MHCAB MEDICAL HEALTH CAB

Aseem Garg, Ankit Singh, Manthan Bhardwaj

[1] *Under Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India*
[2]*Under Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India*
[3] *Under Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India*
[4] *Under Graduate Student, Department of Computer Science and Engineering, SRM University, Chennai, India*

ABSTRACT

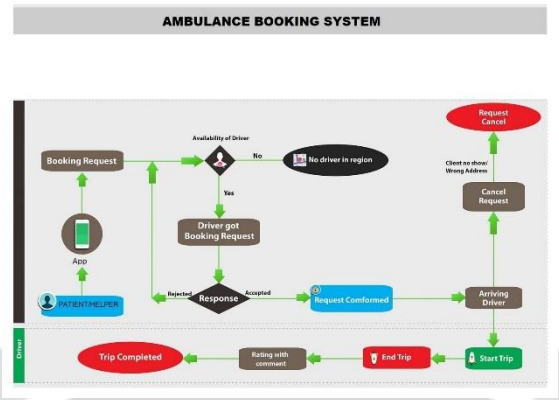
MHCAB is a proof-of-concept ubiquitous computing. An application that allows people to book nearby Ambulances using their cell phones or PDAs equipped with short-range wireless network interfaces. MHCab discovers and books ambulances using mobile ad hoc networks of vehicles. We have implemented an MHCab prototype on top of matching algorithm, a middleware architecture based on execution migration, which we had developed to provide a common execution environment for outdoor ubiquitous computing applications. The experimental and simulation results have demonstrated the feasibility of MHCab.

Keyword : - *Cab booking system, Booking system.*

1 Introduction

The idea behind the app is that various daily number of emergency cases comes where a patient has to rush to the nearest hospital, and in that case of emergency, it is very tough to find the nearest hospital and demand for the ambulance if available as soon as possible. Even after a person can contact with the hospital, the problem arises in the amount of time it takes to reach to the patient's location and to take him to the hospital. This project will try to eradicate this problem by providing an application where you can find the nearest ambulance equipped with all the emergency services. By using this application, the person will be able to find an Ambulance closest to the patient. The best part is that it will be as easy as booking a cab. Further, this app will maintain a patient's medical history Which will be transferred to the hospital where the Patient is going to be admitted, and this will be of a great help as it will reduce the time that is taken for doing basic tests. This will reduce the time as well as the expenditure. People can select the desired ambulance; the patient will Be given the price estimate range according to the distance to the hospital, which will not be changed irrespective of the actual time taken for reaching the destination. After successfully booking the ambulance user will be given the details of the driver and the Ambulance arriving to pick him up. There will be a complete segment in the app to deal with patient's issues, patients care support, return policies and all other backend features. In the admin panel, patients and driver current location tracking, patient, and driver approval and removal, patient care backend support. The MHCab dispatching system, on the other hand, is simpler, faster, and more scalable since it works in a completely decentralized fashion, and there is no need to gather the locations of all the ambulance in real-time. MHCab provides all these benefits because it defines a system architecture in which the clients and vehicles communicate using only short-range wireless network interfaces. This design decision, however, makes MHCab a "best effort" service. Clients can switch to the standard centralized methods to book an ambulance if they fail to get an ambulance using MHCab within a short period. Hence, the MHCab system can be incrementally deployed and coexist with current centralized systems. MHCab is useful in cities with high traffic cities, such as Delhi or Mumbai, where the contention to get ambulance during certain periods (e.g., people getting out from a show) is very annoying.

2 Design



MHCab consists of a mobile ad hoc network of computers embedded in ambulances and client handheld devices, which communicate using short-range wireless network interfaces such as IEEE 802.11. Instead of booking ambulances through a centralized dispatcher, MHCab clients book near by ambulances communicating directly with other MHCab nodes (i.e., taxis) over a mobile ad hoc network. This decentralized architecture provides a simple, cheap, and scalable solution to a real-life problem. However, MHCab presents new challenges which do not exist in traditional systems based on centralized dispatching centers. For instance, we need a distributed protocol to ensure that at most one ambulance arrives at the site of the client who requested the ambulance. Furthermore, we must ensure that any free ambulance accepts only one client request at any point in time. To provide an automatic booking mechanism, we also need accurate location information for both clients and ambulances (e.g., the client's street address has to be present in the booking request). Finally, MHCab needs to provide a mechanism for the client and driver to authenticate each other when they meet. This section presents the MHCab system architecture and its protocol that satisfies these requirements.

