Improved Performance for Network Simulation

A Thesis

Submitted to the Faculty

of

Drexel University

by

Bryan J. Willman

in partial fulfillment of the

requirements for the degree

of

Master of Science in Computer Engineering

December 2007

Acknowledgements

I would like to sincerely thank my advisor, Dr. Steven Weber, who has made this work possible. This thesis is the culmination of my graduate studies, and is the direct product of his input and insightful guidance. I greatly enjoyed working under Dr. Weber. I am extremely grateful for both the time and intellectual stimulation that he has graciously shared with me during my time at Drexel.

I am grateful to all the students and faculty that I had the opportunity to interact with. Many thanks to my best friend as a graduate student, Ananth Kini, who helped provide several corrections and revisions throughout this thesis. I wish to thank Jeffrey Wildman for all his help and our many productive discussions. Also, I would like to thank the National Science Foundation who have provided financial support for this work.

Finally, I wish to thank my mother, who has been my best friend and biggest supporter in life. Her love and moral support have nurtured and encouraged me to pursue my personal and academic interests. It is to her that I dedicate this work.

Table of Contents

List of Tables

List of Figures

Abstract

Improved Performance for Network Simulation

Bryan J. Willman

Over the course of designing and implementing two discrete event simulators, the commercial simulator packages CSIM and DesmoJ were leveraged to allow for rapid development of both wired and wireless network models. However, the two resulting simulators demonstrated poor scalability due to the use of multi-threading to maintain state for simulation elements. By using a simple single-process discrete event simulation engine, the running-time showed a marked decrease when compared to multi-threaded simulators.

In one case study, we simulate a simple two-link MPLS network which employs two congestion control mechanisms for inelastic traffic, namely preemption and adaptation. Performance metrics measured include: the per-class blocking probability, customer average fraction of time streams travel on the preferred path, customer average fraction of time at the maximum subscription rate, the customer average rate of adaptation, and the time average rate of preemption. We compare the performance of preemption and adaptation individually and collectively against the base case where neither congestion mechanism is used. At the cost of increased number of rate adaptations and preemption events for a range of regimes, we show that the combined use of preemption and adaptation improves the quality of service and alignment of high priority traffic while increasing the effective network capacity. As a performance enhancement to the simulator developed to conducted these experiments, we switched to a single-process discrete event simulation engine in place of multi-threaded simulator. We note a large improvement for the running time as the simulation time and capacity increase.

A second case study was conducted on a wireless simulator. In an effort to simplify the simulator and improve performance we again moved from a commercial thread-based simulator (CSIM) to a single-process discrete event simulation engine. Results of the running-time vs network size for the single-process simulator showed a constant-time improvement over the thread-based simulator. To further improve performance, a complementary technique known as model abstraction is also applied. Model abstraction is a technique that reduces execution time by removing unnecessary simulation detail. In this thesis we propose three abstractions of the IEEE 802.11 protocol. The Goodput Ratio vs Transmission Power and End-to-end delay vs offered load performance metrics are compared against the OPNET commercial simulator.

1. Introduction

In recent years there has been an enormous growth in global communications services, wireless communication, and mobile devices. The advent of new wireless applications and technologies come at a time when demands on the global Internet are ever increasing. Network designers face new challenges to support the growing demand for data, audio, and video applications. As a result, new protocols and technologies are being submitted and evaluated to meet the changing operational requirements.

We are currently entering the mobile computing revolution that is poised to change the way people live, work, and interact. Mobile computing devices are becoming increasingly prevalent thanks to the increased deployment of wireless area networks, advances in hardware, and the introduction of new and useful mobile applications. Many forward-thinking researchers envision a futuristic computing environment where mobile devices are as ubiquitous as the telephone and seamlessly collaborate to perform services automatically [77]. Wireless networks consist of radios which communicate via a wireless channel. One form of a wireless network uses a wired backbone, where the last hop is wireless. Another form of wireless networking are ad hoc networks. Ad hoc networks are self-organizing wireless nodes that operate without any preexisting infrastructure. Mobiles move around free to join and leave the network. In addition to providing seamless next-generation communication between heterogeneous devices, ad hoc networks are attractive for military, disaster recovery, and other tactical communications. Relevant areas of wireless networking research include sensor networks, mobile agents, and ad hoc networking routing protocols.

