ON THE PROBABILITY OF *K*-CONNECTIVITY IN WIRELESS AD HOC NETWORKS UNDER DIFFERENT MOBILITY MODELS

Natarajan Meghanathan[1] and Sireesha Gorla[2]

[1,2]Jackson State University, 1400 Lynch St, Jackson, MS, USA
[1]natarajan.meghanathan@jsums.edu, [2]sireesha.gorla@gmail.com

ABSTRACT

We compare the probability of k-Connectivity of an ad hoc network under Random Way Point (RWP), City Section and Manhattan mobility models. A Network is said to be k-Connected if there exists at least k edge disjoint paths between any pair of nodes in that network at any given time and velocity. Initially, for each of the three mobility models, the movement of the each node in the ad hoc network at a given velocity and time are captured and stored in the Node Movement Database (NMDB). Using the movements in the NMDB, the location of the node at a given time is computed and stored in the Node Location Database (NLDB). A weighted graph is created using the location of the nodes from NLDB, which is converted into a residual graph. The k-Connectivity of this residual graph is obtained by running Ford-Fulkerson's algorithm on it. Ford Fulkerson's algorithm computes the maximum flow of a network by recording the flows assigned to different routes from each node to all the other nodes in the network. When run for a particular source-destination pair (s, d) pair on a residual network graph with unit edge weights as capacity, the maximum flow determined by Ford-Fulkerson' algorithm is the number of edge disjoint s-d paths on the network graph. Simulations show that the RWP model yields the highest probability of k-Connectivity compared to City Section and Manhattan mobility models for a majority of different node densities and velocities considered. Simulation results also show that, for all the three mobility models, as the k value increases, the probability of k-Connectivity decreases for a given density and velocity and as the density increases the probability of k-Connectivity increases.

KEYWORDS

Wireless Ad hoc Networks, k-Connectivity, Mobility Models, Probability, Ford-Fulkerson Algorithm, Simulations

1. INTRODUCTION

A mobile ad hoc network (MANET) is a collection of mobile wireless hosts which communicate directly with each other in the absence of a fixed infrastructure [1], with some constraints on the bandwidth of the wireless links. Communication between any two hosts, which are outside the transmission range of each other is performed through the intermediate hosts. The network in a MANET is decentralized where each wireless host has the routing functionality incorporated within it. Variable wireless link quality, propagation path loss, fading, multi-user interference, limited battery power, and rapid and unpredictable topological changes are some of the issues that need to be dealt in a MANET.

Vehicular Ad-hoc Networks (VANET) is an emerging, new type of MANET, where vehicles on the road form a MANET using wireless technology. Limited bandwidth, multi-hop communication and self-organization are some of the common characteristics that VANET shares with MANET. The main issue in a VANET is that the nodes move in a high speed with respect to each other and this in turn results in very frequent topology changes [2]. Battery power is not an issue with VANETs.

Evaluating the characteristics of ad hoc networking protocols is usually done through the use of simulation. Mobility model is an important component of a network simulation and usually plays an important role in understanding real world MANETs. A mobility model describes the movement patterns of mobile nodes within a network and the change of location, velocity and acceleration over time [3]. Initially the nodes are distributed randomly within a network and the mobility model controls the node movement within the network [4].

A number of mobility models were introduced for ad hoc networks and they vary widely in the movement characteristics of the nodes. The Random Waypoint mobility model, commonly used in MANET simulation studies, assumes that nodes can move randomly anywhere within a network region. On the other hand, the City Section and Manhattan mobility models commonly used in VANET simulation studies assume the network is composed of horizontal and vertical streets and a node is allowed to move only along these streets [3].

A Network is said to be *k*-connected if there exists at least *k* edge disjoint paths between any pair of nodes in that network at any given time and velocity. Equivalently, it is connected even if *k* nodes are removed. *k*-Connectivity of a network is different for different mobility model. Connectivity is one of the most important properties of a MANET. *k*-Connectivity of a network is a helpful tool to balance the load and energy level at the nodes and to enable secure reliable communication. In a *k*-connected wireless ad hoc and sensor networks, fault tolerance and robustness increase with increasing *k* value.

The rest of the paper is organized as follows: In Section 2, we briefly review the three mobility models considered. Section 3 describes the algorithms proposed to extract, store and use the node mobility profiles for each of the three mobility models. Section 4 briefly reviews the Ford-Fulkerson algorithm [5] and its use to determine the *k*-connectivity of an ad hoc network. Section 5 describes the simulation environment and presents the analysis of *k*-connectivity of an ad hoc network at different instants of the simulation as well as under diverse conditions of network density and mobility. Section 6 concludes the paper.