2.1 System Architecture

Mhcab comprises of two sorts of elements: customer stations and driver stations. A customer station is a PDA (or cellphone), while a driver station is a framework installed in the rescue vehicle. We expect that All driver stations speak with each other utilizing IEEE 802.11, the accepted standard for high data transfer capacity, short-go remote systems administration. Furthermore, every driver station has a GPS collector that can report its flow area when required. All driver stations that are inside the radio scope of each other shape a specially appointed system of hubs which convey and play out the disseminated calculation important to book a free rescue vehicle. The customer stations need to join such a system to infuse their solicitations for the free emergency vehicle. A customer station discusses straightforwardly with ambulances in its transmission run, and the ambulances in the system forward the demand until the point when a free emergency vehicle is found. Dissimilar to driver stations which are homogeneous, customer stations can have distinctive abilities. Other than capable PDAs furnished with IEEE 802.11 remote system interfaces and GPS beneficiaries, phones prepared without GPS recipients can likewise be utilized as customer stations. In such a case, In any case, the customer station needs to associate with a passage station that advances the demand to the system of ambulances and sends the appropriate response back to the customer. A passage station is a PC furnished with a GPS beneficiary and numerous system interfaces (i.e., Bluetooth and IEEE 802.11), which can be co-situated with Hotspots (i.e., IEEE 802.11 access focuses) at open places, for example, eateries, stores, theaters, transport stops, pay phones, and building anterooms. The part of a portal station is to send area data on demands got from lighter customers that can't consolidate a GPS beneficiary (i.e., given the scope of Bluetooth, the area of the passage a decent estimation for the area of the customer) and play out a difference in conventions between customer stations outfitted with Bluetooth and driver stations furnished with IEEE 802.11.

3 Prototype Implementation

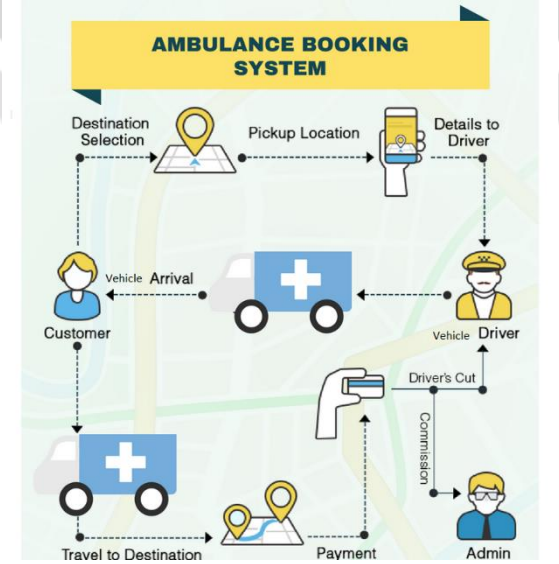
We have implemented mhCab using Smart Messages [13], a middleware architecture based on execution migration, which we developed to provide a common execution environment for outdoor ubiquitous computing applications. This section presents a short overview of Smart Messages, the mhCab prototype, the routing algorithms used by mhCab to find the free ambulance, and experimental results over ad hoc networks of PDAs.

3.1 MHCab Prototype

We have actualized the mhCab model to finish everything off the SM stage introduced on android running Linux. The iPAQs utilize Orinoco's 802.11 cards for remotecorrespondence, and each of them is associated with a Geko201 GPS collector. We have shown in an alternate venture [18] that autos can be mapped with high exactness On the streets notwithstanding the low precision gave by crude GPS information (e.g., crude GPS information gives all things considered an exactness of 10 meters MHCab model has two sorts of graphical UIs, comparing to the customer station and driver station, individually. Every UI speaks with the SM stage using extraordinary bi-directional I/O labels, called UI labels. These labels hold on for the whole term of the UI procedure and carry on likewise to a maker shopper roundabout cushion. We have utilized the Open Palmtop Coordinated Condition (OPIE) to build up the UIs of MHCab. OPIE is an open source graphical client condition for PDA's and different gadgets running Linux.

3.2 MHCab Routing Algorithms

This directing calculation assembles steering tables on-interest for hubs lying in a TTL-bounces run from the customer. The directing table at a hub can be shared among various BookSMs, and it comprises of sections with the likelihood of finding a free vehicle through the hub's neighbors. Since the course to each neighbor could be negated because of versatility, each steering passage records its last refresh time; the framework utilizes this data to age directing sections. The Probabilistic On-Request component is like the receptive directing conventions in MANET, for example, AODV [31]. BookSM triggers the Probabilistic On-Request course disclosure on any hub where the directing table does not exist, the greatest likelihood is too little, or the correspondence fizzled in light of the fact that the hub with most extreme likelihood isn't a neighbor of the present hub any longer. To do as such, BookSM makes a DiscoverySM, which relocates in the system to search for a free vehicle and pieces on a steering tag for a timeout esteem.