Accurate analysis of proposed wireless systems require the development of efficient modeling tools. Simulation is an invaluable tool used to model complex systems where the desired network size is large in scale. Simulators have a large number of applications and are the primary tool for the network engineer. They allow researchers to easily study the effects of proposed policies and changes to existing systems that otherwise may not be possible with mathematical analysis. Analytical models provide tractable analysis, yet they are often inflexible for the general case. Moreover, mathematical models are often fraught with simplifying assumptions that may not always hold or are downright false. Therefore, simulation emerges as the ideal choice to model realistic systems. Other areas where simulation is applied include: manufacturing, aerospace, transportation, healthcare, communication, defense, information processing, and queuing systems in general [69].

One fundamental problem with simulation is obtaining meaningful results in a timely fashion. Current simulators are unable to simulate large-scale wireless networks in moderate detail. Memory and processing speed limitations often constrain the feasible size of the network to a couple thousand nodes. As an example, the NS-2 simulator requires over a week of simulation time on a 1 GHz Pentium-III PC to simulate 600 seconds of an ad hoc network consisting of only 300 nodes [53]. In another example, the authors in [17] compare the performance of DSR and AODV using the NS-2 simulator. The largest scenario simulated was a 100 node ad hoc network on an area of 2200 x $600m^2$, for a duration of 500 seconds. The authors complain that they were unable to simulate larger scenarios due to the excessive run time of the simulator.

While it is sometimes reasonable to analyze small networks, the current lack of scalability found in high fidelity simulators prohibits the study of larger systems of interest. One

method for improving scalability is to employ performance programming techniques. For example, the NS-2 simulator suffers a performance penalty by invoking the Tcl interpreter. Simply replacing the interpreted code with compiled functional equivalents offer an order of magnitude improvement. Performance programming techniques improve performance in sections of code that monopolize a large and/or unnecessary fraction of the running time.

In addition to performance programming, previous studies of network simulation suggest two primary methods for improving simulation performance, parallel programming [46, 44, 72] and model abstraction [5, 12, 30]. Parallel programming divides computation among multiple processors either by geography or channel frequency. While parallel programming improves scalability without sacrificing accuracy, it has the drawback of complexity and relies on a multiprocessor architecture or a dedicated network of distributed computers. Parallel programming is seen as a complementary technique to abstraction. Model abstraction is a technique that aims to reduce execution time by removing unnecessary simulation detail, hopefully obtaining comparable results. Essentially, abstraction is a trade-off between performance and accuracy which may not always be appropriate. A common example of model abstraction is a network fluid model that represents a stream of packets as a flow of CBR traffic.

Over the course of designing and implementing two discrete event simulators, the commercial simulator packages CSIM and DesmoJ were initially leveraged to allow for rapid development of both wired and wireless network models. However, the two resulting simulators demonstrated poor scalability, namely due to the use of multi-threading in the state maintenance of simulation elements. Using a simple single-process discrete event simulation engine, the rate of growth for the running time showed a significant decrease. In

addition, the technique of model abstraction was applied to further improve performance.

1.1 Scope of Thesis

Network simulation is a broad subject area that encompasses many facets of research. In this thesis we perform two case studies, using two separate simulators. In the first case study, we investigate the combined use of two congestion control protocols applicable to streaming media, namely preemption and adaptation, on a simple two-link network. Previous research has addressed the use of preemption and adaptation independently, whereas we investigate the use of both as complementary methods to reduce congestion. We compare performance using several metrics. The MPLS simulator developed for this study was originally implemented using the Desmo-J [57] and JFreePlot [26] libraries.

In a second study, we investigate performance improvements for the IEEE 802.11 data link protocol used in wireless networks. To this end, we developed three abstraction models that attempt to remove unnecessary simulation detail for a decreased runtime. The simulator for this study was originally implemented using the CSIM simulation library. We compare the goodput ratio vs transmitting power, the end-to-end delay vs offered load, and the runtime factor vs network size of our abstraction models to an industry accepted standard simulator, OPNET [19].