2. REVIEW OF THE MOBILITY MODELS

In this section, we provide a brief overview of the Random Waypoint mobility model commonly used in MANET simulation studies and the widely used City Section and Manhattan mobility models for VANET simulation studies. All the three mobility models [3] assume the network is confined within fixed boundary conditions. The Random Waypoint mobility model assumes that the nodes can move anywhere within a network region. The City Section and the Manhattan mobility models assume the network to be divided into grids: square blocks of identical block length. The network is thus basically composed of a number of horizontal and vertical streets. Each street has two lanes, one for each direction (north and south direction for vertical streets, east and west direction for horizontal streets). A node is allowed to move only along the grids of horizontal and vertical streets.

2.1 Random Waypoint Mobility Model

Initially, the nodes are assumed to be placed at random locations in the network. The movement of each node is independent of the other nodes in the network. The mobility of a particular node is described as follows: The node chooses a random target location to move. The velocity with which the node moves to this chosen location is uniformly randomly selected from the interval $[v_{min},\ldots,v_{max}]$. The node moves in a straight line (in a particular direction) to the chosen location with the chosen velocity. After reaching the target location, the node may stop there for a certain time called the pause time. The node then continues to choose another target location and moves to that location with a new velocity chosen again from the interval $[v_{min},\ldots,v_{max}]$. The selection of each target location and a velocity to move to that location is independent of the current node location and the velocity with which the node reached that location. In Figure 1, we observe that nodes *A* and *B* move independent of each other, in random directions with randomly chosen velocities.

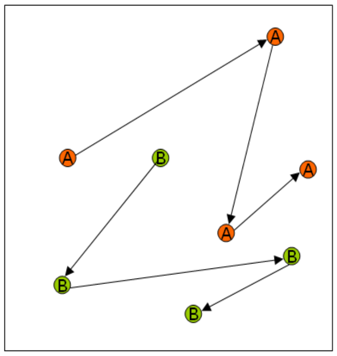
Figure 1: Movement under Random Waypoint Model

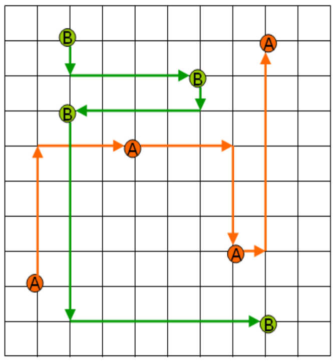
Figure 2: Movement under City Section Model

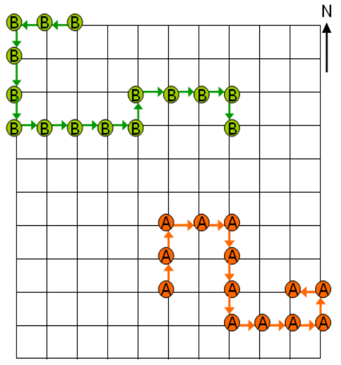

Figure 3: Movement under Manhattan Model

2.2 City Section Mobility Model

Initially, the nodes are assumed to be randomly placed in the street intersections. Each street (i.e., one side of a square block) is assumed to have a particular speed limit. Based on this speed limit and the block length, one can determine the time it would take move in the street. Each node placed at a particular street intersection chooses a random target street intersection to move. The node then moves to the chosen street intersection on a path that will incur the least amount of travel time. If two or more paths incur the least amount of travel time, the tie is broken arbitrarily. After reaching the targeted street intersection, the node may stay there for a pause time and then again choose a random target street intersection to move. The node then moves towards the new chosen street intersection on the path that will incur the least amount of travel time. This procedure is repeated independently by each node. In Figure 2, the movement of two nodes *A* and *B* according to the City Section mobility model has been illustrated.

2.3 Manhattan Mobility Model

Initially, the nodes are assumed to be randomly placed in the street intersections. The movement of a node is decided one street at a time. To start with, each node has equal chance (i.e., probability) of choosing any of the streets leading from its initial location. In Figure 3, to start with, node *A* has 25% chance to move in each of the four possible directions (east, west, north or south), where as node *B* can move only either to the west, east or south with a 1/3 chance for each direction. After a node begins to move in the chosen direction and reaches the next street intersection, the subsequent street in which the node will move is chosen probabilistically. If a node can continue to move in the same direction or can also change directions, then the node has 50% chance of continuing in the same direction, 25% chance of turning to the east/north and 25% chance of turning to the west/south, depending on the direction of the previous movement. If a node has only two options, then the node has an equal (50%) chance of exploring either of the two options. For example, in Figure 3, once node *A* reaches the rightmost boundary of the network, the node can either move to the north or to the south, each with a probability of 0.5 and the node chooses the north direction. After moving to the street intersection in the north, node *A* can either continue to move northwards or turn left and move eastwards, each with a probability of 0.5. If a node has only one option to move (this occurs when the node reaches any of the four corners of the network), then the node has no other choice except to explore that option. For example, in Figure 3, we observe node *B* that was traveling westward, reaches the street intersection, which is the corner of the network. The only option for node *B* is then to turn to the left and proceed southwards.

3. Algorithms to Generate Node Mobility Profile and Determine Node Locations at a Particular Time Instant

This section outlines the algorithms to generate the mobility profile for each node in the network and also outlines the algorithms to determine the location of a node at any time instant based on the mobility profiles generated.