We have received a timeout conspire as opposed to sitting tight for a specific number of DiscoverySMs to return (i.e., we could have sat tight for one DiscoverySM for each immediate neighbor) since it is irrational to do as such in an unpredictable specially appointed system. In the event that another BookSM ask for lands at a similar hub, it will hinder on a similar tag as opposed to making another DiscoverySM. In this manner, just the main demand refreshes the directing table at that hub. The second and consequent solicitations will utilize the steering table made by the primary demand when they proceed after timeout. Figure 4 demonstrates the Probabilistic On-Request course disclosure in real life. The DiscoverySM brings forth itself at every hub. The youngster SM communicates itself to its neighbors, and the parent hinders with a timeout sitting tight for every one of its kids to return with directing data (i.e., the likelihood to locate a free vehicle through this hub). Every kid stores the address of its source hub (which as per the calculation is one-jump away). Once a DiscoverySM achieves a hub found TTL-jumps far from the customer, it will report the likelihood to its source as either 0.5 or 0.0, contingent upon whether the vehicle is free or possessed, separately. Note that each hub navigated by a DiscoverySM, with the exception of the last bounce, has a blocked DiscoverySM. At the point when the DiscoverySM unblocks (after timeout), it will accumulate every one of the probabilities announced by its tyke SMs and report back to its source hub. This procedure proceeds until the best level DiscoverySM is come to.

The courses made by this calculation shape a Coordinated Non-cyclic Chart, established at the customer station. The DAGs made by various customer stations could cover each other. The likelihood, Pn ($1 \leq n < TTL$), announced by every hub found n trusts far from the customer to its source is given by:

Pn =1/21 + max(Pn+1) if n is involved,

½ max(Pn+1) if n is accessible.

The timeout for the parent DiscoverySM at every hub is

characterized by:

2 × oneHopT time × (TTL − currentHops) × L,

Where oneHopTime is the ideal opportunity for single bounce relocation. Accordingly, (2 × oneHopTime × (TTL − currentHops)) is the aggregate relocation time. To represent diverse variables influencing the movement time, (for example, Macintosh layer conflict), we presented (L > 1) as a fudge factor. L is our endeavor to guarantee that the youngster Disclosure SMS could get back before the parent SM times out. As indicated by our trial comes about, the more conflicts, the bigger L ought to be. For example, a fudge factor L = 3 functions admirably for vehicles with 6 or less neighbors. At the point when a customer conveys a demand, BookSM just checks the nearby directing table and moves to the neighbor with the biggest likelihood. In the meantime, it refreshes the steering table in view of the TTL utilized by the DiscoverySM. When it picks the following bounce, BookSM isolates the relating likelihood by 2^(N−hopCount), where N is the evaluated number of jumps from the customer to the free vehicle (we have picked N = TTL/2) and jump Tally is the quantity of 2 hubs as of now went by on the way from the customer to the free vehicle. For example, the likelihood is diminished by 1/2^N at the main hub and 1/2^(N−1) at the second hub. The likelihood is decreased by 1/2 for every hub found more distant than TTL/2 jumps far from the customer. This procedure rehashes until the point that a free vehicle is come to or no further course is accessible. On the off chance that a free vehicle is found and the driver has consented to get the customer, the BookSM makes a ReportSM. This SM utilizes land directing to send the booking data back to the customer. Then again, if no further course is accessible and no free vehicle is discovered, BookSM plays out a new course revelation from the present hub. The overhead of this steering plan is required to be lower than that of Flooding when hubs move gradually, and many solicitations happen inside a constrained area and time interim (e.g., when many individuals book vehicles outside a theater when the play simply wrapped up).

3.3 Matching algorithm

INPUT:

(Required) A list of car objects.

(Required) A list of person objects.

OUTPUT:

A list of matching ambulance and person objects.

2. AN AMBULANCE OBJECT

An ambulance object: A dictionary that could have up to 5 attributes:

(Required) Coordinates: Latitude and longitude of the car.

(Optional) Plate number: The car plate number.

(Optional) Driver name: Name of the driver.

(Optional) Car model: Corresponding ambulance model.

(Optional) Rating: The overall rating of the ambulance.

Example of an ambulance object:


```
{
   "plateNumber": "UMY341",
   "driverName": "James Black",
   "carModel": "Tesla Model S",
   "rating": 4.95,
   "coordinates": {
      "lat": 47.601324,
      "long": -122.331820
}
```


3. A PERSON OBJECT

A person object: A dictionary that could have up to 3 attributes:

(Required) Coordinates: Latitude and longitude of the person.