In both case studies, we attempted to improve efficiency and reduce complexity by removing the third-party libraries and replacing them with a simple single-process discrete event scheduler. We compare the runtimes of both the single-process and commercial library implementations for the same scenarios.

1.2 Outline of Thesis

Section 2 provides an introduction to computer simulation, including the simulation infrastructure. An introduction to modeling of ad hoc wireless protocols is presented in Section 3. Included is a brief analysis of the computational complexity. In Section 3.1.5, model abstraction is used to improve scalability of 802.11 models. Section 3.1.8 presents a problem encountered when utilizing commercial simulation packages and our proposed solution. Section 4 describes the congestion response protocols implemented in the MPLS simulator. Section 5 summarizes our research and suggests possible topics of future work.

2. Simulation

Simulators today generally may be categorized as either continuous, discrete, or Monte Carlo. Monte Carlo simulation is used to iteratively solve a deterministic model using random inputs. These iterations turn the deterministic system into a stochastic process. The notion of time is not required. In a continuous simulator, the simulation time is modulated by continuous variables which are expressed as differential equations. During execution the simulator integrates the differential equations. Solving differential equations may be computationally expensive. Consequently, they are often used when there are a small number of continuous components in the system. The third method of simulation, discrete event simulation, updates time at *discrete* instants in time when an event occurs that causes the system state to change. Discrete components may be used in conjunction with continuous components in what is termed *combined* simulation. *Hybrid* simulation is used to describe simulators which use an analytical sub-model within a discrete event simulator. The focus of this thesis will be on the efficient implementation and design of discrete event simulators in a networking context.

Developing a simulator involves the modeling of real world systems in software. The concept of entities are used in software to represent real-world objects. These entities may be temporary or permanent and have relationships as dictated by logic statements. The logical relationships define the overall behavior of the model. As an example, we could model a telephone switching network using entities of switches, links, and calls. The links and switches forming the network topology would be constant entities, while calls would

travel on links for a temporary period of time before leaving the system. The notion of time is maintained and updated by the simulation engine.

2.1 Discrete Event Simulation

The primary data structure for the discrete event simulator is the event queue that contains a time ordered list of events. Events contain the time a specific action is to occur and the logic associated with that action. It is the task of the simulation engine to order events. The pseudocode in Algorithm 1 describes how events are scheduled in a discrete event simulator. At it's core, the simulation engine is a while loop that sets the current time to the time of the first event in the event queue. The event queue in this example is implemented as two separate queues, but in practice a single queue could just as well have been used. One queue is used for customer arrivals while the second contains departure events. The determination of which queue has the next event is arbitrated by taking the minimum time as shown in line three. The simulation engine removes the first event using a queue *pop* operation, and may subsequently generate one or more future events. The time is drawn randomly with a distribution defined in the $rand()$ function (not defined here). When events are added to the queue, a sort operation is required to maintain time order. It is important to note that an event may schedule an additional event for the present or future.

Figure 2.2 illustrates the role of the various components that typically make up a discrete event simulator. Collection of results and providing statistical distributions are additional features provided by the simulation engine.

Memory management and list processing consume a large proportion of the simulation time for the simulation executive. When the number of events grows large the choice

Algorithm 1 Discrete event simulation

```
1:  #INITIALIZE: time, sim_duration, arrival_mean, departure_mean
2:  while time ≤ sim_duration do
3:     time = min(arrival_queue.top(), depart_queue.top())
4:     if time = arrival_queue.top() then
5:        arrival_queue.pop
6:        #SCHEDULE ARRIVAL
7:        arrival_queue.add(rand(arrival_mean) + time)
8:        arrival_queue.sort()
9:        #SCHEDULE DEPARTURE
10:       depart_queue.add(rand(departure_mean) + time)
11:       depart_queue.sort()
12:    else
13:       #PROCESS DEPARTURE
14:       depart_queue.pop()
15:    end if
16: end while
```

of data structure used for the event queue becomes of primary importance. A min-heap implementation of a priority queue is commonly used to order events by time. The basic operations on the event queue are the insert and remove-min events, also referred to as enqueue and dequeue respectively. Figure 2.1 provides an example of a min-heap where each node is represented by it's time value. Scheduling a new event translates into an enqueue operation on the heap which takes $O(\log n)$ time for a n-element heap. When an event is processed a dequeue or *pop* operation is used to remove and return an event from the top of the heap in $O(\log n)$ as well. Accessing the top element of the heap is a constant time operation.

