CHAPTER 6
SIMULATION AND PERFORMANCE METRICS

6.1 ABOUT NS2

Network Simulator (Version 2), widely known as NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Both wired and wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be simulated using NS2. In general, NS2 provides users with a method of specifying such network protocols and simulating their corresponding behaviors.

Ever since its birth in 1989, NS2 has gained constant popularity in the networking research community due to its flexibility and modular nature. Several revolutions and revisions have marked the growing maturity of the tool, thanks to the some of the key players in this field. Among these are the University of California and Cornell University who developed the REAL network simulator 1, the foundation of NS is on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) has been supporting the development of NS through the Virtual InterNetwork Testbed (VINT) project. Currently, the National Science Foundation (NSF) has joined the ride in development. In addition to all these, a group of researchers and developers in the community are constantly working to continue to ensure that NS2 strong and versatile.

6.2 ARCHITECTURE OF NS2

Fig 6.1, shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users feed the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects and scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g., _o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. The member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. It is important to the readers are to learn about C++ and OTcl languages to get a better understanding of these architecture.

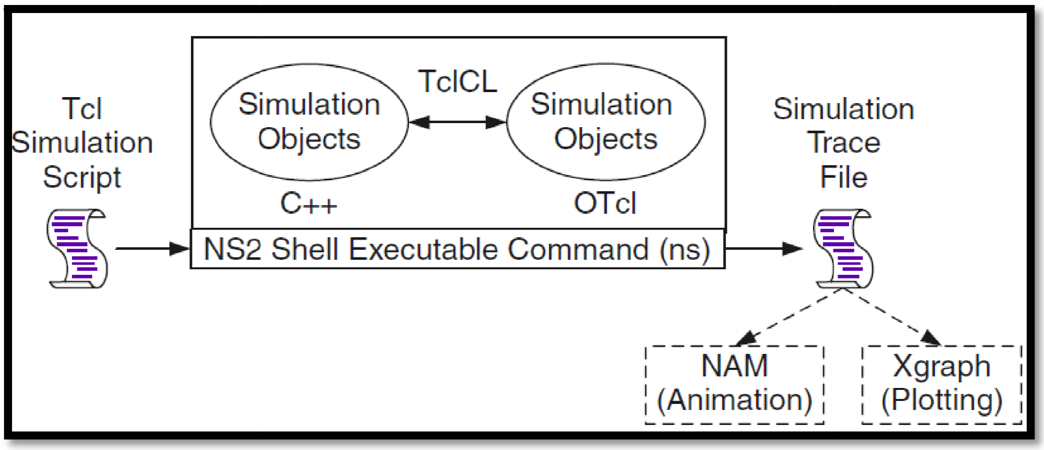

Fig 6.1: Basic architecture of NS.

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects to be insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results both graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

6.3 DISCRETE-EVENT SIMULATION

NS2 is a discrete-event simulator, where actions are associated with events rather than time. An event in a discrete-event simulator consists of execution time, a set of actions, and a reference to the next event (Fig 6.2). These events connect to each other and form a chain of events on the simulation timeline. Unlike a time-driven simulator, in an event-driven simulator, the time between a pair of events does not need to be constant. When the simulation starts, events in the chain are executed from left to right (i.e. chronologically). The next section, discusses the simulation concept of NS2.

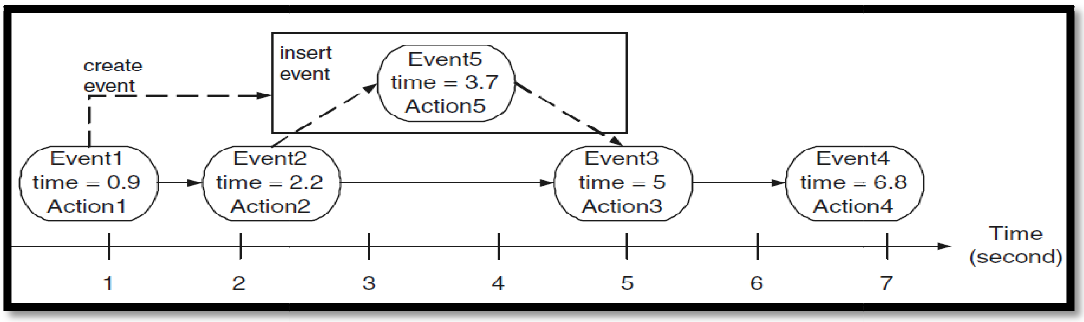

Fig 6.2: Discrete-Event Simulation.

Fig 6.2 demonstrates a sample chain of events in a discrete-event simulation. It can be observed that event contains execution time and a reference to the next event. In this figure, Event1 creates and inserts Event5 after Event2 (the execution time of Event 5 is at 3.7 second).

6.3.1 NS2 Simulation Concept

NS2 simulation consists of two major phases.

Phase I: Network Configuration Phase

In this phase, NS2 constructs a network and sets up an initial chain of events. The initial chain of events consists of events which are scheduled to occur at certain times (e.g., start FTP (File Transfer Protocol) traffic at 1 second.). These events are called at-events. This phase corresponds to every line in a Tcl simulation script before executing instproc run{} of the Simulator object.

Phase II: Simulation Phase

This part corresponds to a single line, which invokes instproc Simulator::run {}. Ironically, this single line contributes to most of the simulation (e.g., 99%). In this part, NS2 moves along the chain of events and executes each event chronologically. Here, the instproc Simulator::run{} starts the simulation by dispatching the first event in the chain of events. In NS2, “dispatching an event” or “firing an event” means “taking actions corresponding to that event”. An action, for example, refers to starting

FTP traffic or creating another event and inserting the created event into the chain of events. In Fig 6.2, at 0.9 s, Event1 creates Event5, which will be dispatched at 3.7 s, and inserts Event5 after Event2. After dispatching an event, NS2 moves down the chain and dispatches the next event. This process repeats until the last event corresponding to instproc halt{} of OTcl class Simulator is dispatched, signifying the end of simulation.