(Optional) Name Name of the given person.

(Optional) Rating: Rating of the given person.

Example of a person object:


```
{
   "name": "Zeus da Deus",
   "rating": 4.88,
   "coordinates": {
      "lat": 47.601023,
      "long": -122.333751
   }
}
```


4. OUTPUT

A list of matching car and person objects: Gives a matching between the group of cars and people. The matching is optimal.

Example of an output:

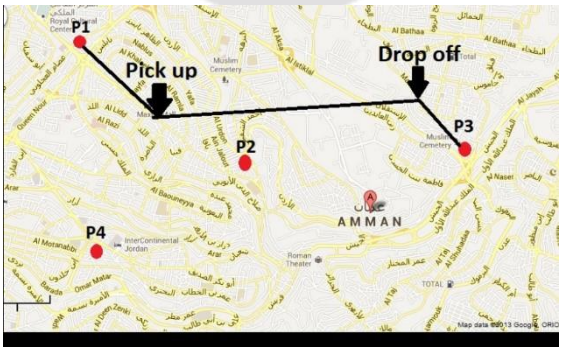

```
[
  {
   "customer": {
     "rating": 4.88,
     "name": "Customer 2",
     "coordinates": {"lat": 47.608575, "long": -122.324481}
   },
   "car": {
     "carModel": "Audi A6",

     "plateNumber": "ABC762",
     "driverName": "Driver 1",
     "coordinates": {
```

```
      "lat": 47.607731,
      "long": -122.323537
    },
    "rating": 4.88
  }
 },
 {
  "customer": {
    "rating": 4.88,
    "name": "Customer 1",
    "coordinates": {"lat": 47.601023, "long": -122.333751}
  },
  "car": {
    "carModel": "Tesla Model S",
    "plateNumber": "UMY341",
    "driverName": "Driver 3",
    "coordinates": {"lat": 47.601324, "long": -122.33182},
    "rating": 4.95
  }
 },
 {
  "customer": {
    "rating": 4.55,
    "name": "Customer 4",
    "coordinates": {"lat": 47.663577, "long": -122.379627}
  },
  "car": {
    "carModel": "Porsche Macan",
    "plateNumber": "HJK827",
    "driverName": "Driver 4",
```

```
    "coordinates": {"lat": 47.66728, "long": -122.383575},
    "rating": 4.99
   }
  },
  {
   "customer": {
    "rating": 4.55,
    "name": "Customer 3",
    "coordinates": {"lat": 47.616474, "long": -122.347827}
   },
   "car": {
    "carModel": "Porsche Macan",
    "plateNumber": "HJK827",
    "driverName": "Driver 2",
    "coordinates": {
     "lat": 47.618209,
     "long": -122.350101
    },
    "rating": 4.99
   }
  }
]
```


3.4 Probabilistic On-Demand



This directing calculation assembles steering tables on-interest for hubs lying in a TTL-bounces run from the customer. The directing table at a hub can be shared among various BookSMs, and it comprises of sections with the likelihood of finding a free vehicle through the hub's neighbors. Since the course to each neighbor could be negated because of versatility, each steering passage records its last refresh time; the framework utilizes this data to age directing sections. The Probabilistic On-Request component is like the receptive directing conventions in MANET, for example, AODV [31]. BookSM triggers the Probabilistic On-Request course disclosure on any hub where the directing table does not exist, the greatest likelihood is too little, or the correspondence fizzled because the hub with most extreme likelihood isn't a neighbor of the present hub any longer. To do as such, BookSM makes a DiscoverySM, which relocates in the system to search for a free vehicle and pieces on a steering tag for a timeout esteem. We have received a timeout conspire as opposed to sitting tight for a specific number of DiscoverySMs to return (i.e., we could have sat tight for one DiscoverySM for each immediate neighbor) since it is irrational to do as such in an unpredictable specially appointed system. If another BookSM asks for lands at a similar hub, it will hinder on a similar tag as opposed to making another DiscoverySM. In this manner, just the main demand refreshes the directing table at that hub. The second and consequent solicitations will utilize the steering table made by the primary demand when they proceed after the timeout. Figure 4 demonstrates the Probabilistic On-Request course disclosure in real life. The DiscoverySM brings forth itself at every hub. The youngster SM communicates itself to its neighbors, and the parent hinders with a timeout sitting tight for every one of its kids to return with directing data (i.e., the likelihood to locate a free vehicle through this hub). Every kid stores the address of its source hub (which as per the calculation is one-jump away). Once a DiscoverySM achieves a hub found TTL-jumps far from the customer, it will report the likelihood to its source as either 0.5 or 0.0, contingent upon whether the vehicle is free or possessed, separately. Note that each hub navigated by a DiscoverySM, except the last bounce, has a blocked DiscoverySM. At the point when the DiscoverySM unblocks (after timeout), it will accumulate every one of the probabilities announced by its tyke SMs and report back to its source hub. This procedure proceeds until the best level DiscoverySM is come to. The courses made by this calculation shape a Coordinated Non-cyclic Chart, established at the customer station. The DAGs made by various customer stations could cover each other. The likelihood, Pn ($1 \leq n < TTL$), announced by every hub found n trusts far from the customer to its source is given by:

Pn =1/21 + max(Pn+1) if n is involved,

½ max(Pn+1) if n is accessible.

