ENHANCING THE SPECIFICATION OF NODE MOBILITY IN MOBILE AD-HOC NETWORKS USING THE MOTION PLANNING FRAMEWORK

David J. Claypool, Justin Dean
Information Technology Division
Naval Research Lab, Washington, DC, USA
Email: {david.claypool,justin.dean}@nrl.navy.mil

Abstract

When modeling and simulating Mobile Ad-hoc Networks (MANETs), the importance of the represented mobility of the nodes has too often been downplayed: oversimplified, contrived to relatively simplistic random motion, and limited to a small set of notional motion patterns. Even when realistic motion is used, the scenarios are often highly static and deterministic. As such, there exists a need to be able to clearly define the motion of nodes in more complex patterns: both in the context of conceptual testing patterns, and also in realistic operational scenarios. This work introduces the features and usability of an ongoing development effort termed the Motion Planning Framework created by the US Naval Research Laboratory. The framework enables modelers, independent of a specific simulation environment, to succinctly express the motion of nodes in a scenario. The extensible language permits users to quickly and universally create arbitrarily complex node movements representing scenarios that more aptly represent real world behavior or more appropriately exercise a given system under test. This work introduces our approach towards creating this framework and gives several examples of how the language can be used.

Introduction

Mobile Ad-Hoc Networks (MANETs) are an extensively studied domain, and at the foundation of this area of study is the idea that the nodes involved are moving in some way through space: creating changes in the network topology. This concept is simple, but the implementation thus far has been fairly limited: those studying MANETs resort to random walks, movements about a grid, pre-recorded movement patterns, and other built-in movement patterns, often with all nodes in the group having identical movement characteristics. There has been extensive works, and resulting findings [1] in looking at these sorts of random movements, but work looking at configuring combinations of movements in complex scenarios is limited.

This work was spawned from development work of the Enhanced Mobile Ad Hoc Network Emulator (EMANE) [2] software developed by the Naval Research Lab but is designed to fulfill the motion planning needs of any mobile network simulation environment. Key to current EMANE development is the evolution of the deployment tool chain - the separation of concerns of various simulation aspects, and the automatic configuration and deployment of the test once the specifications for a given run are made. Shown in Figure 1 is a simplified process diagram by which EMANE deployments are designed to be made: it gives a context specifically for the approach taken by the motion planning framework developed.

Figure 1. Simplified EMANE Deployment Process

In an actual deployment of multiple wireless mobile networks, there are numerous nodes moving in differing ways: and in particular in using the EMANE environment we saw the need to create this Motion Planning Framework (MPF) that permits planners of mobile network simulations to define in simple terms the movement of nodes in more complex ways than currently exist. The MPF is comprised of the XML based Motion Plan Language (MPL), a preliminary MPL compiler, and a visualization engine that permits MPL instances to be viewed. This work presents several examples of these scenarios as an illustrative way of seeing the applicability and scalability of the language. The approach section diagrams and details this process and the resulting output files.

Applicability and Purpose

The goals of the functional motion planning framework are numerous. One of the most important requirements is that such a language should support two distinct types of definitions: logical motion patterns and operational scenarios. More often than not, these two types of scenarios are blurred in simulation studies, and so a quick overview of the purpose of each is useful, in the context of driving the design of the MPF.

Logical motions are not particularly meant to replicate any particular realistic operation scenario: but instead they move nodes in ways that can deterministically push the networks under simulation into configurations that explicitly test specific boundary conditions or other carefully controlled layouts. For example, such scenarios of 50 nodes randomly moving within a square region may not represent anything realistic, but it provides the baseline dynamic connectivity experienced in a mobile environment, particularly when parameterized and examined at varying levels of node speed. The MPF allows users to construct logical scenarios which greatly transcend those of simple random motion. In the
examples section of this work are illustrations of how this language permits much more complex and potentially much more useful experimental motion scenarios. These scenarios tend to be driven by expressions; nodes positioned and parameterized according to equations in elaborate testing patterns.