In general, the priority queue implementation can perform any operation in $O(\log n)$, yet there are additional data structures (e.g. splay trees, binomial trees) that have been shown by Jones in [36] to improve upon the base performance of the heap in practice. In an

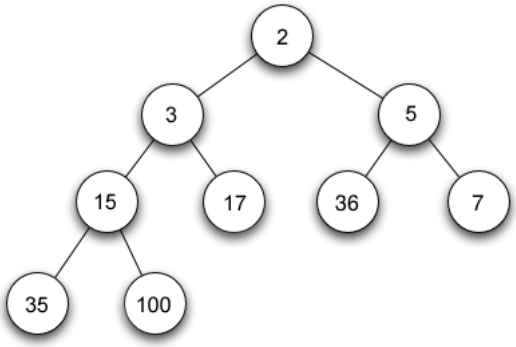

Figure 2.1: A min-heap

extension of Jones' work, LaMarca & Ladner apply concepts of performance programming to investigate the effects of caching on the heap and other priority queue data structures. They show improvements up to 75% when considering cache performance in algorithm implementation. However, the most promising data structure in the literature is the Dynamic Calendar Queue [2], an improvement upon the originally proposed calendar queue, that promises a $O(1)$ running time on average. The Calendar Queue divides time into N buckets of width w, whereby each bucket contains a priority queue for a period of time, (or max priority of size $\frac{n}{N}$), where there are n events in the queue. To ensure optimal performance, the size of the buckets w, (i.e. the time duration) is dynamically re-sized during execution to maintain small priority queues.

2.2 Object Oriented Design

The implementation and design of the simulation engine play a major role in simulation performance as well as production time. Object oriented design principles can be applied to simulation to produce modular components called classes that can easily be modified and

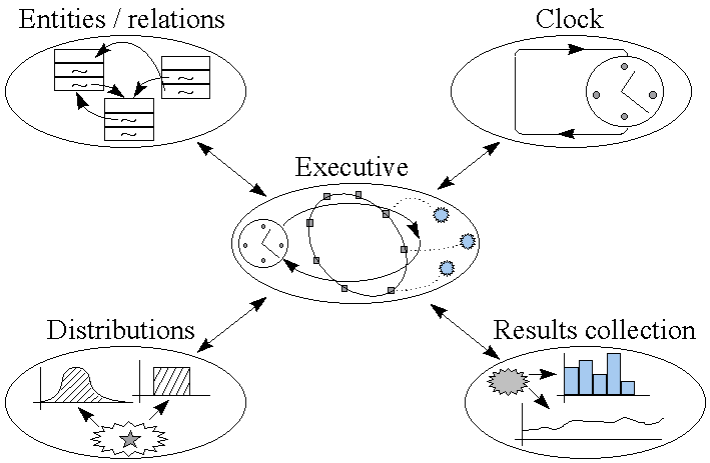

Figure 2.2: Discrete event simulator framework [39]

reused. Libraries of self contained classes can be produced to create simulation applications. These classes may represent real world objects as software entities or simply provide simulation processes and routines. The major strengths of Object Oriented Programming (OOP) derives from the storage and usage of data. Procedural code disperses data and routines throughout the code base, whereas OOP objects contain the data and routines in one place. The Event class in Algorithm 2 is a class in the wireless simulator that serves as an example of the object oriented paradigm. Instantiation and removal of an Event object are automatically handled in the Event constructor and destructor respectively. There is no need to copy code (e.g., memory allocation, etc.) that is needed each time a new Event is created or destroyed. Moreover, functionality related to this part of the simulator can be maintained in a separate Event class file.

