Mobile Network Discrete-Event Simulation

Semyon Fishman
sf69@drexel.edu

April 5, 2010

1 Introduction

We present a discrete-event simulation process for modeling systems of mobile limited range transmitter-receiver radio devices. The goal is to simulate the physical movement of these devices, while simultaneously maintaining a graph of the communication network between them. Such a simulation would aid in developing statistical relationships between the movement of radio devices and the network graph they form.

2 System Definition

Vertices of directed graph $G(V, E)$ are the mobile radio devices, and edges are the unidirectional communication links between them. The graph is a variable entity that mutates throughout the execution of the simulation by adding and removing edges. The addition and removal of edges is determined by the Euclidean distance between the vertices, and their transmitter-receiver ranges. The movement of vertices is dictated by a mobility model, which may or may not be unique to a vertex. A mobility model works by assigning the next waypoint to a vertex when it arrives at its present waypoint.

2.1 Vertex Movement

The displacement of a vertex in $m$-dimensional Euclidean space is defined by $m$ time-dependent $n$-degree polynomial equations $s_1(t), s_2(t), \ldots, s_m(t)$, one for each dimension, of the form,

$$s(t) = c_0 + c_1(t - T_S) + c_2(t - T_S)^2 + \cdots + c_n(t - T_S)^n$$

for $T_S \leq t \leq T_E$ (1)
where $T_S$ is the start timestamp, or the time when the vertex reached its previous waypoint; and $T_E$ is the end timestamp, or the time when the vertex will reach its present waypoint.  

Vertices continually move from one waypoint to the next. Upon arrival of a vertex at a waypoint, $T_E - T_S$ time units after departing from its previous waypoint, new displacement equations $s'_1(t), s'_2(t), \ldots, s'_m(t)$ and timestamps $T'_S$ and $T'_E$ are set. The new displacement equations must start where the previous equations ended, i.e.,

$$s'(T_E) = s(T_E) \quad \text{must hold} \quad (2)$$

That is, the constant term of $s'(t)$ is set to $s(T_E)$, and the new start timestamp is set to the previous end timestamp,

$$c'_0 \leftarrow s(T_E)$$

$$T'_S \leftarrow T_E \quad (3)$$

The rest of the displacement coefficients $c'_1, c'_2, \ldots, c'_n$ along with the new end timestamp $T'_E$ are arbitrarily decided by the vertex’s mobility pattern. $T'_E$ must be greater than $T'_S$.  

The Euclidean distance $d_{\alpha,\beta}(t)$ between two vertices $\alpha$ and $\beta$ is thus

$$d_{\alpha,\beta}(t) = d_{\beta,\alpha}(t) = \sqrt{[s_{1\alpha}(t) - s_{1\beta}(t)]^2 + \cdots + [s_{n\alpha}(t) - s_{n\beta}(t)]^2} \quad (4)$$

for $\max(T_{S\alpha}, T_{S\beta}) \leq t \leq \min(T_{E\alpha}, T_{E\beta})$  

The Greek letter subscript indicates which vertex the variable or equation belongs to.  

The initial displacement of vertices, that is the displacement at the start of the simulation, is chosen arbitrarily. The initial timestamp $T_I$, commonly chosen to be 0, is the time when the simulation starts. The first start timestamp must be equal to the initial time, and the first end timestamp must be greater, i.e.

$$T_E > T_S = T_I \quad \text{at} \quad t = T_I \quad (5)$$

---

\[1\] Equation 1 can be rewritten in standard polynomial form as

$$s(t) = \hat{c}_0 + \hat{c}_1 \cdot t + \hat{c}_2 \cdot t^2 + \cdots + \hat{c}_n \cdot t^n$$

where $\hat{c}_d = \sum_{i=0}^{n-d} (-1)^i (\hat{c}_{i+d} \cdot T_S^i)$
2.2 Communication Radii

Each vertex has a transmitter radius $T$ and a receiver radius $R$. The transmitter radius indicates the maximal distance from which the vertex’s signal can be received; that is how ‘loud’ the vertex speaks. While the receiver radius indicates the maximal distance from which the vertex can receive signal; that is how well can the vertex ‘hear’.

More formally we define this rule as follows: at any time $t$ a signal can be transmitted from vertex $\alpha$ to vertex $\beta$, if and only if

$$d_{\alpha,\beta}(t) \leq \max(T_{\alpha}, R_{\beta}) \quad (6)$$

The variable edge set $E$ of $G$ at any time $t$ is thus,

$$E = \{(\alpha, \beta) : d_{\alpha,\beta}(t) \leq \max(T_{\alpha}, R_{\beta}) \land \forall \alpha, \beta \in V\} \quad (7)$$

3 Discrete-Time Calculations

Given displacement equations for vertices $\alpha$ and $\beta$ we can determine the discrete times, between $\max(T_{S\alpha}, T_{S\beta})$ and $\min(T_{E\alpha}, T_{E\beta})$, when edge $(\alpha, \beta)$ is added or removed.