The second type of motion plans the MPF strives to enable are operational scenarios, which are designed to explore realistic operating environments. In these scenarios, nodes move according to realistic conditions: for example, a collection of vehicles may move from a base at a specific time as a convoy, these types of scenarios mimic real world “war gaming” scenarios, and mobility patterns have been studied before to generate these sorts of scenarios [3]. Giving each node a set of unique waypoints that coincidently are close to others in the convoy is neither intuitive or reusable. The MPF permits numerous types of operational behaviors such as formations and loitering behaviors, to define in familiar terms the types of motions desired. The MPL supports scripting of motions in a logical mission plan to allow scenarios to play out in phases and for nodes to move according to a given course of action, which may involve numerous types of motions, each playing a part at a given time. These types of motion plans emphasize scripting and sequencing of motion types, particularly for mission phases and realistic operations. These two types of motion plans, while quite different in scope, are too often simplified into a variant of the random walk or a simple waypoint model.

Usability and applicability of the language is another goal the MPF strives for: the XML-based language is human-readable, and designed to be succinct yet powerful. Additionally, the opportunity to create a graphical planning environment independent of other ongoing work on the MPF exists and could greatly enhance the accessibility of the MPL for those who do not have an understanding of XML-based languages. The MPF is designed to be simulation environment agnostic - it is not until the user needs a specific format that the generator will compile the motion to a given file form.

Scalability is highly important to the MPF, and by adding the ability to iterate and repeat node placements with expression parsing for defining individual parameters for each node’s qualities, the user of the MPF can easily lay down hundreds of nodes of various configurations with small segments of XML. Additionally, the positioning environment can be specified seamlessly in Cartesian or longitude/latitude pairs for motion patterns that extend only several meters to a global scenario with nodes moving locally on several continents.

Extensibility is yet another goal: the key to this extensibility is that the motion patterns themselves are referenced by the language, but could be expanded at any time: the high level “motion” XML element could, at a later time, reference a new “type” of motion pattern. There are an incredible number of motion patterns defined in previous works, such as even driven from the movement of players in a first-person shooter [4]. There exists a clearly defined interface for creating and integrating new motion pattern types. Additionally supporting extensibility, libraries of common motion types and even sets of regions and points can be created as a library and reused in other scenarios, imported as a resource for elements within.

Finally, the approach taken towards this language supports stochastic scenarios, both in the logical and operational domains. Such randomization and associated parameterization of the defined scenarios greatly enhances one’s ability to truly test a network in varying environments. This concept is particularly important in the operational category. The language allows the user to quickly define scenarios that are randomly generated, but operationally consistent with each run of the motion plan. The examples section illustrates how generative stochastic operational scenario would work and shows just how much such scenarios can vary from one compilation to another.

**Existing Work**

There are numerous examples of excellent mobility patterns that have been defined to replicate operational conditions: on their own, they are incredibly useful within their domain. This language intends to support the ability to call upon these highly specialized motion patterns – and even supply them with pattern-specific parameter values, as part of an extensible framework approach. This problem has been recognized, and studied in the domain of specific focus areas, such as urban outdoor movement [5]. It has been shown that the realism of mobility patterns has a huge impact on the performance of wireless networks under simulation testing, suggesting that motion patterns are not emphasized enough when conducting tests [6].

Additionally, the concept of nodes moving in relation to other nodes is another well studied area [7] and was addressed in this framework as well. Such dependence is an incredibly useful and powerful concept to include in the framework in form of loitering, formation following, and swarming behaviors. Perhaps most significantly, the work outlined in [8] provides a similar concept and user interface to what is outlined here, without as many motion patterns defined and without the context of the overall tool chain. The proposed unified framework presented in [9], in particular describing the data structures of mobile node positioning and velocity, contains numerous similarities to the under-the-hood processing of the MPF presented here.