6.4 NS2 COMPONENTS

A network object is one of the main NS2 components, which is responsible for packet forwarding. NS2 implements network objects using the polymorphism concept in Object-Oriented Programming (OOP). Polymorphism allows network objects to take different actions ways under different contexts. For example, a Connector object immediately passes the received packet to the next network object, while a Queue1 object enques the received packets and forwards only the head of the line packet.

Based on the functionality, NS2 modules (or objects) can be classified into following four types:

- ✓ **Network objects** are responsible for sending, receiving, creating, and destroying packet-related objects. Since these objects are those derived from class NsObject, hereforth, they will be referred to as NsObjects.
- ✓ **Packet-related objects** are various types of packets which are passed around a network.
- ✓ **Simulation-related objects** control simulation timing, and supervise the entire simulation. Some examples of simulation-related objects are events, handlers, the Scheduler, and the Simulator.

- ✓ **Helper objects** do not explicitly participate in packet forwarding. However, they implicitly help to complete the simulation. For example, a routing module calculates routes from a source to a destination, while network address identifies each of the network objects.

6.5 NS2 C++ Class Hierarchy

This section gives an overview of C++ class hierarchies. The entire hierarchy consists of over 100 C++ classes and struct data types. Here, only part of the hierarchy is discussed (in Fig 6.3).

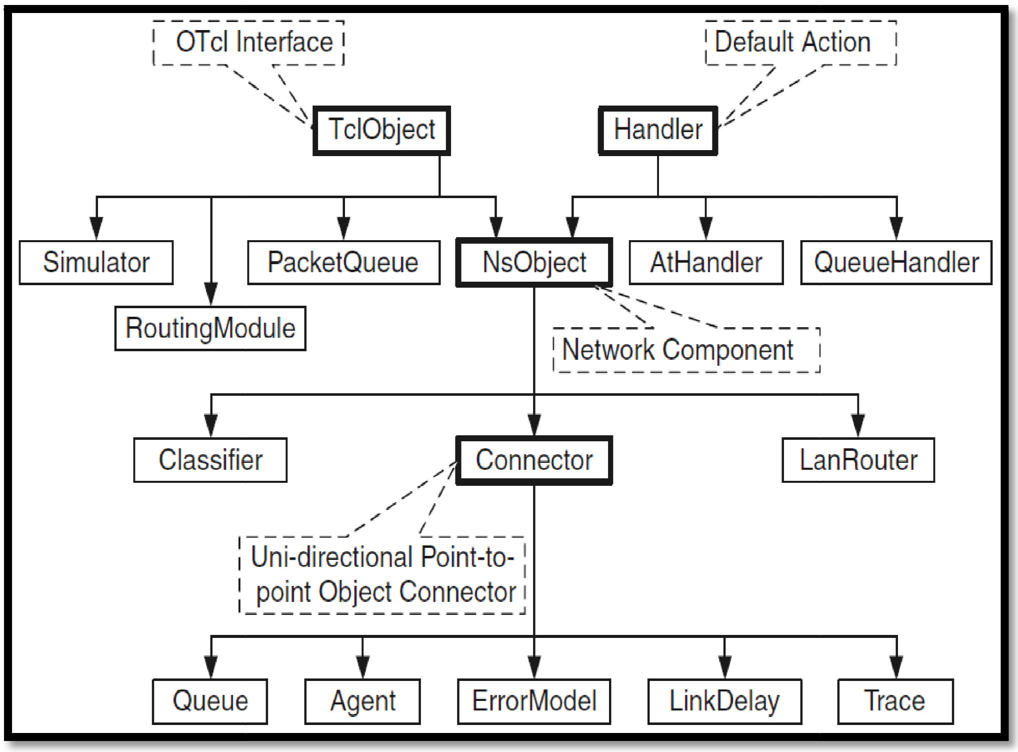

Fig 6.3: A Part of NS2 C++ Class Hierarchy.

All classes deriving from class TclObject form the compiled hierarchy. Classes in this hierarchy can be accessed from the OTcl domain. For example, they can be created by the global OTcl procedure "new{...}".

Classes derived directly from class TclObject include network classes (e.g., NsObject), packet-related classes (e.g., PacketQueue), simulation-related classes (e.g., Scheduler), and helper classes (e.g., Routing- Module). Again, those classes which do not need OTcl counterparts (e.g., classes derived from class Handler) form their own standalone hierarchies. These hierarchies are neither a part of the compiled hierarchy, nor that of the interpreted hierarchy.

Class Handler specifies an action associated with an event. Again, class Handler contains a pure virtual function handle(e). Therefore, its derived classes are responsible for providing the implementation of function handle(e).

Program for Function handle of class QueueHandler.

```
//~/ns/queue/queue.cc

void QueueHandler::handle(Event*)
{
queue_.resume();
}
```

OTcl and C++ classes Simulator are the main classes which supervise the entire simulation. Like the Scheduler object, there can be only one Simulator object throughout a simulation. This object will hence foth will be referred to as the Simulator hereafter. The Simulator contains two types of key components: simulation objects and information-storing objects. A Simulation object (e.g., the Scheduler) is the key component which it derives during the Network Configuration Phase which will be used in the Simulation Phase [Schildt., 2002].

Information-storing objects (e.g., the reference to created nodes) contain information which is shared among several objects. For example, NS2 needs to know all the created nodes and links in order to construct a routing table. These information-storing objects are created via various instprocs (e.g., Simulator::node{}) during the Network Configuration Phase. In the Simulation Phase, most objects access these information-storing objects via its instvar ns2), which is the reference to the Simulator.