Another benefit of using OOP is that of inheritance, which makes it easy to extend the functionality of existing objects (base classes) by creating new classes that inherit the functionality of a base class. For example, the Event class in Algorithm 2 is extended by

Algorithm 2 OOP example - the event class

```
class Event
{
  public:
    Event() {}
    Event(Node *n, double t, std::string e_type)
    {
      setNode(n);
      setTime(t);
      setType(e_type);
    }
    virtual  Event()
  {
    //std::cout<<"Event Destructor"<<std::endl;
  }

    //Mutator Functions
    void setNode(Node *n)
    {
      if(n) { node=n; { else { std::cout<<"Warning: Null pointer passed to Packet::setSourceNode(...)"<<std::endl; }
    }
    void setTime(double t) {time = t;}
    void setType(std::string e_type) {type = e_type; }

    //Accessor Functions
    Node* getNode() const {return node;}
    double getTime() const {return time;}
    std::string getType() const return type;

    //Pure virtual function to be inherited
    virtual void processEvent() = 0;

    //Supported Operators
    bool operator<(const Event &e2) { return (getTime() < e2.getTime()); }
    bool operator>(const Event &e2) { return (getTime() > e2.getTime()); }
    bool operator<=(const Event &e2) { return (getTime() <= e2.getTime());}
    bool operator>=(const Event &e2) { return (getTime() >= e2.getTime());}
    friend std::ostream &operator<< (std::ostream &output, const Event *e); }

  private:
    Node *node;
    double time;
    std::string type;
};
```

an inherited class $TXEvent$ which implements a more specific type of event when a node transmits a datalink frame. Virtual functions of the base Event class can be overridden in the $TXEvent$ class to perform functionality specific to the datalink layer.

2.3 Random Number Generators

The Pseudo Random Number Generator (PRNG) is a central component of the stochastic simulator. The choice of PRNG can have a significant effect on simulation results. The authors in [58] found that as much as 50% of the CPU cycles for most network simulations were devoted to generating random numbers. Moreover, common random number generators have been shown by L'Ecuyer [42] to be insufficient for certain applications and statistical tests. In [58], the authors estimate that the NS-2 PRNG has a cycle length of only a few thousand numbers before potential non-uniformity or cycling of numbers. This is clearly insufficient for simulations consisting of thousands of nodes and millions of transmissions. PRNG's are libraries developed to produce mutually independent, uniformly distributed random variables over the range $[0, 1]$. Other distributions may then be generated by applying the appropriate transformations. A desirable PRNG, as described by L'Ecuyer, will have the following properties:

- **Long Period** - Every Random Number Generator is based on a state that eventually repeats itself. In practice a period of at least 2^{60} is sufficient.

- **Theoretical Basis** - Another desirable property is that the PRNG have good statistical properties. However, if a theoretical basis does not exist it does not imply that the PRNG is inadequate.

- **Efficient** - Especially for simulation purposes, a good PRNG will produce random numbers quickly, while consuming a small amount of memory.

- **Reproducibility** - It is important to be able to reproduce random streams for the purposes of program verification and variance reduction. This is typically implemented by specifying a random seed.

- **Pass Statistical Tests** - There are several tests that can be performed to determine the quality of a PRNG. The most popular of these are the stringent DIEHARD tests [48]. Not all PRNG's pass every test, therefore better PRNG's will only fail the most difficult or complex of tests.

- **Portability** - The ability to use a PRNG on multiple hardware platforms and operating systems is necessary for practical reasons.

The Mersenne Twister presented in 1994 by Makoto Matsumoto and Takuji Nishimura [49], has been adopted as the PRNG of choice for simulation purposes, including the simulators developed in this thesis. The Mersenne Twister uses a Twisted Generalized Feedback Shift Register, the *twist* guarantees equidistribution in 623-dimensions with up to 32 bits of accuracy. Comparatively, conventional linear congruent PNRG's show obvious patterns for just 2-dimensional plots. The Mersenne Twister also has a huge period of $2^{19937} - 1$, while consuming only 624 words of memory. It has passed the DIEHARD [48] and other stringent statistical tests with a speed comparable to the fastest generators. In it's native form it is not suitable for cryptographic random number generation, however combining a hashing algorithm mitigates this problem at the cost of performance.