3.1 Random Waypoint Model Node Movement Generator

```
Input: Velocity $v$, Simulation Time $st$, Node ID $i$
Auxiliary Variables:
        startTime; // the beginning time of a direction change (waypoint)
        endTime; // the ending time of a waypoint
        time $t$; // current time of node movement
        velocity $v$; // velocity of the node
Initialization:
        startTime $\leftarrow$ 0
        endTime $\leftarrow$ 0
        $t \leftarrow 0$
Output: NMDB$_i$; // Node mobility database for node $i$


Begin RWP-Node-Movement-Generator

Step 1: Generate a random point $(x_1, y_1)$ within a 1000*1000 Square Unit area.
Step 2: Generate a random point $(x_2, y_2)$
Step 3: Compute distance = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
Step 4: Compute Angle = $\frac{(x_1 - x_2)}{(y_1 - y_2)}$
Step 5: Compute transTime = distance / $v$
Step 6: endTime $\leftarrow$ endTime + transTime
Step 7: Store [startTime, endTime; $(x_1, y_1)$ $(x_2, y_2)$, $v$] in a Node Mobility Database (NMDB)
Step 8:
            $x_1 \leftarrow x_2$,
            $y_1 \leftarrow y_2$,
            startTime $\leftarrow$ endTime,
            $t \leftarrow t$ + transTime
Step 9: if $(t \leq st)$
              go to Step2
           else
              return NMDB$_i$

End RWP-Node-Movement-Generator
```

Figure 4: Algorithm to Generate Mobility Profile under the Random Waypoint Model

3.2 City Section Node Movement Generator

```
City Section Mobility Model
Let there be a set of nodes 'N', where N = {$N_1$, $N_2$, $N_3$……..$N_n$}, where $n$ is the number of nodes.
Input: Street Intersection Graph SIG (maxRows, maxCols, blockLength, ILDB)
                maxRows – Number of horizontal roads in the graph
                maxColumns – Number of vertical roads in the graph
                blockLength – The length of a block of road in the graph
                ILDB – Database storing the location of each intersection in the SIG,
        Speed Limit (Velocity) $v$ m/s, Simulation Time $st$
Auxiliary Variables:
    startTime; // the beginning time of a direction change (waypoint)
    endTime; // the ending time of a waypoint
    time t; // current time of node movement
Initialization:
    startTime $\leftarrow$ 0
    endTime $\leftarrow$ 0
    $t \leftarrow 0$
Output: $NMDB_i$; // Node Mobility database for node $i$

Begin City Section-Node-Movement-Generator

Step1: Generate a Random Intersection Point ($x_1$, $y_1$) with in the given graph
Step2: Generate a Random Intersection Point ($x_2$, $y_2$)
Step3: Find the path $P$ with the minimum number of street intersections between ($x_1$, $y_1$)
       and ($x_2$, $y_2$) using the Dijkstra's shortest path algorithm.
Step4: Compute distanceTraveled = (blockLength) * ($P_{size}$)
       where $P_{size}$ – the number of intermediate street intersections in $P$
Step5: Compute $transTime = \frac{\text{distanceTraveled}}{v}$
Step6: endTime $\leftarrow$ endTime + transTime
Step7: Store [endTime; ($x_1$, $y_1$) ($x_2$, $y_2$), $v$] in a Node Mobility Database (NMDB)
Step8:
       $x_1 \leftarrow x_2$,
       $y_1 \leftarrow y_2$,
       startTime $\leftarrow$ endTime,
       $t \leftarrow t$ + transTime
Step9: if ($t \leq$ st) go to Step2
     else
         return $NMDB_i$
End City Section-Node-Movement-Generator
```

Figure 5: Algorithm to Generate Mobility Profile under the City Section Mobility Model