6.6 MAIN COMPONENTS OF A SIMULATION

Interpreted Hierarchy

Created by various instprocs, the main OTcl simulation components are as follows:

- ✓ **The Scheduler** (scheduler_ created by instproc Simulator::init)maintains the chain of events and executes the events chronologically.
- ✓ **The null agent** (nullAgent_ created by instproc Simulator::init) provides the common packet dropping point.5
- ✓ **Node reference** (Node_ created by instproc Simulator::node) is an associative array whose elements are the created nodes and indices are node IDs.
- ✓ **Link reference** (link_ created by instprocs simplex-link{...} or duplexlink{...}) is an associative array. Associated with an index with format “sid:did”, each element of link_ is the created uni-directional link which carries packet from node “sid” to node “did”.

6.7 SETTINGS AND PERFORMANCE METRICS

100 mobile nodes starting from IP address 192.168.1.1 to 192.168.1.250 move in a 1500 x 1500 meter rectangular region for 100 seconds (simulation time). The

channel capacity of mobile nodes is set to 2 Mbps. Distributed Coordination Function (DCF) of IEEE 802.11 is used for wireless LANs. It has the functionality to notify the network layer about link breakage. It is assumed that each node moves independently with the variant mobility speed between 0.5 to 1.5 m/s. The transmission range is 100 to 250 meters. The simulated traffic is Constant Bit Rate (CBR) with varying initial energy between 0.5 joules. The simulation settings are also represented in tabular format, as shown in Table 6.1.

TABLE 6.1: Simulation Settings

No. of Nodes	**50, 75, 100, 125, 150 and 200**
Area Size	**1500 X 1500 meters**
MAC	**802.11b**
Radio Range	**250 meters**
Simulation Time	**100 seconds**
Traffic Source	**CBR**
Packet Size	**512 KB**
Mobility Model	**Random Waypoint Model**
Speed	**0.5 to 1.5 m/s**
Initial Energy	**0.5 Joules**

6.8 MOBILITY MODEL

An important factor in mobile ad-hoc networks is the movement of nodes, which is characterized by speed, direction and rate of change. Mobility in the "physical world" is unpredictable, often unrepeatable, and it has a dramatic effect on the protocols developed to support node movement. Therefore, different "synthetic" types of mobility models have been proposed to simulate new protocols. The term 'Synthetic' means to realistically represent node movement without using network traces.

Camp et al., 2002 discusses the following seven different synthetic entity mobility models based on random directions and speeds:

1. **Random Walk Mobility Model:** A simple mobility model based on random directions and speeds.

2. **Random Waypoint Mobility Model:** A model based on random waypoints and random speeds that includes pause times between changes in destination and speed, which will be discussed in more detail in the following section.

3. **Random Direction Mobility Model:** A model that forces mobile nodes to travel to the edge of the simulation area before changing direction and speed.

4. **Boundless Simulation Area Mobility Model:** A model that converts a 2D rectangular simulation area into a torus-shaped simulation area.

5. **Gauss-Markov Mobility Model:** A model that uses one tuning parameter to vary the degree of randomness in the mobility pattern.

6. **Probabilistic Version of the Random Walk Mobility Model:** A model that utilizes a set of probabilities to determine the next position of a mobile node.

7. **City Section Mobility Model:** A simulation area that represents streets within a city.

To thoroughly and systematically study a new Mobile Ad-hoc Network protocol, it is important to simulate this protocol and evaluate its protocol performance. Protocol simulation has several key parameters, including mobility model and communicating traffic pattern, among others.

The mobility model is designed to describe the movement pattern of mobile users, and also their location, velocity and acceleration as they change over time. Since mobility patterns may play a significant role in determining the protocol performance, it is desirable for mobility models to emulate the movement pattern of targeted real life applications in a reasonable way. Otherwise, the observations made and the conclusions drawn from the simulation studies could be misleading. Thus, when evaluating MANET protocols, it is necessary to choose the proper underlying mobility model. For example, the nodes in Random Waypoint model behave quite differently when compared to nodes moving in groups. It is not appropriate to evaluate the applications where nodes tend to move together using Random Waypoint model. Therefore, there is a real need for developing a deeper understanding of mobility models and their impact on protocol performance.

One intuitive method of creating realistic mobility patterns would be to construct trace-based mobility models, in which accurate information about the mobility traces of users could be provided. However, since MANETs have not been implemented and deployed on a wide scale, obtaining real mobility traces becomes a major challenge. Therefore, various researchers have proposed different kinds of mobility models, attempting to capture various characteristics of mobility and represent mobility in a somewhat 'realistic' fashion. Much of the current research has focused on the so-called synthetic mobility models [Camp et al., 2002] that are not trace-driven.

One frequently used mobility model in MANET simulations is the Random Waypoint model [Broch et al.,1998], in which nodes move independently to a randomly chosen destination with a randomly selected velocity. The simplicity of Random Waypoint model may have been one reason for its widespread use in simulations. However, MANETs may be used in different applications where complex mobility patterns exist. Hence, recent research has started focusing on the alternative mobility models with different mobility characteristics. In these models, the movement of a node is more or less restricted by its history, or other nodes in the neighborhood or the environment.

6.8.1 Random-Based Mobility Models

In random-based mobility models, the mobile nodes move randomly and freely without restrictions. To be more specific, the destination, speed and direction are all chosen randomly and independently of other nodes. This kind of model has been used in many simulation studies.