2.4 Verification and Validation

Throughout the course of modifying and developing the simulator, the model needs to be verified and validated before conducting experiments. Verification is a measure of how accurate the problem or model has been transformed into software. In other words, verification deals with building the model correctly. Validation is a comparison of simulation results with that of a known baseline, i.e. validation provides a metric for building the correct model. Verification may be performed using methods of code inspection and model checking, while validation is ideally performed by comparing measurements and results of real networks to the results of the developed model. However, due to the hardware costs and complex tests required, it is often infeasible to test a proposed protocol or simulation model against a real system. Instead, results are often compared to analytical approximations. In addition to the protocol modules, the simulation framework and the complex interaction between various models needs to be verified and validated as well.

After the simulator has been verified and validated, the simulator needs to be configured. Each protocol has associated with it many variables that could vary between versions or simulators. To permit repeatable results, configuration parameters should be documented or provided. Additionally, the scenario lists several key parameters such as the network topology, traffic rate, simulation area, etc. Because a set of benchmark scenarios have not been standardized, researchers have selected scenario parameters arbitrarily. This can result in misleading results where inputs can be selected to produce desirable output. For this reason it becomes necessary to sweep a range of input parameters.

2.5 Simulation Initialization

An important step that is all too often overlooked or skipped is the simulation setup. The first phase of the simulation setup is determining the type, and stopping criteria. The simulation may be set to run for a predetermined fixed-time or may stop when reaching steady state. Researchers are often interested in the long term average. However, a common pitfall is to claim the simulation results have reached steady state without assuring the degree of convergence.

Execution and collection of data are two subtle issues of simulation. Simulation typically begins at an initial state where there are no customers in the systems; over the course of the simulation the system fills up. Consequently, the results are skewed by the initial transient from the idle state to point that the long term average is achieved. This is referred to as the steady state. A *warm-up* period may be used to alleviate this bias, where the collection of data does not begin until steady state is reached. This results in wasted simulation time, moreover it is not usually apparent when to start collecting data. One formalized approach measures the arrival and departure rates for the system, data collection begins once the difference becomes negligible [81]. A relatively new technique known as Perfect Simulation [62] has been discovered where it is possible to sample from the stationary distribution at the beginning of the simulation. Le Boudec recently applied Palm calculus to obtain the stationary distribution and perfect simulation to begin simulation of mobility models for ad hoc wireless networks without a transient period [41].

2.6 Simulation Stopping Criteria

Once data collection begins, a natural question is when to stop the simulation. Typically, a simulation is run until the metric of interest converges to an average, also referred to as a point estimate. Alternatively, the simulation may be run for a preset period of time (runtime), but this is generally a poor practice. Point estimates do not provide a guarantee of accuracy which motivates use of a *confidence interval* as the stopping criterion. A *confidence interval* is a range of values in which the true solution is believed to lie within, with an associated confidence level. The ascribed *confidence level* denotes the probability that the solution is within the confines of the confidence interval. A reasonable stopping criteria for a simulation is to iterate until the width of the confidence interval for a performance metric of interest is acceptably small, say v [74].

2.6.1 Confidence Interval

For an observed performance metric of an iid sequence, $X \sim X_1 ... X_n$, we would like to compute confidence interval around the point estimate $\hat{X}$ such that

$$\mathbb{P}\left(\mu \in \left[\hat{X} - \Delta, \hat{X} + \Delta\right]\right) \approx 1 - \alpha, \tag{2.1}$$

where μ is the mean of the sequence X, 2Δ is the width of the confidence interval, and α is the power of the interval. By the Central Limit Theorum, the above approximation holds with equality as $n \to \infty$. Typical values for α are 0.2, 0.1, and 0.05 which translate into a confidence level of 90%, 95%, and 99% respectively. Algorithm 3 illustrates how we compute the confidence interval for a sample that exhibits an expectation equivalent to that