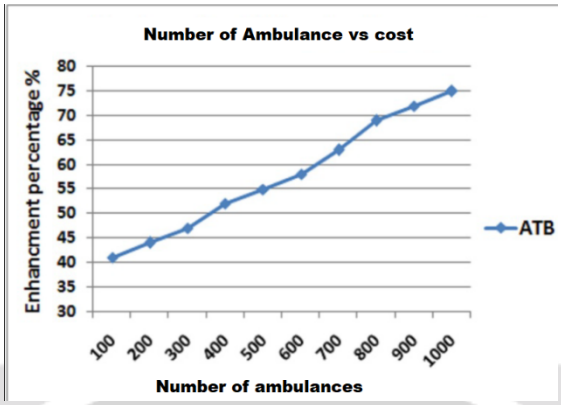
The timeout for the parent DiscoverySM at every hub is

characterized by:

2 × oneHopT time × (TTL − currentHops) × L,

Where oneHopTime is the ideal opportunity for single bounce relocation. Accordingly, (2 × oneHopTime × (TTL − currentHops)) is the aggregate relocation time. To represent diverse variables influencing the movement time, (for example, Macintosh layer conflict), we presented (L > 1) as a fudge factor. L is our endeavor to guarantee that the youngster Disclosure SMS could get back before the parent SM times out. As indicated by our trial comes about, the more conflicts, the bigger L ought to be. For example, a fudge factor L = 3 functions admirably for vehicles with 6 or less neighbors. At the point when a customer conveys a demand, BookSM just checks the nearby directing table and moves to the neighbor with the biggest likelihood. In the meantime, it refreshes the steering table in view of the TTL utilized by the DiscoverySM. When it picks the following bounce, BookSM isolates the relating likelihood by 2^(N−hopCount), where N is the evaluated number of jumps from the customer to the free vehicle (we have picked N = TTL/2) and jump Tally is the quantity of 2 hubs as of now went by on the way from the customer to the free vehicle. For example, the likelihood is diminished by 1/2^N at the main hub and 1/2^(N−1) at the second hub. The likelihood is decreased by 1/2 for every hub found more distant than TTL/2 jumps far from the customer. This procedure rehashes until the point that a free vehicle is come to or no further course is accessible. On the off chance that a free vehicle is found and the driver has consented to get the customer, the BookSM makes a ReportSM. This SM utilizes land directing to send the booking data back to the customer. Then again, if no further course is accessible and no free vehicle is discovered, BookSM plays out a new course revelation from the present hub. The overhead of this steering plan is required to be lower than that of Flooding when hubs move gradually, and many solicitations happen inside a constrained area and time interim (e.g., when many individuals book vehicles outside a theater when the play simply wrapped up).

3.5 Probabilistic Proactive



In an exceedingly powerful system, the Probabilistic On-Request calculation could perform ineffectively. For instance, the passage with the biggest likelihood can wind up noticeably invalid before it is used (i.e., the hub moves out of remote range). This circumstance triggers DiscoverySM, a generally costly process since the BookSM needs to sit tight for 2 × TTL hub traversals while directing tables are remade sans preparation. To have "fresher" data in the steering table, we propose a third directing calculation: Probabilistic Proactive. Proactive directing calculations are known to perform ineffectively in profoundly portable systems [14]. This calculation, be that as it may, can possibly function admirably for MHvehicle in light of the fact that MHvehicle does not require end-to-end exchanges of mass information, but instead to locate any free vehicle in a given locale. Like the Probabilistic On-Request conspire, this calculation constructs directing tables with probabilities to discover free vehicles in every passage. The directing updates, be that as it may, are activated proactively by the nearness of free vehicles, not by on-request asks for from customer stations. Each free vehicle utilizing this calculation communicates an UpdateSM occasionally. UpdateSMs contain a little TTL incentive to restrain the extent of flooding and a special identifier freevehicleID of the free vehicle which produced it. After accepting this refresh, the immediate neighbors of this free vehicle refresh their directing table with (freevehicleID, 0.5) and decrement the TTL of the UpdateSM. The UpdateSM is then additionally engendered until the TTL achieves 0. Each bounce gets a couple (previousvehicleID, one 2N) in the UpdateSM, where N, ($1 \leq N \leq TTL$), is the quantity of jumps from the free vehicle. In the event that previousvehicleID isn't in the steering table, another passage will be made. Something else, the present likelihood is refreshed with the new esteem. Litoke the Probabilistic On-Request conspire, a maturing interim is utilized to cleanse old data. This calculation produces covering DAGs established at each free vehicle. At the point when a customer asks for a free vehicle, BookSM will relocate bounce by-jump through the sections with the biggest likelihood (up to TTL bounces) until the point that a free vehicle is found. On the off chance that a free vehicle is discovered, MHvehicle will continue with the following period of the three-way handshake convention. A BookSM relocating to a free vehicle refreshes the probabilities in the steering tables similarly it does in the Probabilistic On-Request calculation. In the event that no free vehicle is found after TTL jumps, BookSM will report the inability to the customer. Then again, if a broken connection is found (i.e., the objective hub vanished), a DiscoverySM will be infused to locate a free vehicle as opposed to remaking the entire steering table.