3.3 Manhattan Node Movement Generator

Let there be a set of nodes 'N', where N = {N_1, N_2, N_3……..N_n}, where *n* is the number of nodes.

```
Input: Street Intersection Graph SIG (maxRows, maxCols, blockLength, ILDB)
            maxRows – Number of horizontal roads in the graph
            maxColumns – Number of Vertical roads in the graph
            blockLength – The length of a block of road in the graph
            ILDB – Database storing the location of each intersection in the SIG,
            $(x_I, y_I)$ -next intersection to which a node moves
            Speed Limit (Velocity) $v$ m/s, Simulation Time $st$
Auxiliary Variables:
    startTime; // the beginning time of a direction change (waypoint)
    endTime; // the ending time of a waypoint
    time t; // current time of node movement
Initialization:
    startTime $\leftarrow$ 0;      endTime $\leftarrow$ 0;      $t \leftarrow 0$
Output: $NMDB_i$; // Node Mobility database for node $i$

Begin Manhattan-Node-Mobility-Generator

Step1: Generate a Random Intersection Point $(x_I, y_I)$ within the given graph SIG
Step2: Let $(x_S, y_S) \leftarrow (x_I, y_I)$
Step3: Let $S_I$ be the set of all neighboring intersections of $(x_S, y_S)$ and $n_I$ be number of
       elements in $S_I$.
Step4: if $(|S_I| = 1)$ // $S_I = [(x_A, y_A)]$
          $(x_I, y_I) \leftarrow (x_A, y_A)$
Step5: if $(n_I = 2)$ // $S_I = [(x_A, y_A), (x_B, y_B)]$
          Generate a random number $r_I$ from 0 to 1
          if $(r_I < 0.5)$
             $(x_I, y_I) \leftarrow (x_A, y_A)$
          else
             nextI $\leftarrow (x_B, y_B)$
Step6: if $(n_I = 3)$ // $S_I = [(x_A, y_A), (x_B, y_B), (x_C, y_C)]$
       Choose the intersection $(x_A, y_A) \in S_I$ which is in the same axis as that of $(x_S, y_S)$
       Let $(x_B, y_B)$ and $(x_C, y_C)$ be the two intersections in $S_I$ that are not in the same axis as that
       of $(x_S, y_S)$ generate a random number $r_n$ from 0 to 1
          if $(r_n < 0.5)$
          nextI $\leftarrow (x_A, y_A)$
          else
             if $(0.5 < r_n < 0.75)$
                nextI $\leftarrow (x_B, y_B)$
                else nextI $\leftarrow (x_C, y_C)$
Step7: Compute distanceTraveled = blockLength
Step8: Compute transTime = $\frac{\text{distanceTraveled}}{v}$
Step9: Assign endTime+ = transTime
       Store [endTime; $(x_S, y_S), (x_I, y_I), v$] in Node Mobility Database (NMDB)
Step10: Assign $x_S \leftarrow x_I$, $y_S \leftarrow y_I$, startTime $\leftarrow$ endTime, t+ $\leftarrow$ transTime
Step11: If $(t \le st)$ go to Step3 Otherwise go to Step1

End Manhattan-Node-Mobility-Generator
```

Figure 6: Algorithm to Generate Mobility Profile under the Manhattan Model

The Node Movement Generator algorithm for each of the three mobility models outputs a Node Mobility Database (NMDB) for each node in the network. The NMDB of a node has the movement information of the node. The information includes the time at which the node started moving, starting location, ending location and the velocity of the node. The node location algorithm of a mobility model takes the corresponding NMDBs of all the nodes in the network and generates a Node Location Database (NLDB) which gives the location of each node at a given time.

3.4 RWP Node Location Generator

```
Let there be set of nodes 'N' where N = {$N_1$, $N_2$, $N_3$……..$N_n$} and T = {$t_1$, $t_2$, $t_3$, $t_4$,……..$t_{st}$} and
N, T∈ $NMDB_i$

Input: time t, Simulation Time st, NMDB of $N_i$;
Output: $NLDB_i$; // Node location database for node i

Begin RWP-Node-Location-Generator

Step1:  if ($t \in$ T) go to Step5
        else go to Step2
Step2: Iterate through NMDB of $N_i$ and find a value of '$t_j$' and '$t_{j+1}$' such that
        $t_j < t < t_{j+1}$
Step3: Compute fraction f = $\frac{t - t_j}{t_{j+1} - t_j}$
Step4: Let $(x_t, y_t)$ be the location at time t then

            Compute $x_t$ = f * $x_{j+1}$ + (1-fr) * $x_j$
            Compute $y_t$ = f * $y_{j+1}$ + (1-fr) * $y_j$
Step5: Store [ $N_i$; $(x_t, y_t)$ ,t] in Node Position Database (NLDB)

End RWP-Node-Location-Generator
```