We set $d_{\alpha,\beta}(t) = \max(T_{\alpha}, R_{\beta})$, rearrange Equation 4 into a polynomial,

$$[s_{1\alpha}(t) - s_{1\beta}(t)]^2 + \cdots + [s_{m\alpha}(t) - s_{m\beta}(t)]^2 - \max(T_1, R_2)^2 = 0 \quad (8)$$

and solve for the set of roots $\tau$ between $\max(T_{S\alpha}, T_{S\beta})$ and $\min(T_{E\alpha}, T_{E\beta})$ of Equation 8. Such that,

$$\tau = \{\tau : d_{\alpha,\beta}(\tau) = \max(T_{\alpha}, R_{\beta}) \land \max(T_{S\alpha}, T_{S\beta}) \leq \tau \leq \min(T_{E\alpha}, T_{E\beta})\}$$

$$0 \leq |\tau| \leq 2n$$

To determine if each value $\tau$ is an edge-addition or edge-removal time we check if

- $\max(T_{S\alpha}, T_{S\beta}) = \tau < \min(T_{E\alpha}, T_{E\beta}),$
- $\max(T_{S\alpha}, T_{S\beta}) < \tau < \min(T_{E\alpha}, T_{E\beta}),$
- $\max(T_{S\alpha}, T_{S\beta}) < \tau = \min(T_{E\alpha}, T_{E\beta}),$ or
- $\max(T_{S\alpha}, T_{S\beta}) = \tau = \min(T_{E\alpha}, T_{E\beta})$

and if the distance at $\tau$ is
• increasing \( \frac{d}{d\tau}d_{\alpha,\beta}(\tau) > 0 \),
• constant \( \frac{d}{d\tau}d_{\alpha,\beta}(\tau) = 0 \), or
• decreasing \( \frac{d}{d\tau}d_{\alpha,\beta}(\tau) < 0 \).

The appropriate action for each of the twelve possible cases is summarized in Table 1. Remove means that the edge \((\alpha \beta)\) is removed at time \(\tau\), add means that an edge \((\alpha \beta)\) is added at time \(\tau\), and ignore indicates no action, that is we ignore the particular \(\tau\).

Cases 4 and 6 cover the most common scenarios when vertex \(\beta\) leaves and enters the radius \(\max(T_\alpha, R_\beta)\); while cases 1, 2, 3, 5, 7, 8, and 9 cover the less common scenarios when vertex \(\beta\) stops, starts, or passes on the boundary of radius \(\max(T_\alpha, R_\beta)\). Cases 10, 11, and 12 take care of the situation when \(T_{Sa} = T_{E\beta}\) or \(T_{S\beta} = T_{E\alpha}\), that is \(\alpha\) or \(\beta\) is about to acquire its next waypoint and time \(\tau\) has already been simulated, thus we ignore it.

Interestingly enough, if we were to change the edge existence rule (Equation 6) to

\[ d_{\alpha,\beta}(t) < \max(T_\alpha, R_\beta) \]  \hspace{1cm} (9)

Table 1 would become,

<table>
<thead>
<tr>
<th>(\max(T_{Sa}, T_{S\beta}))</th>
<th>Increasing</th>
<th>Constant</th>
<th>Decreasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau &lt; \min(T_{E\alpha}, T_{E\beta}))</td>
<td>1. Ignore</td>
<td>2. Ignore</td>
<td>3. Add</td>
</tr>
<tr>
<td>(\tau &lt; \min(T_{E\alpha}, T_{E\beta}))</td>
<td>4. Remove</td>
<td>5. Ignore</td>
<td>6. Add</td>
</tr>
<tr>
<td>(\tau = \min(T_{E\alpha}, T_{E\beta}))</td>
<td>7. Remove</td>
<td>8. Ignore</td>
<td>9. Ignore</td>
</tr>
<tr>
<td>(\tau = \min(T_{E\alpha}, T_{E\beta}))</td>
<td>10. Ignore</td>
<td>11. Ignore</td>
<td>12. Ignore</td>
</tr>
</tbody>
</table>

Table 2: Action taken for each of the twelve possible cases of \(\tau\) for Equation 9

This process is easily extended to the entire graph \(G(V, E)\) by repeating the process for every pair \(\alpha, \beta \in V\). Thus we can determine the complete
structure of the graph for time,

\[
\max(T_{S\alpha}, T_{S\beta}, \ldots, T_{S\omega}) \leq t \leq \min(T_{E\alpha}, T_{E\beta}, \ldots, T_{E\omega})
\]  \hspace{1cm} (10)

where \( V = \{\alpha, \beta, \ldots, \omega\} \).

Assuming that it takes constant time to run the process for one pair of vertices, we conclude that each vertex waypoint assignment takes \( O(|V|) \) time to complete. \( ^2 \)

\section{Simulation Process}

Using the process described in Section 3 we can implement a software discrete-event simulator to model the system defined in Section 2. Three types of events are required for this simulation:

1. Vertex waypoint arrival events;
2. Edge-addition events; and
3. Edge-removal events

Events are scheduled to execute at some future time \( \tau > t \), and take instant time to occur.

A \textit{vertex waypoint arrival event} is scheduled to occur when a vertex arrives at its present waypoint. The event sets the vertex’s next waypoint (as described in Section 2.1); and schedules the appropriate edge-addition and edge-removal events between the given vertex and all other vertices (as described in Section 3). This event type both schedules future events and changes the state of the system. An \textit{edge-addition event} or \textit{edge-removal event} is scheduled, (by a vertex waypoint arrival event), to occur when an edge is added or removed respectively, also changing the state of the system.

Note that this simulation process may never terminate by itself so an external termination factor must be used. For example, we may declare a variable \( T_{\text{run}} \) such that all events scheduled to occur after it are ignored, or run the simulation process for a fixed amount of events.

\( ^2 \)This assumption will likely not hold if our displacement equations are of 3\textsuperscript{rd} or higher degrees, which will lead us to solve Equation 8 of degree six or higher for which general formulas do not exist and approximation methods that may not take constant time to converge may be used.