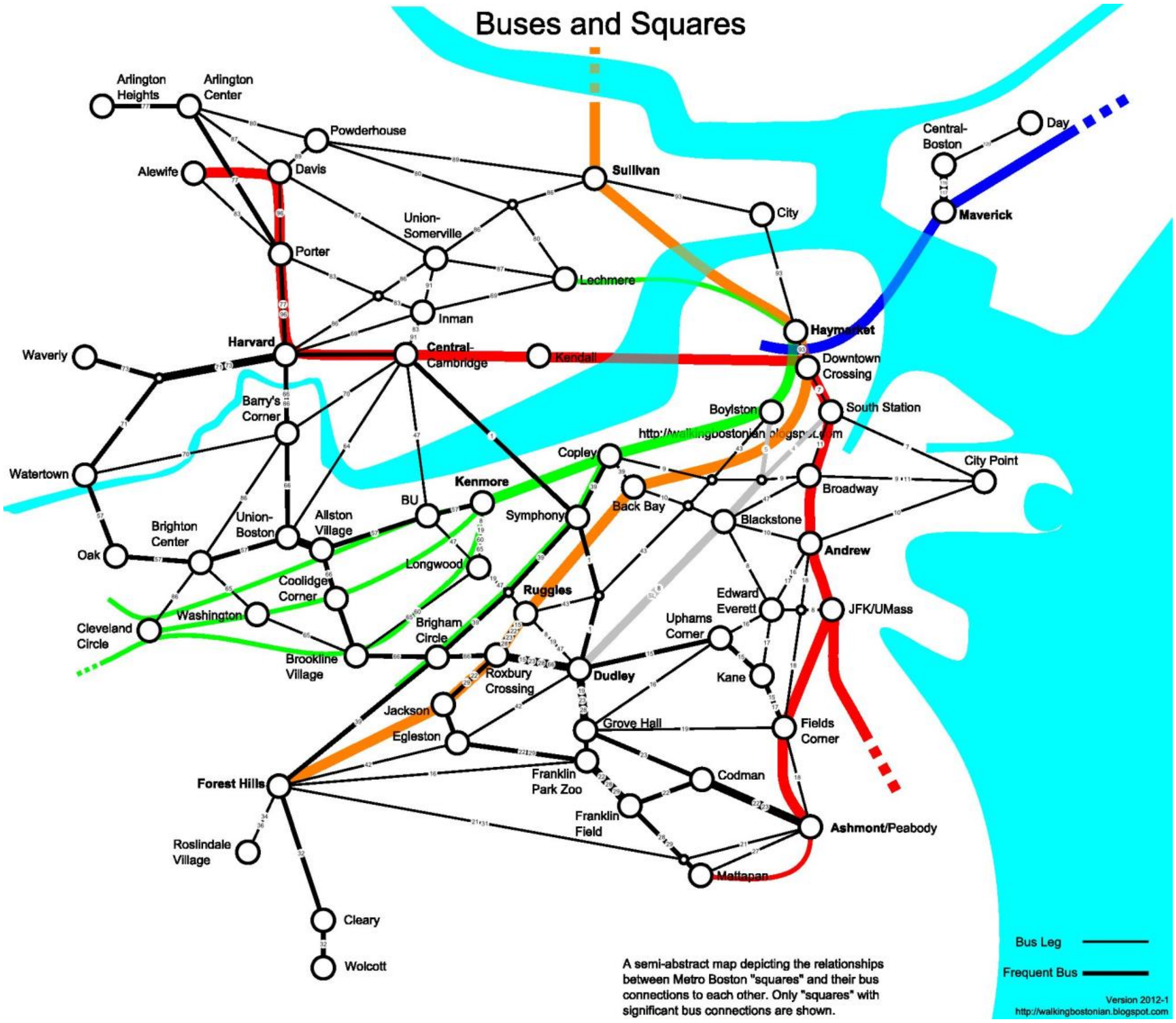

Figure 4: Semi-Abstract map of Metro Boston Bus Connections [4]

In Figure 4, the bus stops with significant or multiple intersecting bus routes can be thought of “squares”. The semi-abstract map will serve as a template for the weighted graph we will work with to identify path-spectrum MBTA bus routes when input with Start and End commute locations.