Figure 7: Algorithm to Generate Node Location under the Random Waypoint Model

3.5 City Section Node Location Generator

```
Let there be a set of nodes 'N' where N = {$N_1$, $N_2$, $N_3$……..$N_n$} and T = {$t_1$, $t_2$, $t_3$, $t_4$,……..$t_{st}$}
and N, T∈ $NMDB_i$

Input: time t, Simulation Time st, Node Mobility Database (NMDB) of $N_i$; Velocity v;
Auxiliary Variables:
    blockLength b;//length of any street between two intersections
    TimePerBlock $T_B$; //time taken to travel a single bockLength of street
Initialization:
    $T_B = \frac{b}{v}$
Output: NLDB_i; // Node location database for node i

Begin City-Section-Node-Location-Generator

Step1: if ($t \in$ T)
            go to Step7
       else
            go to Step2
Step2: Iterate through NMDB and find a value of '$t_j$' and '$t_{j+1}$' such that
         $t_j < t < t_{j+1}$
Step3: Find the shortest path P on the street intersection graph.
         Let P be represented as $(x_j, y_j), (x_{k1}, y_{k1}), (x_{k2}, y_{k2}), \ldots\ldots\ldots(x_{kh}, y_{kh}), (x_{j+1}, y_{j+1})$,
         where $k_1, k_2, k_3, \ldots\ldots\ldots\ldots k_h$ are the street intersections forming the shortest path,
         and $t_{k1}, t_{k2}, t_{k3}, \ldots\ldots\ldots t_{kh}$ the times respectively.
         and $h$ is the number for street intersections between $(x_j, y_j)$ and $(x_{j+1}, y_{j+1})$
         Let the $l$ be the count, and $t_l$ be the time and count
         Initialize $l = 1$ and $t_l = t_j$.
Step4: Let $X_{start} = x_{kl}$, $Y_{start} = y_{kl}$ and $X_{end} = x_{kl+1}$, $Y_{end} = y_{kl+1}$
Step5: if ($t_l \geq t >= t_l + T_B$)
            $l = l + 1$
            Repeat Step4
       else
            Compute fraction $f = \frac{t - t_{kl}}{t_{kl+1} - t}$
Step6: Let $(x_t, y_t)$ be the location at time t then
         Compute $x_t = f * x_{kl+1} + (1-f) * x_{kl}$
         Compute $y_t = f * y_{kl+1} + (1-f) * y_{kl}$
Step7: Store [ $N_i$; $(x_t, y_t)$, $t$ ] in Node Position Database (NPDB)

End City-Section-Node-Location-Generator
```

Figure 8: Algorithm to Generate Node Location under the City Section Mobility Model

3.6 Manhattan Node Location Generator

Let there be a set of nodes 'N' where N = {N_1, N_2, N_3……..N_n} and T = {t_1, t_2, t_3, t_4,……..t_{st}} and N, T$\in$ NMDB$_i$

```
Input: time t, Simulation Time st, NMDB of N_i;
Output: NLDB_i; // Node location database for node i

Begin Manhattan-Node-Location-Generator

Step1: if ($t \in$ T)
            go to Step5
       else
```

```
        go to Step2
Step2: Iterate through NMDB of $N_i$ and find a value of '$t_j$' and '$t_{j+1}$' such that
        $t_j < t < t_{j+1}$
Step3: Compute fraction f = $\frac{t - t_j}{t_{j+1} - t_j}$
Step4: Let $(x_t, y_t)$ be the location at time $t$ then

            Compute $x_t = f * x_{j+1} + (1-fr) * x_j$
            Compute $y_t = f * y_{j+1} + (1-fr) * y_j$

Step5: Store [ $N_i$; $(x_t, y_t)$ ,$t$] in Node Position Database (NLDB)

End Manhattan-Node-Location-Generator
```

Figure 9: Algorithm to Generate Node Location under the Manhattan Mobility Model

4. Determining the k-Connectivity of a Residual Graph using Ford-Fulkerson Algorithm

From the NLDBs obtained using the Node Location Generators for a mobility model, a graph is created depending on the distances between the nodes and the transmission range of each node. A residual graph [5] is a directed graph where each edge has a positive residual capacity and is labeled by its residual capacity. For a given graph $G = (V, E)$ with source s and destination t, let f be the flow in G and $u, v \in V$ be a pair of vertices then, the additional amount of net flow that can be pushed from u to v before exceeding the capacity $c(u, v)$ is the residual capacity of (u, v), which is given by: $c_f(u, v) = c(u, v) - f(u ,v)$.

```
Input: Given a NLDB at a particular time $t$,
        Transmission rage $R$
Output: $G = (V, E)$
          $V$ – the set of all vertices corresponding to the nodes $N_1, N_2, \ldots, N_n$ where n is
               the number of nodes in the network
           $E$ – the set of all edges such that the distance between the two constituent nodes
               of an edge is less than or equal to the transmission range $R$.
Begin Graph Generator
  for $\forall i \in V$
    for $\forall j \in V-\{i\}$
      Step 1: Compute the distance $d_{ij} = \sqrt{(xi - xj)^2 + (yi - yj)^2}$
      Step 2: if ($d_{ij} <= R$)
                $(i, j) \in E$
                weight $(i, j) \leftarrow 1$
              end if
        end for
      end for
  End Graph Generator
```