6.8.2 Random Waypoint Mobility Model

The Random WayPoint (RWP) mobility model has been widely used in mobile ad-hoc network simulations. This mobility model is a simple and straightforward stochastic model. In RWP, a mobile node moves on a finite continuous plane from its current position to a new location by randomly choosing its destination coordinates, its speed of movement, and the amount of time it will pause before when it reaches the destination. On reaching the destination, the node pauses for some time distributed according to some random variable and the process repeats itself.

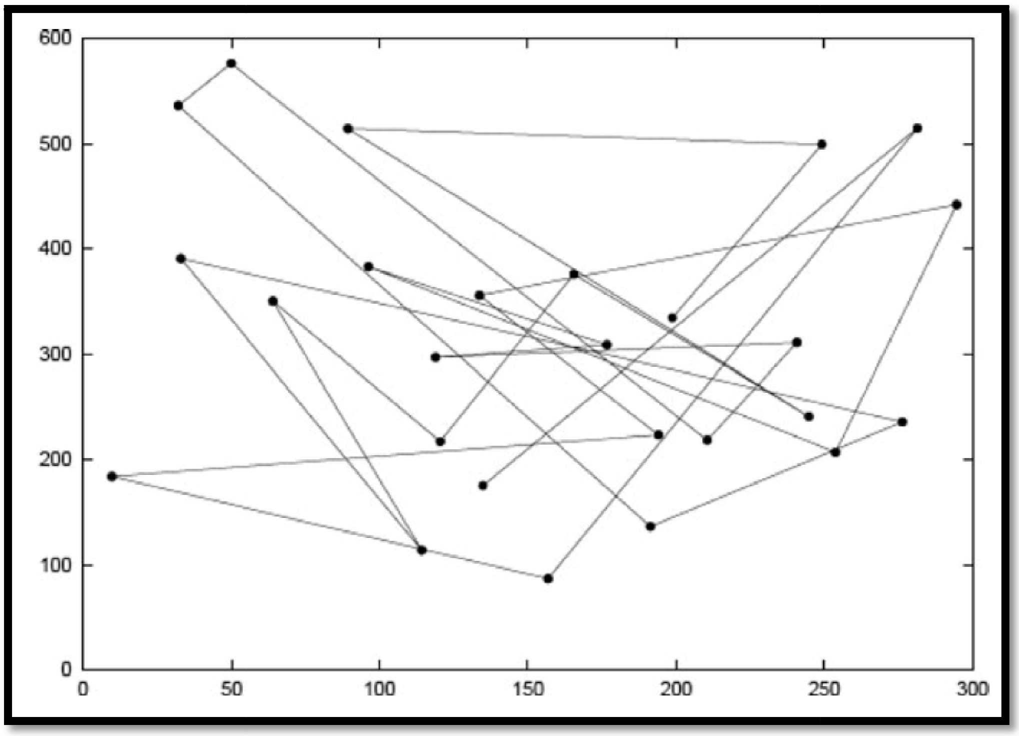

Fig 6.4 Traveling pattern of a mobile node using the Random Waypoint Mobility model

Once the pause time expires, the node chooses a new destination, speed, and pause time. The movement of a node from the starting position (waypoint) to its next destination (waypoint) is defined as one movement epoch, movement period, or transition time. The distance traveled between the movements of a node from the starting waypoint to its next waypoint is defined as the transition length. The destination points ("waypoints") are uniformly randomly chosen in the system area.

Figure 6.4 shows a example traveling pattern of a mobile node using the random waypoint mobility model starting at a randomly chosen point or position

(133, 180); the speed of the mobile node in the figure is uniformly chosen between 0 and 10 m/s. It is to be noted that the movement pattern of a mobile node using the random waypoint mobility model is similar to the random walk mobility model, if pause time is zero. In most of the performance investigations that use random waypoint mobility model, the mobile nodes are initially distributed randomly around the simulation area. This initial random distribution of mobile nodes is not representative of the manner in which nodes distribute themselves during their movement.

6.8.3 Transition Length and Duration

The RWP model correlates the speed and the direction change behavior. The time between two direction change events is no longer an adjustable input parameter of the model, later it depends on the speed of the nodes and the size and shape of the area. For a given area, a higher speed results in a higher frequency of direction changes. The speed behavior and the direction change behavior are investigated.

Consider RWP movement in a rectangular area of size $a \times b$ and again derive the distribution of the transition length L. Without loss of generality it is assumed that $a \geq b$. The spatial distribution of the 2D waypoints $P = (Px, Py)$ is now given by the uniform distribution

$$f P_x P_y(x,y) = \begin{cases} \frac{1}{ab} for\ 0 \leq x \leq a\ and\ 0 \leq y \leq b \\ 0\ \ else \end{cases} --- (1)$$

The distance between two points $P1 = (Px_1, Py_1)$ and $P2 = (Px_2, Py_2)$ is

$$L = \|P_2 - P_1\| = \sqrt{|P_{x1} - P_{x2}|^2 + |P_{y1} - P_{y2}|^2} = \sqrt{L_x^2 + L_y^2} --- (2)$$

Note that the random variable $L_x = |P_{x1} - P_{x2}|$ represents the random distance between two uniformly distributed coordinates *Px*1 and *Px*2 on a 1D line segment [0, *a*]. The same holds true for $L_y = |P_{y1} - P_{y2}|$ if it is replaced *a* by *b*. In addition, both random distances are independent of each other, and therefore the joint *pdf* of L_x and L_y is given by

$$fL_{x,}L_y(l_x, l_y) = fL\,(l_x) fL\,(l_y) = \frac{4}{a^2b^2}\,(-l_x + a)\,(-l_y + b) \text{for } 0 \leq l_x \leq a \; and \; 0 \leq l_y \leq b \; and \; 0, \text{otherwise} --- (3)$$