**Approach**

The MPF is shown in the process diagram in Figure 2 and is simulation framework independent: at the highest level it supplies a front-end tool set for effectively describing the motion of nodes which can then be compiled into a time-stamped list of locations in the world. The language itself drives the xml syntax that can clearly and succinctly define the motion of nodes, which reference motion patterns containing the logic necessary to determine position based on the given parameters of a given motion pattern type. The MPF compiler processes the XML, making instances of the motion patterns as motion instances, and calling upon the logic embedded in the pattern definition to determine the position of nodes. A common underlying motion plan xml file can create, through a compiler for a given environment, an instance for the given simulator.
The motion plan XML files makes reference to motion patterns, which are abstract ways in which a node might move and are defined as relative position as a function of time. These patterns may also be a function of the position of other nodes, and very complex dependency relationships can arise in such cases. Motion patterns can be combined in various ways, and sequenced, as described later. A given motion plan references such patterns and creates parameterized instances of them to create a source set of motion instances. At this point, the positions are not yet determined, but the initial conditions and specific parameters are set. These motion instances can then be collectively compiled into a deterministic motion record. At this point, each node has a well-defined position at any time.

This motion planning toolset is comprised of a well-defined XML schema, tools for processing such a schema into motion formats applicable for various simulations, and tools for assisting in visualizing such XML models. The first parser for developing the common motion instances and records is currently written in the C# language. Once in this form, it is easily translated into simulation-framework specific formats, such as those for NS2 and NRL’s own in-house visualization tools, SDT3d.

**Motion Concepts**

To create an initial environment for using the MPF, an initial set of motion patterns were implemented to provide a foundation to use the language. We outline these patterns here to illustrate the ways in which they can be used currently. Minimally, we include simple notions of a node wandering about a region: either through the random waypoint model or the random direction model. While simplistic, we can combine this type of motion to represent more complex behaviors, and a fair amount of work has been performed to make such random walks more realistic [10]. Another simplistic motion pattern already is included in the concept of following a path or a loop, for a given number of iterations or indefinitely. Likewise, simple waypoint specification is also available, both as a go-to point and as a direction for a specified for an indefinite amount of time.

Motion patterns can be independent from one another, like those mentioned already, but can also be dependent on the position of other nodes in the world: node can follow another node in formation, or follow another node by loitering around it in a defined pattern. The loitering of a node about another node is defined by a path object, which is used as the time-varying relative offset of a node from another node. This allows flexibility in the type of loitering performed: figure-8’s, for example, for aircraft loitering above Naval vessels, versus a UAV performing simple circles around a ground node. The mobility model properties presented in [11] is highly useful in understanding the kinds of properties various models can exhibit.

Extending this dependency further, nodes can have positions that are mutually dependent: A more unique motion pattern included is a scattering/clustering pattern, in which nodes move in close proximity to them, and closer to nodes that are far away. Figure 3 illustrates this particular motion pattern, and it can be quite useful, particularly at slow speeds, in generating networks that are varying in crowdedness over time. It is interesting to note that the orderly crystalline pattern of nodes is the result of relatively simplistic rules.

The concept of roadways has already been included and in support of this data, a Road Wandering mobility pattern is already included: nodes will wander randomly, but remain on roadways, picking from available routes at random when they hit an intersection. This motion pattern has been included as it is used often and certainly to an even greater level of complexity [12], and allows a familiar baseline for users to create similar scenarios to what they use already.

The most obvious form of combined motion patterns being the concept of sequencing motion patterns. Sequencing is highly parameterized, and even includes the ability to automatically include or exclude transition time from the desired duration: A node can be configured to loiter above a node for 60 seconds, then another node for 30 seconds. The 30 second timer can be configured to not start until the node has successfully reached a loitering position: greatly reducing the complexity of calculations from the end user perspective. With relatively simple lines of XML, a user can specify a node visit a list of nodes for one minute each, taking the time required to perform moving between nodes as effectively as possible within the physical limits of the node. Sequencing is the most common motion type used in operational scenarios, because it allows events to be easily scripted.
Stacking is another compound motion, and allows users to simply perform vector addition on two or more underlying types: for example, a node may wander above a region that is in of itself moving relative to another node: aircraft swarming above a carrier group in a random, but realistic pattern. The types of motions explained here have already been implemented, but the language is explicitly designed to be extensible to add other motion pattern types. Captured motion data can also be implemented. Any given instance of the MPL can include any number of instances of these motion patterns, which can even depend on each other, allowing for extremely complex motion scenarios to be developed.