Figure 10: Algorithm to Create a Weighted Graph for a Given NLDB

```
Input: Residual Graph $G_R$, initially $G_R = G(V, E)$

Auxiliary Variables:
        flow $f$;
        capacity $c$;
        flow capacity $c_f$;
        count connectivity $k_C$; //count which keeps track of the connectivity
Initialization: $k_C \leftarrow 0$

Output: $k$-Connectivity Database (KCDB) that has the set of all source-destination ($s$-$d$) paths
that has $k$-edge disjoint paths; In this research, $0 \leq k \leq 40$
Each entry in KCDB is a tuple $[k, SD_k]$ where $k$ is the number of edge-disjoint paths and $SD_k$ is
the set of all $s$-$d$ pairs that have $k$-edge disjoint paths

Begin Ford-Fulkerson-Algorithm for k-Connectivity
  for $\forall$ $s$-$d$ pair where $s \in V$ and $d \in V$
    $k_C \leftarrow 0$ // the number of edge-disjoint paths between $s$ and $d$
    Step1: for each edge $(u, v) \in E$
             do $f[u, v] \leftarrow 0$
                $f[v, u] \leftarrow 0$
                $c[u, v] \leftarrow$ weight$(u, v) \leftarrow 1$
                if $(v, u) \notin E$
                  $c[v, u] \leftarrow 0$
    Step2: if there exits an s-d path $P$ (i.e., a path from node $s$ to node $d$) in $GR$
             do $c_f(P) \leftarrow \min\{c_f(u, v): (u, v)$ is in $P\}$
                for each edge $(u, v)$ in $P$
                  do $f[u, v] \leftarrow f[u, v] + c_f(P)$
                     $f[v, u] \leftarrow -f[u, v]$
                     $c(u, v) = c(u, v) - f(u, v)$
                     $c(v, u) = c(v, u) - f(v, u)$
                     $k_C \leftarrow k_C + 1$
                       go to Step2
    Step3: Add $(s, d)$ to $SD_{kC}$
  end for

End Ford-Fulkerson-Algorithm for k-Connectivity
```

Figure 11: Finding the *k*-Connectivity of a Residual Graph using Ford-Fulkerson Algorithm

5. SIMULATIONS

Simulations have been conducted in a discrete-event simulator implemented by the authors in Java. The network dimensions are 1000m x 1000m. The network density is varied with 25 nodes (low density), 50 nodes (medium density) and 75 nodes (high density). The simulation time is 1000 seconds. The velocity is uniformly distributed in the range [0…. $V_{\max}$]. The $V_{\max}$ values used are 5m/s (representing low node mobility), 15m/s (representing medium node mobility) and 30m/s (representing high node mobility). Pause time is 0 seconds. The transmission range of each node is 250m. The mobility models used are Random Waypoint, City Section and Manhattan models. Using For-Fulkerson's algorithm, the *k*-Connectivity of the network is calculated at *k* = 1, 2, …, 10 and the time instants at which *k*-Connectivity is captured are 100, 600 and 900th seconds, as illustrated in Figures 12 through 20.

In low density networks, for all conditions node mobility (5m/s, refer Figure 12; 15m/s, refer Figure 13; and 30m/s, refer Figure 14), the Random Waypoint mobility model has the highest probability of *k*-Connectivity at lower values of *k* (*k* = 1, 2, 3) while Manhattan has the highest probability of *k*-Connectivity at medium (*k* = 4, 5, 6, 7) and higher (*k* = 4, 5, 6, 7) values. For medium density networks, for all conditions node mobility (5m/s, refer Figure 15; 15m/s, refer Figure 16; and 30m/s, refer Figure 17), the Random Waypoint mobility model has the highest probability of *k*-Connectivity at lower (*k* = 1, 2, 3), medium (*k* = 4, 5, 6, 7) and higher (*k* = 8, 9, 10) values of *k*. For high density networks, in conditions of low node mobility (5m/s, refer Figure 18), Random Waypoint mobility model has the highest probability of *k*-Connectivity at lower (*k* = 1, 2, 3), medium (*k* = 4, 5, 6, 7) and higher (*k* = 8, 9, 10) values of *k* at low velocity and high density. In conditions of moderate node mobility (15m/s, refer Figure 19), the Manhattan mobility model has the highest probability of *k*-Connectivity at lower values of *k* (*k* = 1, 2, 3) while the Random Waypoint model has the highest probability of *k*-Connectivity at medium (*k* = 4, 5, 6, 7) and higher (*k* = 8, 9, 10) values of *k* at medium mobility and high density. In conditions of high node mobility (30m/s, refer Figure 20), the Manhattan mobility model has the highest probability of *k*-Connectivity at lower values of *k* (*k* = 1, 2, 3) while the Random Waypoint model has the highest probability of *k*-Connectivity at medium (*k* = 4, 5, 6, 7) and higher (*k* = 8, 9, 10) values of *k* at high mobility and high density.

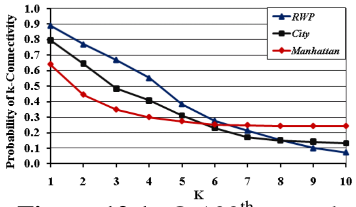

Figure 12.1: @ 100th second

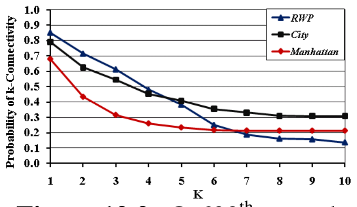

Figure 12.2: @ 600th second

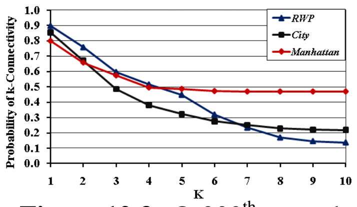

Figure 12.3: @ 900th second

Figure 12: Probability of *k*-Connectivity (Low Density, Low Mobility)