With this expression, it is possible to derive the cdf Pr $(L \leq l)$ by integration of L_x,L_y ($l_{x,}l_y$) over the circular area$D = \sqrt{l_x^2 + l_y^2} \leq l$ in the *(l*$_x$*-l*$_y$*)*- space, that is,

$$\Pr(L \leq l) = \iint_D^d f_{L_x} f_y \, (l_x, l_y) - - - (4)$$

As in the case of 1D, it cannot compute this integral in a straightforward manner, namely, by setting the right-hand side of Equation (3) in Equation (4), but must take into account that $fL_x, L_y\,(\,l_x, l_y) = 0 \; for \; l_x > a \; or \; l_y > a$. Thus , it is distinguished between three cases:

$$P_r(L \leq l) =$$

$$\begin{cases} \int_0^1 \int_0^{\sqrt{l^2 - l_x^2 b^2}} f(l_{x,} l_y) dl_{x,} dl_y \quad for \;\; 0 \leq l \leq b \\ \int_0^b \int_0^{\sqrt{l^2-b^2}} f(l_{y,} l_x) dl_{y,} dl_x + \int_0^{\sqrt{l^2 - l_x^2}} \int_{\sqrt{l^2-b^2}}^1 f(l_x, l_y) dl_{y,} dl_x \quad for \; b < l < a \\ \int_0^b \int_0^{\sqrt{l^2-b^2}} f(l_{y,} l_x) dl_{y,} dl_x + \int_0^{\sqrt{l^2 - l_x^2}} \int_{\sqrt{l^2-b^2}}^1 f(l_x, l_y) dl_{y,} dl_x \quad for \; a \; \leq l \leq \sqrt{l^2 + b^2} \\ \qquad\qquad l \end{cases}$$

$$---(5)$$

Solving these integrals, taking the derivative with respect to *l*, and performing some trigonometric simplifications lead to the *pdf* of the transition length *L* of nodes moving according to the RWP model in a rectangular area of size $a \times b$, $a \geq b$:

$$f_L(l) = \frac{4l}{a^2b^2} f_0(l) - - - (6)$$

With

$$f_0(l) = \begin{cases} \frac{\pi}{2}ab - al - bl + \frac{1}{2}l^2 & for\ 0 \leq l \leq b \\ ab \arcsin\frac{b}{l} + a\sqrt{l^2 - b^2} - \frac{1}{2}b^2 - al & for\ b \leq l \leq a \\ ab \arcsin\frac{b}{l} + a\sqrt{l^2 - b^2} - \frac{1}{2}b^2 - ab\ arccos\frac{a}{l} + b\sqrt{l^2 - a^2} - \frac{1}{2}a^2 - \frac{1}{2}l^2 & for\ a \leq l \leq \sqrt{a^2 + b^2} \\ 0 \ \ otherwise \end{cases}$$

$$- - -(7)$$

For arbitrary a, the value of $f_L(l)$ is obtained by $f_L(l) = \frac{1}{a} f_l$(l). The expected value of L is

$$E\{L^2\} = \frac{1}{15}\left[\frac{a^2}{b^2} + \frac{b^2}{a^2} + \sqrt{a^2+b^2}\ (3 - \frac{a^2}{b^2} - \frac{b^2}{a^2})\right]$$

$$+ \frac{1}{6}\left[\frac{b^2}{a} \arccos h\ \frac{\sqrt{a^2+b^2}}{b} + \frac{a^2}{b} \arccos h\ \frac{\sqrt{a^2+b^2}}{a}\right] - - - (8)$$

With arcos h (x) = In $\left(x + \sqrt{x^2 - 1}\right)$. Figure 4.4 shows the curve for E$\{L\}$/a over (b/a). For example, the expected length within a square a size a x a is E$\{L\}$ = The second moment of L is given by E$\{L^2\} = \frac{1}{6}\ (a^2 + b^2)$

It should be noted that the moments for the ID case are obtained, that is, $lim_{b \to \infty}\ E\{L\} = \frac{1}{3}a\ and\ limb_{b \to 0}\ E\{L\}\ = \frac{1}{6}a^2$.

The distance *pdf* $f_L(l)$ of a circular system area of radius a is derived as follows: i. P=p is the starting waypoint, ii. The conditional probability $P_r(L \leq l/p = p)$ is derived using basic geometric equation on the intersection area of two circles in polar coordinates, and iii. P$(L \leq L)$is the integration of the *pdf* $f_L(l)$ over all [possible starting waypoints in the area. The final result of these operations provides the transition length L of nodes moving according to the Rwp model on a disk [8] of radius a: in accordance of the following [1]:

$$f_L(l) = \begin{cases} \frac{8}{\pi a}\frac{l}{2a}\left[\arccos\frac{l}{2a} - \frac{l}{2a}\sqrt{1-\left(\frac{l}{2a}\right)^2}\right] & for\ \ 0 \leq l \leq 2a --- (9) \\ 0 & \end{cases}$$

Again, normalized random variable $L = L/2a$ for simplicity is introduced. The expected value of L is

$$E(L) = \int_0^{2a} lfL\,(l)dl = \frac{128}{45\pi}\ a = 0.9045a --- (10)$$

and its second moment is

$$E\{L^2\} = \int_0^{2a} l^2\, fL\,(l)dl = \ a^2\ \ --- (11)$$

6.8.4 Transition Time

The transition time is the time taken by a node to move from one waypoint to the next waypoint. The above results are used on the transition length to calculate the stochastic properties of the transition time. The random variable and an outcome, are denoted by τ. It is considered as follows: $V_i = v = const\ \forall i\ and\ v > 0.$ It implies that the speed of a node is constant during the entire movement. In this situation, it has:

$$T = \frac{1}{v} L - - - (12)$$

Hence, the expected transition time is

$$E\{T\} = \frac{1}{v} E\{L\} - - - (13)$$

and its *pdf* can be computed by

$$fr(\tau) = v f_L(v\tau) - - - (14)$$

With $E\{L\}$ and f_L taken from Equations (7) and (8) or Equations (10) and (11), respectively.