Examples

The first scenario presented represents a distribution of mobile nodes that can explicitly test a routing algorithms' ability to effectively handle varying levels of movement within a given configuration: a topic of interest right now with various hybrid routing algorithms. Termed “The Gauntlet”, the field of interest is an elongated rectangle populated with nodes which each randomly wander within a square contained in the rectangle, with nodes and associated wandering regions distributed evenly across the length of the rectangle. Nodes near the center move more rapidly than those near the edge, and finally, those on the two ends do not move at all. Two nodes are affixed to the ends of the rectangle. Figure 4 illustrates this scenario, and as shown, the nodes near the edges of the rectangle are moving more slowly than those moving rapidly in the center.

![Figure 4: The Gauntlet, at three different times](image)

Using the language proposed here, this scenario is very easy to write. Part of the framework is an entire specification and training material for this language, so this is supplied and intended for those familiar with XML and could potentially deduce it's meaning. As shown in Code Section 1, the language supports iteration and variables, as it is in a way a functional programming language in itself written in XML:

```xml
<motionPlan>
  <scopeBounds
    X="-30" Y="-30"
    width="860" height="860" />
  <variable var="MAXSPEED" value="5.0" />
  <variable var="NODECOUNT" value="30" />
  <path name="MainRegion">
    <parameter name="show">yes</parameter>
    <point>0,0,0</point>
    <point>800,0,0</point>
    <point>800,200,0</point>
    <point>0,200,0</point>
  </path>
  <Node name="A">
    <motion type="stationary">
      <point>0,100,0</point>
    </motion>
  </Node>
  <Node name="B">
    <motion type="stationary">
      <point>800,100,0</point>
    </motion>
  </Node>
  <repeat var="N" end="$NODECOUNT-2">
    <variable var="SPEEDFACTOR" value="1.0-2*Math.Abs(0.5-$N$/28.0)" />
    <path name="Region$N" >
      <parameter name="show">no</parameter>
      <point>1.0*$N$/($NODECOUNT-3)*(800-200),0,0</point>
      <point>1.0*$N$/($NODECOUNT-3)*(800-200)+200,0,0</point>
      <point>1.0*$N$/($NODECOUNT-3)*(800-200)+200,200,0</point>
      <point>1.0*$N$/($NODECOUNT-3)*800,200,0</point>
    </path>
    <Node>
      <motion type="wanderVector">
        <path ref="Region$N" />
        <parameter name="stepRate">0.25</parameter>
        <parameter name="twistRate">0.1</parameter>
        <parameter name="speed">$MAXSPEED*$SPEEDFACTOR</parameter>
      </motion>
    </Node>
  </repeat>
</motionPlan>
```

**Code Section 1: Complete Code for “The Gauntlet”**

The next scenario presented without associated code, is termed “The Uncertain Bridge, and is related to the previous scenario. In this example, multiple routes exist for traffic to reach nodes at each end of the field. Nodes in the center region move back and forth, randomly breaking links across the field. This scenario tests the ability of algorithms to handle multiple intermittent routing options. The entire scenario could be parameterized for how often the various links break, and how many of the links are active at any given time. Within a given chosen set of parameters, each compiled instance is randomly generated - so the connections that break will be different each compiled motion plan. Shown in Figure 5 are two timestamps of a single instance of this scenario. Because the scenario is highly parameterized, networks of greater size are easily created with the reassignment of a single variable value.

![Figure 5: The Uncertain Bridge, at two different times](image)
result, when examining two nodes at opposite ends of the outer ring, reveals there is an unstable but low-hop route cutting across the middle of the rings, and a highly stable route going around the rings. Under various configurations, the number of hops around the ring can be double that of the number of hops across the ring. It is an interesting scenario to evaluate how a routing algorithm will attempt to balance these two routing options.