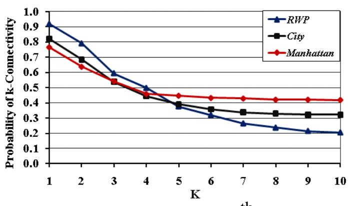

Figure 13.1: @ 100th second

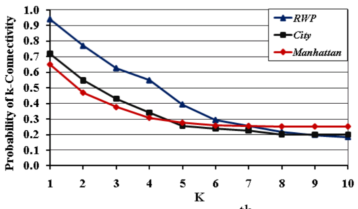

Figure 13.2: @ 600th second

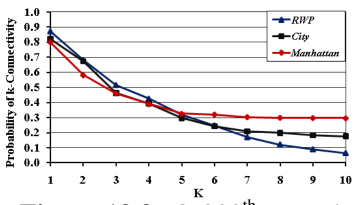

Figure 13.3: @ 900th second

Figure 13: Probability of *k*-Connectivity (Low Density, Moderate Mobility)

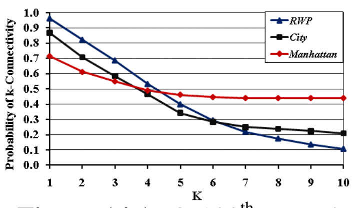

Figure 14.1: @ 100th second

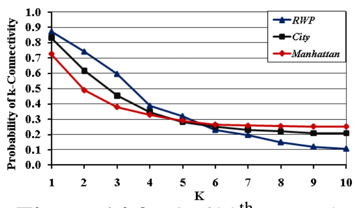

Figure 14.2: @ 600th second

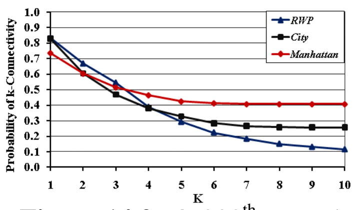

Figure 14.3: @ 900th second

Figure 14: Probability of *k*-Connectivity (Low Density, High Mobility)

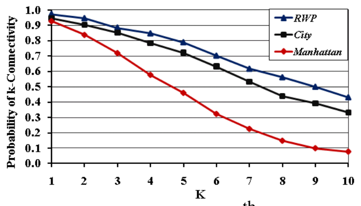

Figure 15.1: @ 100th second

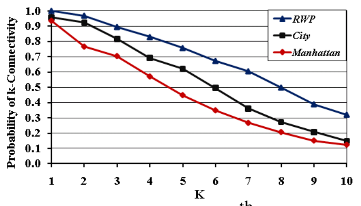

Figure 15.2: @ 600th second

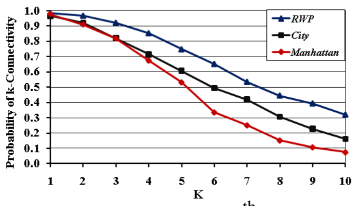

Figure 15.3: @ 900th second

Figure 15: Probability of *k*-Connectivity (Moderate Density, Low Mobility)

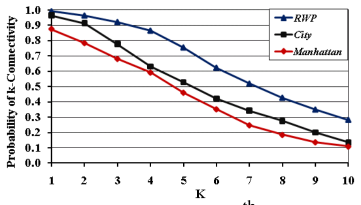

Figure 16.1: @ 100th second

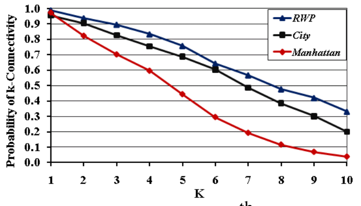

Figure 16.2: @ 600th second

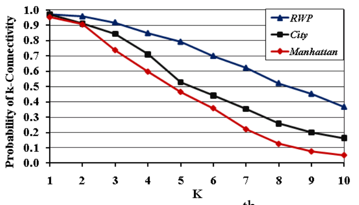

Figure 16.3: @ 900th second

Figure 16: Probability of *k*-Connectivity (Moderate Density, Moderate Mobility)

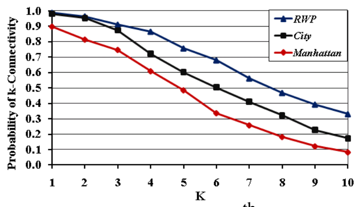

Figure 17.1: @ 100th second

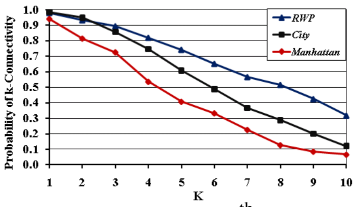

Figure 17.2: @ 600th second

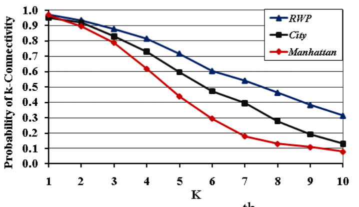

Figure 17.3: @ 900th second

Figure 17: Probability of *k*-Connectivity (Moderate Density, High Mobility)