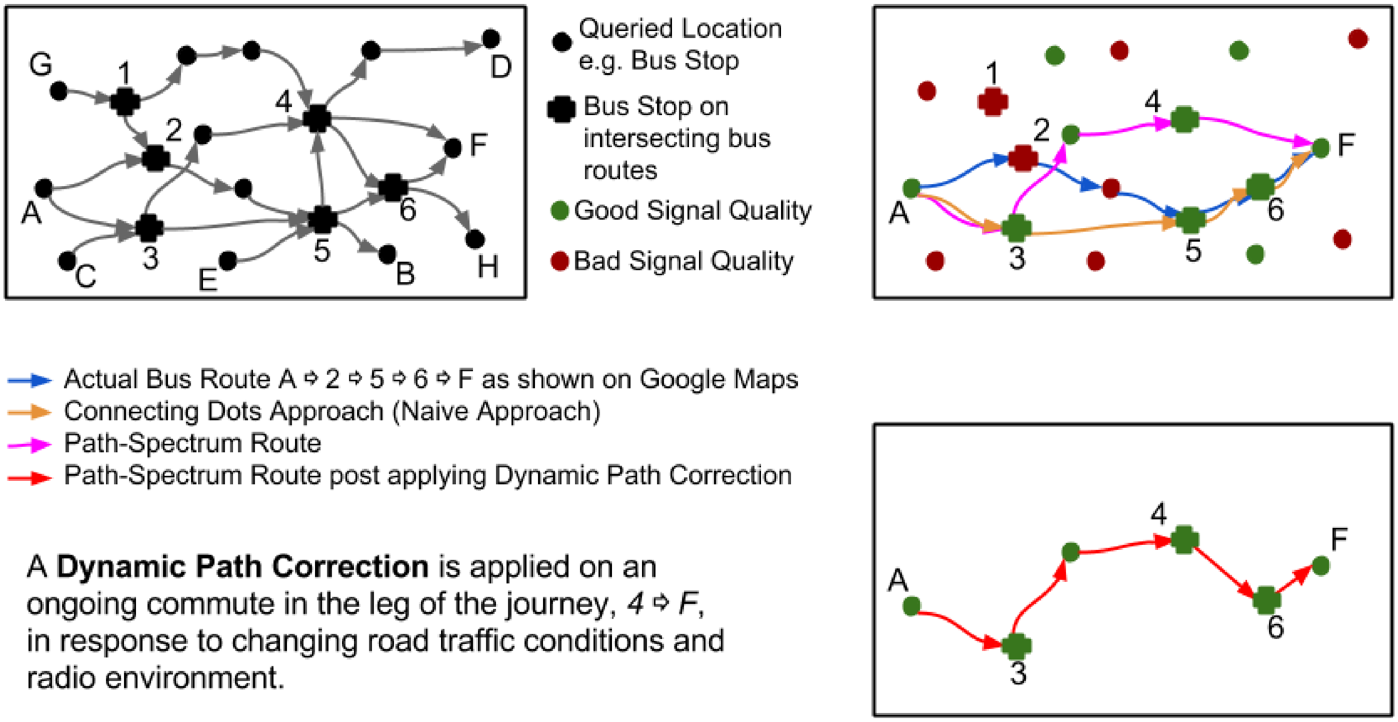

Figure 5: Illustration of our approach for identifying path-spectrum bus routes

In Figure 5, let P: = {A→B, A→F, C→D, C→H, E→H, E→D, G→B, G→F} be the set of pre-determined inbound direct bus routes. The directed path segments indicate that the buses are traveling inbound. Let's consider the commute from A to F. The bus route A→F is shown in blue is the shortest time route from A to F possibly taking into consideration the commute options preferred by the mobile user. However, the mobile user commuting along the blue route will suffer poor connectivity as the route includes two bus stops neighborhoods having bad signal quality (shown in dark red).

The naive approach is to commute along the connected-dots-route (shown in dark yellow) to reach the destination. Although the naive approach accounts for excellent signal quality, it completely ignores the road traffic conditions. The suggested route might possibly suffer from heavy road traffic congestion and as a result might take way too long to reach the destination. Further, it does not account for the user's demands on the data connection. The path-spectrum route (shown in bright pink) on the other hand, best trades-off travel time for good signal quality. Also, we might apply a path correction enroute (shown in bright red) to the path-spectrum route to factor in the temporal changes in traffic conditions and the radio environment.

The above mentioned trade-off can be adequately quantified by designing a cost metric i.e. a weight associated with a path segment. The weight can be thought of as the cost incurred by the mobile user in traversing that path segment.