4 Performance Evaluation

To compare the performance of the three routing algorithms under different scenarios, we have simulated using the ns-2 simulator [9], enhanced with the CMU-wireless extensions [8]. Different scenarios are used to test sensitivities of the algorithms to various application and network parameters. In this section, we describe our experiments and present the corresponding results.

4.1 Scenario Generator

We have built up our situation generator device given set dest, a generator instrument for irregular way point versatility show, created at Carnegie Mellon College to create activity situation for urban communities with lattice streets (e.g.,Manhattan). In our movement display, we util ize a considerable number of substituted single bearing streets along each measurement of The network. Vehicles can change their paths inside a similar street freely of each other. The likelihood of remaining in a similar path is 0.6, though the likelihood of changing the path is 0.4 either to one side or one side. Likewise, vehicles can switch their streets at crossing points. With level with probabilities, a vehicle picks either to remain on a similar street or change to the street it meets with. At the point when a vehicle achieves the keep going convergence out and about, it is compelled to change to the street it meets with. Customers carry on also to the vehicles at the crossing points, aside from that they can move in the two bearings of a street. In this way, customers at crossing points can pick consistently between the three new headings.

The vehicles select their speed as [average speed ± (0.25×average speed×rand())], where rand() restores a

Consistently disseminated irregular number from the range [0, 1]. The customers select their speed consistently from the range [0, 1] meters/sec. At first, every vehicle advances toward an irregular goal along its street utilizing its as of now chose speed. Once a vehicle achieves its goal, it chooses another arbitrary goal along its street and additionally another speed. On the off chance that the chose goal is the following convergence out and about, the vehicle sets its goal to the crossing point. Customers select their new goals in a comparative way. For every one of the reproductions, we settled the width of the frameworks to 2,000 meters, while the length is 3,000 meters. The streets are circulated consistently as six vertical streets and ten level streets with two paths for every street. The normal hole between progressive vehicles on a similar path is 185 meters. The situation generator places [number of paths × (city widthgap× number vertical streets + city length

gap×number flat roads)] = 410 vehicles and 200 customers equitably circulated on the streets. We utilized IEEE 802.11b as the remote media, with an information transmission rate of 11Mb and a transmission scope of 250 meters. Amid our open-air tests, we discovered that the remote transmission run is under 250 meters. Nonetheless, we have possessed the capacity to reestablish this transmission extend utilizing outer receiving wires.

4.2 Simulation Results

For all the reproduction runs, the initial 200 seconds speak to a warm-up period (i.e., no vehicle ask for happens inside this period). Every customer sends a vehicle ask (BookSM) Once amid the reproduction time frame, at once picked haphazardly after the warm-up period. BookSM is unicasted for the on-request and proactive directing instruments, while it is communicated for the flooding instrument.

Note that on the off chance that a customer utilizes the Probabilistic On-request or Probabilistic Proactive steering instruments and has no directing table data, it will communicate the BookSM or DiscoverSM (to improve the work, we will allude just to BookSM).