The speed of the node from a random distribution $f_v(\tau)$ at each waypoint (and then stays constant during one transition) instead of considering the speed as constant. hence T will be:

$$T = \frac{L}{V} - - - (15)$$

In this case, the random variable T is formed as a function $g(L, V) = L/V$ of two random variables L and V. In general, the expected value of a variable $g(L, V)$ can be expressed in terms of the joint *pdf* f_{LV}(l, v) as [15]

$$E\{g(L, V)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g\ (l, v) f_{LV}(l, v) dl dv - - - (16)$$

In this case, L and V are independent, and thus their joint *pdf* is $f_{LV}\ (l, v) = f_L\ (l) f_V(v)$. The expected value can then be simplified as follows:

$$E\{T\} = \ E\{T\} \int_{v_{min}}^{v_{max}} \frac{1}{v} fv\ (v) dv - - - (17)$$

6.8.5 Transition Length and Duration

Transition Length and duration can be explained more in detail by using uniform speed distribution.

Uniform Speed Distribution

If uniform speed distribution is employed within $[v_{max}, v_{min}]$, the expected transition time is

$$fr\ (\tau) = \int_{v_{min}}^{v_{max}} vf_L\ (v\tau)fv\ (v)dv \text{ For } 0 \leq \tau \leq \tau_{max}\ with \frac{l_{max}}{v_{min}} \text{ and } \mathrm{fr}(\tau) = 0$$

$$---(18)$$

$$E\{T\} = \frac{In\ (v_{max/v_{min}})}{v_{max} - v_{min}}\ E\{L\} --- (19)$$

Note that $lim_{v_{max}\rightarrow v_{min}} \mathrm{E}\frac{\{\mathrm{L}\}}{v} E_L/v_{min}$ corresponds to Equation (17) for constant speed v $= v_{min} = const.$ Further, it should also be noted that the expected time form $v_{min} = 0$ is undefined. This is very reasonable because if a node chooses V=0, the movement transition will take an infinite time. If the maximum speed can be expressed as a multiple of the minimum speed, that is, $v_{max} = kv_{min}$, with k > 1, it is obtained that

$$E\{T\} = \frac{In\ k\ E\ \{L\}}{k-l\ v_{min}} --- (20)$$

6.8.6 Time Duration

A node moves from one waypoint to another and then pauses from a certain time before changing direction in RWP model. The total T of an RWP period is then composed of a movement transition time T and pause time T_p. Now, it is possible to apply the above results for extending analysis for this case where a node rests a certain pause time in each way point as follows:

$$T = T + T_p - - - (21)$$

This linear combination of two independent random variables yields an expected value

$$E\{T'\} = E\{T\} + E\{T_p\} - - - (22)$$

and the *pdf*

$$f_{r'}(\tau') = \int_0^{\tau'} fr\,(\tau) fr_p(\tau' - \tau) d\tau \; for\; \tau' \geq 0 - - - (23)$$

The value of E{T′}represents the average time between two direction changes. Thus, the direction change frequency is given by $1/E\{T\}$ in unit 1/s.

6.8.7 Spatial Node Distribution

This section investigates the spatial distribution of nodes in RWP model considering a rectangle or circular system area A. In earlier investigation, the distance and time between two consecutive waypoints was analyzed. However, these waypoints, (which represent the starting and ending points of a node's movement period), are uniformly distributed per definition. In practical it is studied only as the single node because all nodes move independently. The random variable considered can be represented as $X = f(X, Y)$ that denotes the Cartesian location of a mobile node in A at an arbitrary time instant t. A particular outcome of this variable is denoted by x. The spatial distribution of a node in terms of the probability density function is provided as stated below:

$$f_x(x) = f_{xy\,(x,y)} = lim_{\delta \to 0} \Pr\left(\left(x - \frac{\delta}{2} < X \leq x + \frac{\delta}{2}\right) \wedge \left(y - \frac{\delta}{2} < Y \leq y + \frac{\delta}{2}\right)\right) / \delta^2$$

$$---(24)$$

Usually, a conversion to polar coordinates $R\sqrt{X^2 + Y^2}$ and $\emptyset$ = arctan (Y/X) yields the joint distribution $f_{x(x)}$ over this subarea, that is,

$$\Pr(node\ in\ A') = P\ (X\ \in A') = \int\int_{A'}^{1} f_{xy}\,(x,y)dA --- (25)$$

In Cartesian coordinates, the differential area element dA is given by $dA = dxdy$ and the resulting probability Pr ($X \in A'$) can be interpreted as the percentage of time that a given mobile RWP node is located in the subarea A' during a long – run movement process with many transitions. Since the simulation is done with many mobile RWP nodes (n>1), it can also be considered as the ensemble average. As a consequence, $E\{n'\} = nPr$ (node in A') denotes the expected number of nodes located in A' at an arbitrarily chosen time instant.

6.8.8 Stochastic Properties of Random Waypoint Model

Random Waypoint model as a discrete time stochastic process was described in Bettstetter et al.,2004. The transition length $L_i^{(j)}$ is defined as the distance covered by node j that moves from one waypoint to another during the *i*th epoch.