We will now formally propose an **algorithmic framework** for the above problem:

Definition:

• A weighted directed acyclic graph (DAG) *G* is a pair (V, E), where V is a finite set of points called vertices and E is a finite set of edges. • The edge e_{ij} is an ordered pair (v_i, v_j) that has an associated weight $w^t_{i\to j}$ or simply w^t_{ij}. An edge (v_i, v_j) is incident from vertex v_i and is incident to vertex v_j.

• A path from a vertex v_o to a vertex v_E is a well-defined sequence of vertices. Any ordered pair of vertices in that sequence, $(v_l, v_m) \in E$.
• The current time step $t \in \{1, 2, ..., T\}$.

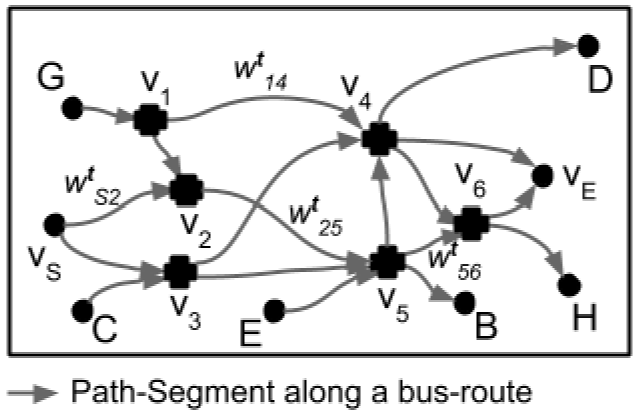

Figure 6: The dynamically weighted DAG *G* with some of the weights shown

Note that a path-segment can possibly include one or more bus stops or locations that are queried. The weight associated with a path-segment is the cost incurred by the mobile user in traversing the same. Here, the start and end vertices are represented $v_S := A$ & $v_E := F$ respectively.

We design the w_{ij}^{t} as a function of the estimated commute time τ_{ij}^{t} and data connection reliability $\gamma_{ij}^{t} \leq 1$ at the current time step t. Estimated commute time takes into account the current road traffic congestion, on-the-road-distance from start location and stop location, bus wait time, bus switching time. c_1 is the constant of proportionality in the model.

$$w_{ij}^{t} \alpha c_1 (1 - \gamma_{ij}^{t}) \times \tau_{ij}^{t}$$

Essentially, we want to determine the 'shortest' path between a source vertex and the destination vertex in the weighted DAG *G*.

Path-Spectrum Route from v_S to v_E:

The directed path from v_S to v_E such that $\sum_{candidate\ paths} w_{ij}$ is minimum.

Path-Spectrum Route with Route Deviations from v_S to v_E:

Note: Vertex v_t represents the current location of the mobile user & $v_0 = v_S$

The directed path from v_t to v_E such that $\sum_{candidate\ paths} w_{ij}^{t}$ is minimum.

We can naturally employ the popular **Dijkstra's algorithm** in solving the above stated problem. Gains in running time of the algorithm can be obtained by using topological sorting on *G* (that represents a linear ordering in *G)*. On the other hand, a **modified Genetic Algorithm (GA)** can be used to solve the above problem for the case of driving routes. GAs do a good job of efficiently searching a large state-space of solutions, and converge on one or more 'good' solutions, but not necessarily the 'best' solution. This algorithmic approach is appropriate for the driving routes as they are not pre-determined and also are largely unconstrained.

More sophisticated cost metrics can be designed for the following route identification scenarios:

1. High bandwidth or a threshold bandwidth requirement.
2. Bounded download limit requirement. Can operate like Rx -> PAUSE -> Rx -> PAUSE.

We expect that the choice of different cost metrics will translate to using variants of the shortest path weighted DAG algorithms.

When we extend this setting to Mobile P2P network, with 1 Tx and N downloaders, as mentioned earlier, we can then think of solving the problem of what partial downloaded chunks to offload with a constraint of not more than D route deviations (unlike path corrections). We might also try to optimize for energy, latency and speed in this setting.

Path-Spectrum Route Identification Algorithm
Inputs: Time of Query, Start and Destination Locations (in English), Data connection mode
Step 1: Determine from input pair of locations the (inter)cardinal direction of travel and construct a simplified directed Boston MBTA Bus Metro Graph *G*.
Step 2: Do a real-time MBTA bus API query to estimate arrival time of the first leg connecting bus for select bus routes in *G*. Drop the routes that require long waiting times in *G* and retain only the quickest candidate routes, say top 15, i.e. ranked based on average commute time. Call this the feasible set of routes F.
Step 3: Do a new OpenSignal API query and lookup earlier API query responses at intermediate locations along the candidate routes in F, on the road, possibly at bus stops.
Step 4: Re-rank the candidate routes in F based on data connection reliability. Trim F by retaining only say, the best 10 routes. Call this the path-spectrum route set P.
Step 5: Render on a street map the top 4 path-spectrum routes with the best route highlighted in a different color.