The greatest number of jumps a demand can spread (i.e., TTL) scanning for a free vehicle is set to 20 bounces.Once a free vehicle gets a vehicle ask for, it sets its status to held for a period picked haphazardly in the vicinity of 2 and 5 seconds and sends back a ReportSM. Once an affirmation (ConfirmSM) is gotten from a customer, the vehicle status is set to booked (involved) for an arbitrary period picked consistently finished the range [300, 1800] seconds. Notwithstanding the directing instrument, a customer re-communicates a vehicle ask for on the off chance that it doesn't get an answer from any free vehicle inside an irregular period picked consistently finished the range [2, 5] seconds. The normal vehicle speed is set to 15 meters/second with zero delay time. We set the fudge factor L for the on-request in strument to 3. For the proactive component, we set the most extreme sections in the steering table (max) to 20, and the intermittent refresh interim (UPD) for updates to 5 seconds. The most extreme number of bounces (i.e., TTL) for UpdateSMs created by free vehicles is set to 3. Every section in the steering table terminates following 2.5×UPD seconds of the last refresh.

4.2.1 Effects of Number of Free vehicles

We first look at the effect of the initial number of free vehicles. An initial free vehicle can be booked during the simulation and then switches its status back to free after the booking period. On the other hand, all the vehicles initialized as booked at the beginning of the simulation remain booked during the simulation period. The initially booked vehicles act only as relays. For these runs, we fix the simulation time to 1500 seconds, with the 200 seconds warm-up period. The clients request vehicles uniformly over the next 1200 seconds, leaving the last 100 seconds for any unfinished requests. A customer begins with a unicast ask for if conceivable. If the unicast can't be utilized or just comes up short, the customer changes to at least one communicate demands until the point when it books a vehicle. For instance, the '155\On-request' bar shows that lone 13 customers were prevailing to get a vehicle utilizing just unicast demands. Whatever remains of the customers needed to change to communicate demands. Of which, 141 customers prevailing to book vehicles utilizing a solitary communicate ask for, while the rest planned out and needed to re-communicate the demand (i.e., 26 customers required two communicates to book a vehicle, customers require three communicates, and the rest of the 12 customers required at least four communicates).

All vehicle requests in the flooding mechanism, where norouting tables are used, are broadcast requests. As thea number of initial free vehicles decreases, a single client request could be repeated several times due to the unavailability of free vehicles or dropping of the reply/confirmation messages because of collisions. Most of the vehicle requests of the proactive mechanism are unicast that exploit the stored routing tables. However, when the number of free vehicles decreases, some of those unicasts fail, and the clients use broadcast requests. For the on-demand mechanism, because the clients were distributed over the city, just a few of them could benefit from other requests and succeed to reserve vehicles using unicast requests, while the rest of the clients switched to broadcast requests. It plots the cumulative number of clients against the response time of the booking process. A client booking ends when a vehicle gets a reservation confirmation from its client. The on-request system is the most exceedingly awful because a free vehicle continues sending the solicitations forcefully after it answers back to the customer to empower different customers to refresh their neighborhood steering tables, while in the flooding component, the proliferation of the demand ends at a free vehicle.The overhead of the instruments in regards to the normal number of messages transmitted per asks for (counting the UpdateSMs for the proactive component). We found that the overhead of the proactive system is higher than the flooding instrument particularly when quantity of free vehicles is extensive. This is because of vethe occasional refresh messages from the vehicles in the framework. The on-request instrument has the most noteworthy overhead due to its forceful telecom system.

Due to a large number of reservations for each request in the flooding and on-demand mechanisms, subsequent vehicle requests may find that all closer vehicles are reserved, and therefore, the requests propagate deep into the network. The on-request system is the most exceedingly awful in light of the fact that a free vehicle continues sending the solicitations forcefully after it answers back to the customer to empower different customers to refresh their neighborhood steering tables, while in the flooding component, the proliferation of the demand ends at a free vehicle.The overhead of the instruments in regards to the normal number of messages transmitted per ask for (counting the UpdateSMs for the proactive component). We found that the overhead of the proactive system is higher than the flooding instrument particularly when the quantity of free vehicles is extensive. This is because of the occasional refresh messages from the vehicles in the framework. The on-request instrument has the most noteworthy overhead due to its forceful telecom system. The figure illustrates how a randomly generated request issimulated. The ABS locates the customer address and then calculates the closest parking to the customer Location, which in figure1 is parking (P1). ABS assigns the first vehicle in the queue from the parking (P1) and removes it out of the queue. The vehicle takes the customer to the required destination and drops off the customer. Then the vehicle is assigned to the nearest parking which in figure1 is parking (P3). For this booking, the total distance is calculated and is summed up with all simulation bookings and compared with the random model. From the distance, time and fuel consumption are calculated. This simulation will indicate the effect of ABS on fuel consumption which is calculated from a distance and the time needed to travel from one point to another. The figure shows the effect of

increasing the number of vehicles on the enhancement on cost, where cost represents the saving in fuel consumption and the time because they are related. The fuel consumption and time are calculated from the distance where the distance is multiplied by a fixed number to represent the fuel cost and time. The ABS saves time and money that can reach up to 75% when compared to having no booking system. When increasing the number of requests

Per hour the saving decreases and that is because there will be a lot of vehicles not available in some parks because they are still busy. Therefore vehicles from other markings are assigned to these bookings which will increase the distance and the cost of the booking. Also, vehicles are assigned to empty parkings instead of the nearest parking to make up for the shortage in those parkings which also increases the cost and time.