Figure 6: Carousel Scenario

Operationally, we will present two scenarios for illustrating the language. The base exploration scenario presents 20 nodes, 8 of which are land based and remain in a notional base camp, randomly wandering within the bounds of the base, 8 additional land-based nodes begin in the base just like the other 8, but after 60 seconds, join a leader node in an expedition out of the base, in formation, to three randomly selected waypoints. These nodes are supported by 3 loitering UAVs overhead, one of which loiters above the moving leader node. Figures 7 through 10 illustrate this sequence of events which plays out in this scenario.

Figure 7: Base Operations: Initial Conditions

Figure 8: Base Operations: Nodes Leaving Base

Figure 9: Base Operations: Nodes Patrolling Points

Figure 10: Base Operations: Nodes Returning

Finally, we present a large stochastic naval operations scenario, representing a notional island and involving nodes spread out over a range of 100s of miles across. The scenario is represented with over 200 nodes, each with various motion patterns, and defined with a high-degree of random variation. In the scenario, half of the land nodes are mobile, a subset of them have air to ground radios, there are 4 clusters of ships moving in the water around the island, along with a collection of independent ships spread about the rest of the region.

All of these nodes are flown over by a collection of lingering aircraft. Each group of nodes have different motion characteristics, but all can be expressed in several hundred lines of XML. Importantly, however, is that this random variation remains operationally consistent. The groups of ships will always patrol a logical area, the land units will always be placed on land, will always roam only on the land, and planes will linger only above the operating region. However, as each of the 3 instances shown in Figure 11 illustrate, each time, from a network perspective, the scenario exhibits much different connectivity characteristics.

Figure 11: Operationally consistent, randomly generated instances of the Maritime Operations Scenario, each shown at initial conditions

The Way Forward

With the language elements defined and the initial set of motion patterns implemented, the obvious initial next steps are to spread the adoption of the language and enhance its usage, as well as expand the number of applicable platforms. Already the instantiated motion records can be exported to common formats such as the MITRE mobility format, ns2 tcl scripts, and those used within the EMANE and SDT project at NRL. Once the motion plan is compiled into a deterministic set of locations it is relatively easy to export to any arbitrary motion scripting format required for a given simulation: All compiled motion plans can be distilled down to a list of time-stamped locations.

More significantly and long term, we envision fundamental progress to be made in implementing this language as a dynamic part of the simulation. Two areas of this expansion include externally-driven motion events, and inline motion even processing during simulation: Event-driven motion triggering involved allowing the motion plans to be triggered by external events. For example, a node may move to a location after a specific type of packet is received over the network, or a node may move to the perceived location of a given transmission to gain better reception. These scenarios involve allowing triggers to be defined that are thrown outside the domain of the motion patterns themselves. The capability
exists to define this in the language, but the integration with specifications other than specific modeling environments.

Inline motion processing is an advanced level of integration, and involves allowing the compilation of the motion patterns to happen at execution time, instead of having the motion patterns compiled all before running the simulation. Analogous to adding a motion processing federate to a multi-modeling capable simulation, this level of integration would be highly depended on the simulation environment targeted. Allowing for such inline processing would open up new motion types and network dynamics.

Once inline motion processing is added for a given simulation environment, it will be possible to create human in the loop scenarios in which the motion is controlled not only by a human, but limited or otherwise influenced by the underlying motion patterns as well. For example, a human in the loop pattern may be constrained according to the physical limitations of the represented vehicle moving.

The Motion Planning Framework is still being developed and evolved, but we have the start of a usable language and a set of tools to begin taking this framework further and developing its usage in real world simulation efforts. We believe a simulation-independent, easy to understand method of defining motion of nodes in mobile networking environments is key to bringing in more advanced scenarios, particularly in the operational domain. Such enhancement will result in more effective simulation efforts, and more realism to the movement of nodes. The specification of the MPL and training materials for the overarching MPF for using the framework are/will be posted on NRL’s public website.

References