If the simulation field is a circular area with radius a. The probability density function of transition length L is

$$f_L(l) = \frac{8l}{2\pi a^2}\left[cos^{-1}\frac{1}{2a} - \frac{1}{2a}\sqrt{1 - \left(\frac{1}{2a}\right)^2}\right] --- (26)$$

Correspondingly, the expected value of transition length L is

$$E[L] = \int_0^{2a} l f_L(l) dl = 0.905a \quad --- (27)$$

and the variance of transition length L is

$$E[L^2] = \int_0^{2a} l^2 f_L(l) dl = a^2 \quad --- (28)$$

Bettstetter et al.,2004 took a further step to derive the probability distribution of transition time as follow

$$f_T(t) = \int_{V_{min}}^{V_{max}} v f_L(vt) f_v(v) dv \quad --- (29)$$

where $f_v(v)$ is the probability distribution function of movement velocity v and $f_L(l)$is the probability distribution function of transition length.

6.9 Critical Ad-Hoc Networking Features

Regardless of the application, there are certain critical features that can determine the efficiency and effectiveness of an ad-hoc network. These features can be categorized into quantitative features and qualitative features as given below.

6.9.1 Quantitative Critical Features

- ✓ **Network Settling Time** – The time required for a collection of mobile wireless nodes to automatically organize itself and transmit the first task reliably. Settling time is extremely important when a network has not been in operation for a while, it must then start-up and send messages promptly.

- ✓ **Network Join** – The time required for an entering node or group of nodes to become integrated with the ad-hoc network.
- ✓ **Network Depart –** The time required for the ad-hoc network to recognize the loss of one or more nodes and to reorganize itself to route around the departed nodes.
- ✓ **Network Recovery Time** – The time required for the network to recover after a condition that dictates reorganization of the network.

  To be more specific, it refers to the time required, (i.e)

  1. For a collapsed portion of the network, due to traffic overload or node failures, to become functional again once the traffic load is reduced or once the nodes become operational
  2. For the network to reorganize due to node mobility and resume reliable communication.
- ✓ **Frequency of Updates Overhead** - The number of control packets bytes or overhead bytes in a packet required in a given period to maintain proper network operation.
- ✓ **Memory Byte Requirement** - Storage space requirements in bytes, including routing tables and other management tables.
- ✓ **Network Scalability Number –** The number of nodes that the ad-hoc network can scale to and reliably preserve communication. The network should be able to scale to 10,000 nodes.

6.9.2 Qualitative Critical Features

Some of the qualitative critical feature are as follows:

- ✓ **Knowledge of Nodal Locations** - The routing algorithm requires local or global knowledge of the network.

- ✓ **Effect to Topology Changes** - The routing algorithm needs complete restructuring or incremental updates.
- ✓ **Adaptation to Radio Communication Environment** - Nodes use the estimation knowledge of fading, shadowing, or multiuser interference on links in their routing decisions
- ✓ **Power Consciousness** - The network employs routing mechanisms that consider the remaining battery life of a node
- ✓ **Single or Multichannel** - The routing algorithm utilize a separate control channel in some applications, for multichannel execution may make the network vulnerable.
- ✓ **Bidirectional or Unidirectional Links** - The routing algorithm performs efficiently on unidirectional links.
- ✓ **Preservation of Network Security** - The routing algorithm uphold the fidelity of the network.
- ✓ **QoS Routing and Handling of Priority Messages** - The routing algorithm support priority messaging and reduction of latency for delay sensitive real time traffic. The network can send priority messages/voice even when it is overloaded with routine traffic levels.
- ✓ **Real-time Voice Services** - The network support simultaneous real-time multicast voice while supporting routine traffic loads associated with situation awareness and other routine services.
- ✓ **Real-time Video Services** - The nodes can receive or transmit video on demand, while still supporting traffic levels associated with situation awareness, voice conversations and other routine services.

6.9.3 Performance Metrics

Performance Metrics are categorized into three 3 types namely,

- ✓ Thread-Task Level Metrics
- ✓ Diagnostic Packet Level Metrics
- ✓ Scenario Metrics

6.9.4 Thread-Task Level Metrics

Thread-Task Level Metrics provide a good method for assessing a MANET algorithm:

- ✓ **Average Power Expended** - Average power expended Watts in a network is a given time period to complete a thread task. This will include power expended in transmitting control messages and information packets.
- ✓ **Task Completion Time** – The time taken to complete a task in seconds. It is a measure of implicit complexity and overhead to complete a given task.

6.9.5 Diagnostic Packet Level Metrics

Diagnostic Packet Level Metrics characterize network behavior at the packet level.

1. Overhead analysis
2. Packet Delivery Ratio
3. Delay
4. Energy Consumption

6.9.6 Scenario Metrics

The following Scenario Metrics describe the network environment and also define the scenario.

- ✓ Nodal Movement/Topology Rate of Change - Average speed of nodes
- ✓ Number of Network Nodes
- ✓ Area Size of Network
- ✓ Density of Nodes per Unit Area
- ✓ Offered load and traffic patterns
- ✓ Number of Unidirectional Links

6.9.7 Strategy for Computing Critical Ad-Hoc Networking Features

Network Settling Time

The nodes of an ad-hoc network must dynamically discover one another and organize themselves into a functioning network capable of reliable communication. Network Settling Time is a meaningful metric to capture the ability of network to do this is the.

The Network Setting Time is the time required for a collection of mobile wireless nodes to automatically organize themselves and transmit the first task reliably.

The first task may be one of the following:

- ✓ **Funnelcast** - A SITREP from each node member to another designated member. For example, a squad reporting in its location and status to a squad leader following a parachute landing.

- ✓ **Multicast** - A multicast voice to all squad members. For example, a squad leader issuing a command or point man issuing a warning.
- ✓ **Unicast -** A single message to another single address. For example, a call fire or sensor to shooter transaction.