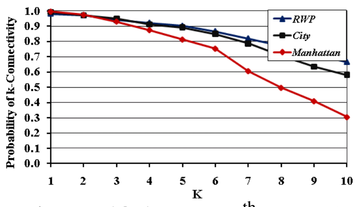

Figure 18.1: @ 100th second

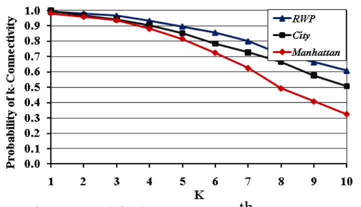

Figure 18.2: @ 600th second

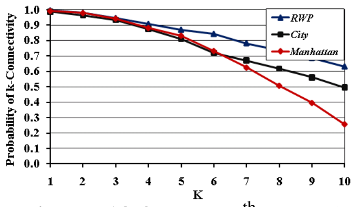

Figure 18.3: @ 900th second

Figure 18: Probability of *k*-Connectivity (Moderate Density, Low Mobility)

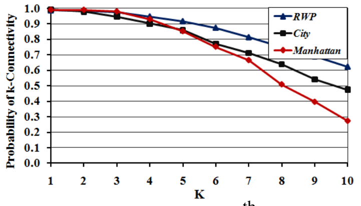

Figure 19.1: @ 100th second

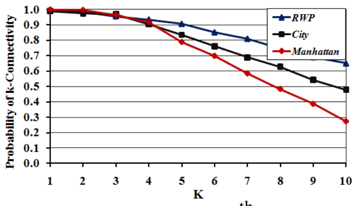

Figure 19.2: @ 600th second

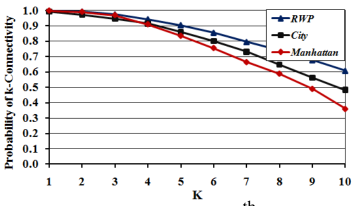

Figure 19.3: @ 900th second

Figure 19: Probability of *k*-Connectivity (Moderate Density, Moderate Mobility)

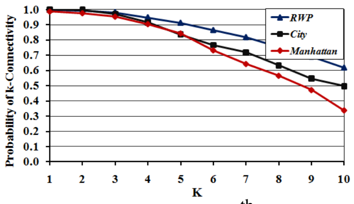

Figure 20.1: @ 100th second

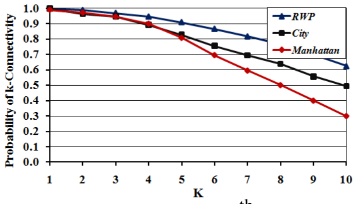

Figure 20.2: @ 600th second

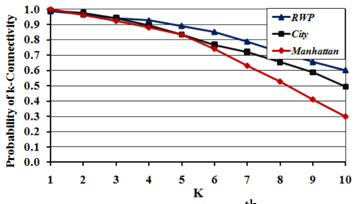

Figure 20.3: @ 900th second

Figure 20: Probability of *k*-Connectivity (Moderate Density, High Mobility)

6. CONCLUSIONS

The Random Waypoint mobility model has the highest probability of *k*-Connectivity when compared to City Section and Manhattan models. At low values of *k* (*k* = 1, 2, and 3), City Section model has better probability of *k*-Connectivity than the Manhattan model for almost scenarios. At medium (*k* = 4, 5, 6, and 7) and high (*k* = 8, 9, and 10) values of *k*, the Manhattan model has the highest probability of *k*-Connectivity at lower densities, while the City Section model has the highest probability of *k*-Connectivity at moderate and higher densities. For each mobility model, with increase in density, the variation in the probability of *k*-Connectivity decreases and the absolute mean value of the *k*-Connectivity increases. For a given density, velocity and *k*, the Random Waypoint mobility model has less variation in *k*-Connectivity compared to the City Section and Manhattan mobility models.

REFERENCES

[1] N. Chatterjee, A. Potluri and A. Negi, "A Self-Organizing Approach to MANET Clustering," Vol. 4882, *Lecture Notes in Computer Science*, pp. 73-78, November 2007.

[2] M. Rudack, M. Meincke, K. Jobmann and M. Lott, "On Traffic Dynamical Aspects Inter-vehicle Communication (IVC)," *Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference* (*VTC03 Spring*), April 2003.

[3] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication and Mobile Computing*, Vol. 2, No. 5, pp. 483-502, September 2002.

[4] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, S. Suri, "Towards Realistic Mobility Models For Mobile Ad hoc Networks," *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, 2003, San Diego, CA, USA.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Single-Source Shortest Paths," *Introduction to Algorithms*, 2nd Edition, MIT Press, 2001.