References

1. **OpenSignal** is a comprehensive database of cell phone towers, cell phone signal strength readings, and Wi-Fi access points around the world; URL: http://opensignal.com/
2. T-Mobile Speed Measurement Tools; URL: t-mo.co/spdapps
3. List of MBTA Bus Routes
4. Square Hopping by Bus: blog entry from The Walking Bostonian
5. MBTA Bus Schedules and Maps; URL: http://www.mbta.com/schedules_and_maps/bus/
6. Live MBTA Subway Mapping; URL: http://sites.harvard.edu/~wuensch/T/subway-map.html
7. Code that creates the Real-time MBTA bus location + Google Maps mashup
8. MBTA data and web services. Related links on the right.
9. MBTA Rider Tools
10. MassGIS Data - MBTA Bus Routes and Stops
11. Traffic Alerts - public data unlike WAZE: Best Route taking into Traffic conditions, Traffic Alerts
12. R Package 'geosphere: Spherical Trigonometry'; http://cran.r-project.org/web/packages/geosphere/index.html
13. Andy Monat's MBTAinfo; URL: http://www.mbtainfo.com/
14. CellSight; URL: http://www.cellsight.com/web/index.html
15. NYC Hot-Spot Locations; URL: https://nycopendata.socrata.com/widgets/a9we-mtpn
16. Can You Track Me Now? (Visualizing Xfinity Wi-Fi Hotspot Coverage) [Part 1] [Part 2]
17. Google Maps API; URL: https://developers.google.com/maps/
18. LTE Cell Scanner/Tracker; URL: https://github.com/Evrytania/LTE-Cell-Scanner
19. Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li, "LTE Radio Analytics Made Easy and Accessible", ACM SIGCOMM 2014,Chicago IL; URL: http://www.mit.edu/~swarun/papers/lteye-sigcomm2014.pdf
20. Guoru Ding et. al., "Cellular-Base-Station Assisted Device-to-Device Communications in TV White Space"; URL: http://arxiv.org/pdf/1506.01394.pdf
21. H. Y. Hsieh and R. Sivakumar "On Using Peer-to-Peer Communication in Cellular Wireless Data Networks", IEEE Trans. Mobile Comp., vol. 3, no. 1, pp.57 -72 2004.

22. Y. Raivio, "Mobile peer-to-peer in cellular networks," in HUTT-110.51 Seminar on Internetworking, Helsinki Institute of Technology, 2005-04-26/27.
23. Kato T., Ishikawa N., Sumino H., Hjelm J., Yu Y. and Murakami S., “A Platform and Applications for Mobile Peer-to-Peer Communications”; URL: http://www2.research.att.com/~rjana/Takeshi_Kato.pdf
24. Alasmary, W.; Valaee, S., "Sensing in Mobile Sensor Networks with Noisy Mobility Knowledge," *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th* , vol., no., pp.1,5, 14-17 Sept. 2014; URL: http://dx.doi.org/10.1109/VTCFall.2014.6966175
25. E. Troja and S. Bakiras, “Efficient location privacy for moving clients in database-driven dynamic spectrum access,” in Proc. IEEE International Conference on Computer Communications and Networks (ICCCN), August 2015; URL: http://jjcweb.jjay.cuny.edu/sbakiras/papers/icccn15.pdf
26. Schmidt, Jeffery C., Robin U. Roberts, and Peter Stanforth. "Systems and methods for determining and specifying spectrum availability for a predetermined travel route." U.S. Patent Application 14/514,672; URL: https://www.google.com/patents/US20150105116
27. Shuo Deng, Ravi Netravali, Anirudh Sivaraman, Hari Balakrishnan, “WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance”, IMC’14, November 5–7, 2014, Vancouver, BC, Canada; URL: http://people.csail.mit.edu/shuodeng/papers/deng-imc2014.pdf

Android Apps:

1. OpenSignal
2. RF Signal Tracker (Developer's Website)
3. Advanced Signal Status (Source Code)
4. Cell Map
5. MobiPerf
6. Network Signal Strength