5 Conclusion and future work

There is a lack of medical transportation infrastructure which gives very high priority and demand for vehicles. Unfortunately, vehicles do not have central organizing offices for booking or scheduling vehicles which forces vehicle drivers to drive around for customers and customers to go down to the main streets looking for vehicles. This results in many problems such as congestion, oil consumption, and a waste of time. In this paper, an automated ambulance scheduling and booking system using Matching algorithm are implemented for mobiles to facilitate the booking of vehicles. Initial simulation results show that using Matching algorithm system can save up to 80% in time and oil consumption over the current situation where no scheduling is available. For the near future, the system should be uploaded and tested in reality for a few number of vehicles. We are considering several extensions to MHCab as Future work. In this paper, we assumed the ambulance drivers are cooperative and willing to propagate data between each other. We intend to explore the execution of MHCab when it utilizes diverse classes of taxicabs in which correspondence happens just between taxis having a place with a similar class. Emergency vehicle classes show the circumstance of various taxis organizations, where each organization isn't willing to course messages for a contending organization. Another augmentation will enable possessed taxis to go about as free hopeful taxicabs in light of their planned drop-off area and time. For instance, it is valuable to consider a taxicab that is booked to drop-off a customer an insignificant 10 feet from the new customer inside a moment, which is far superior to booking 15 minutes away free taxi. We likewise plan to ponder the utilization of needs for the free taxis given, for instance, their separations to the customer, to what extent they have been sitting out of gear, or the quantity of customers served in the most recent hour. The objective of such need framework is to ensure reasonableness and optimality.

6 References

[1] http://www.citycab.com.sg/services/

nts/sms booking.html.

[2] http://www.comfort-transportation.com.sg/

booking svcs.html.

[3] Active Networks. http://nms.lcs.mit.edu/activeware/.

[4] Auriga Systems. http://www.auriga.co.uk/.

[5] HP iPAQ. http://h71016.www7.hp.com.

[6] Motorola A760. http://www.motorola.com.

[7] Sony Ericsson P900,

http://www.sonyericsson.com/p900/.

[8] The Monarch Group at Rice University.

http://www.monarch.cs.rice.edu/.

[9] The Network Simulator ns-2,

http://www.isi.edu/nsnam/ns/.

[10] Toshiba PocketPC. http://www.csd.toshiba.com.

[11] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing, 7(5):275–286, October 2003.

[12] C. Borcea, C. Intanagonwiwat, A. Saxena, and L. Iftode. Self-Routing in Pervasive Computing Environments using Smart Messages. In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), pages 87–96, Dallas-Fort Worth, TX, March 2003.

[13] C. Borcea, D. Iyer, P. Kang, A. Saxena, and L. Iftode. Cooperative Computing for Distributed Embedded Systems. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), pages 227–236, Vienna, Austria, July 2002.

[14] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 85–97, ACM Press New York, NY, USA, 1998.

[15] Z. Chen, H. Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In Proceedings of the 2nd ACM International Symposium on Mobile Ad-hoc Networking and Computing, pages 247–250,

Long Beach, CA, Oct 2001.

[16] I. Chisalita and N. Shahmehri. A Peer-to-Peer Approach to Vehicular Communication for the Support of Traffic Safety Applications. In Proceedings of the 5th IEEE International Conference on Intelligent Transportation Systems, Singapore, Sept 2002.

[17] D.Wetherall. Active Network Vision Reality: Lessons from a Capsule-based System. In Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP 1999), pages 64–79, Charleston, SC, December 1999. ACM Press, New York, NY.

[18] S. Dashtinezhad, T. Nadeem, B. Dorohonceanu, C. Borcea, P. Kang, and L. Iftode. TrafficView: A Driver Assistant Device for Traffic Monitoring based on Car-to-Car Communication. In Proceedings of the 59th IEEE Semiannual Vehicular Technology Conference, May 2004.

[19] R. Gray, G. Cybenko, D. Kotz, and D. Rus. Mobile Agents: Motivations and State of the Art. In J. Bradshaw, editor, Handbook of Agent Technology. AAAI/MIT Press, 2002.

[20] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications:the Fleetnet project. In Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 259–262, Long Beach, CA, 2001.