The Network Settling Time for the above three cases will be determined for different network topologies. The network topology might be any one of the following:

- ✓ **Fully Connected Stationary Network -** Any node can communicate directly with any other node in the network (dense network).
- ✓ **Strongly Connected Stationary Network -** Any node can communicate with any other node in the network within two hops. All nodes have at least two neighbors.
- ✓ **Weakly Connected Stationary Network -** Some nodes might have only one neighbor and some nodes may be separated by up to five hops.
- ✓ **Strongly Connected Mobile Network** - Added nodal mobility.
- ✓ **Weakly Connected Mobile Network -** Added nodal mobility

Network Join

A mobile ad-hoc network must be able to accommodate a new node or group of nodes into the network and to continue efficient operation. A meaningful metric to access this ability is the Network**.**

Join Time.

Network Join Time is the time required for an entering node or group of nodes to become integrated into the ad-hoc network and participate in the routing functions of the network. In each configuration, two situations will be considered. First, a single new node will enter the network in a specified location. Second, a group of 5 nodes will enter the network in pre-determined locations.

Network Depart

A mobile ad-hoc network must be able to accommodate the loss of one or more nodes from the network and continue efficient operation. Network Depart is a meaningful metric to access this ability is the Network.

Depart Time.

The Network Depart Time is the time required for the network to recognize the loss of one or more nodes, and reorganize itself to route around the departed nodes.

In each configuration, two situations will be considered. First, a single centrally located node will depart the network. This will give a worst-case analysis, because this case requires the most of reorganization from the network. Second, a group of 5 nodes will depart from the network.

Network Recovery Time

Dynamic conditions in a mobile ad-hoc network would require reorganization of the network. Portions of a mobile ad-hoc network might collapse or become in operational due to excess traffic or mechanical node failures. Moreover, since the nodes are moving, the network topology is constantly changing either in a predicted and uniform fashion, or totally unpredictably. Network Recovery Time is a meaningful metric to capture the ability of the network to reorganize itself and resume reliable communication in the network.

Network Recovery Time is the time required for a network to reorganize itself and resume reliable communication after 1 a portion of the network collapses due to traffic overload or node failures, or 2 the topology changes due to nodal mobility and variable link quality.

- ✓ Traffic Overload - The traffic generation rate at the 5 connected nodes will be increased in increments until that portion of the network becomes saturated and collapses.
- ✓ Node Failures - 5 connected nodes will be turned into simulate node failures. This is similar to the scenario for Network Depart, with the nodes being connected. The nodes will be turned on again.

6.9.8 Frequency of Updates

In a mobile ad-hoc network, control messages/overhead is necessary to disseminate necessary information for network organization and routing execution. The overheads, however, will depend on the protocols being used. A meaningful metric to capture the overhead required for a protocol is the Frequency of Updates - Overhead. More specifically, the Frequency of Updates - Overhead is the amount of overhead information measured in bytes in a given period that is required to maintain proper network operation. The amount of overhead required would implicitly depend on the network configuration, i.e., a sparse network might require a higher exchange of control information when compared to a dense network.

For each configuration, the Frequency of Updates - Overhead will be measured at two different stages of network operation:

- ✓ Initialization - From network startup until the Network Settling, the bytes of overhead exchanged in the network will be computed.
- ✓ Maintenance - For a period of X seconds during stable network operation, the bytes of overhead exchanged in the network will be computed.

These two network stages would capture the overall overhead of a protocol since some protocols might have more expensive bytes overhead at one stage of network operation when compared to the other.

Memory Byte Requirement

In a mobile ad-hoc network, routing paths are determined by the nodes themselves. Hence, memory storage requirements are imposed on all nodes. The amount of storage necessary will depend on the characteristics of the routing protocol, e.g, hop-by-hop routing vs. source routing and proactive routing vs. reactive routing. A meaningful metric to capture the amount of storage required for a protocol is the Memory Byte Requirement. More specifically, the Memory Byte Requirement is the amount of memory storage required in bytes to store routing tables, neighbor tables, and other management tables. The following two network configurations will be considered:

1) Network Scalability Number

A mobile ad-hoc network should be able to scale in number of nodes and still provide efficient functionality. Network Scalability Number is a meaningful metric to capture this ability.

The Network Scalability Number is the number of network nodes that the ad-hoc network can scale to and reliably preserve communication.

For each configuration, the number of nodes will be increased by N = 100 nodes at a time. The value of N can be modified based on feedback from experiments. For each increment, a multicast voice message will be transmitted by a designated node, e.g, platoon leader. Each node that receives the transmission will acknowledge the packet through the acknowledgment mechanisms provided either as passive acknowledgments or as active acknowledgments. The largest number of nodes that can function reliably by successfully transmitting multicast voice messages will be computed as the Network Scalability Number.

2) Qualitative Critical Features

Global assessments of the network do not depend on the type of network topology or messages being transmitted.

6.9.9 Strategy for Computing Performance Metrics

Average Power Expended

The average power expended in the network is highly dependent on the hardware utilized. A protocol may perform efficiently on a particular radio platform and inefficiently on another. Hence, a simulation will be executed for each radio node architecture provided in SEAMLSS, i.e., WL, HMT, and IEEE 802.11.The average power expended will be computed as the power utilized in

- ✓ transmitting information packets
- ✓ transmitting control packets
- ✓ receiving information packets
- ✓ receiving control packets

The power expended in sending information and control packets is simply the transmitter power utilized. The power expended in receiving packets is difficult to be quantified precisely. As an initial approach, a cost of Y will be charged for receiving an information packet, and a cost of Y=2 will be charged for receiving a control packet. These costs may be modified if necessary based on experimental results.

6.10 Summary

This chapter dealt with overview of Network Simulator and its architecture. C++ class hierarchies which are base for NS2 and the components responsible for packet forwarding are discussed. Performance Metrics and the Settings used for the simulation used are discussed. Different mobility models based on random directions and speeds were discussed.