of the mean μ (e.g. Bernoulli, or Normally distributed). Utilizing the recursive form of the mean estimator we compute the point estimate as follows

$$\begin{aligned} \hat{X}_n &= \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{n}\left(\sum_{i=1}^{n-1} X_i + X_n\right) \\ &= \frac{1}{n}\left((n-1)\hat{X}_{n-1} + X_n\right) \\ &= \frac{n-1}{n}\hat{X}_{n-1} + \frac{1}{n}X_n \end{aligned}$$

where n is the number of random samples. Similarly, we can find a recursive point estimator for the variance. First define $V_n = \frac{1}{n-1}\sum_{i=1}^{n} X_i^2$.

$$\begin{aligned} S_n &= \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{X}_n)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i^2 - 2X_i\hat{X}_n + \hat{X}_n^2\right) \\ &= \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - 2n\hat{X}_n^2 + n\hat{X}_n^2\right) = V_n - \frac{n}{n-1}\hat{X}_n^2 \end{aligned}$$

V_n may also be computed recursively

$$V_n = \frac{n-2}{n-1}V_{n-1} + \frac{1}{n-1}X_n^2 \tag{2.2}$$

By replacing the unknown variance σ with it's point estimator, we can show that Δ in Equation 2.1 can be defined as

$$\Delta_n = |\hat{X}_n - \mu| \leq \frac{\sigma}{\sqrt{n}} z_{\frac{\alpha}{2}} = \frac{\sqrt{S_n}}{\sqrt{n}} z_{\frac{\alpha}{2}} \tag{2.3}$$

for random variable $Z \sim \mathbb{N}(0,1)$ we obtain the following definition for $z_{\frac{\alpha}{2}}$

$$\mathbb{P}(Z \leq z_{\frac{\alpha}{2}}) \quad = \quad F_Z(Z_{\frac{\alpha}{2}}) \quad = \quad 1 - \frac{\alpha}{2} \tag{2.4}$$

Algorithm 3 Algorithm for computing confidence intervals

procedure $ConfInt(\alpha, v, n_{min})$

1: Compute $z_{\frac{\alpha}{2}}$
2: $n \Leftarrow 0, \hat{X}_0 \Leftarrow 0, V_0 \Leftarrow 0, S_0 \Leftarrow 0, \Delta_0 \Leftarrow 0$
3: **while** $\Delta_n > \frac{v}{2}$ or $n < n_{min}$ **do**
4: $n \Leftarrow n+1$
5: Generate X_n
6: Update $\hat{X}_n \Leftarrow \frac{n-1}{n}\hat{X}_{n-1} + \frac{1}{n}X_n$
7: Update $V_n \Leftarrow \frac{n-2}{n-1}V_{n-1} + \frac{1}{n-1}X_n^2$
8: Compute $\Delta_n \Leftarrow \frac{\sqrt{S_n}}{\sqrt{n}} z_{\frac{\alpha}{2}}$
9: **end while**
10: Return $\left[\hat{X}_n - \frac{v}{2}, \hat{X}_n + \frac{v}{2}\right]$

2.7 Common Wireless Ad Hoc Simulation Pitfalls

Recent research has sounded the alarm about a lack of credibility found in the majority of the mobile ad hoc wireless networking literature. In a recent publication titled "The Incredibles", Kurkowski et al. [40] surveyed 114 peer-reviewed papers published in MobiHoc on the basis of four areas of credibility: repeatability, bias, rigor, and sound statistics. Their findings uncovered several shocking problems that plague MANET simulation studies. Less than 15% of the simulation papers were repeatable by researchers. In 30% of the papers, the simulator used is not even identified. 64.1% made no mention of the number of simulation iterations used. Less than 7% addressed initialization bias that is introduced

by starting the system from unusual system states. In 98 out of 112 papers (88.4%) a confidence interval was not employed, and none of the simulation papers mention the PRNG used. These facts lead the authors to believe less than 15% of the MobiHoc simulation results are credible.

Todd Andel and Alec Yasinsac [3] propose several basic principles that should be followed by researchers to improve simulation credibility. In order to permit a fellow researcher to repeat experiments, the authors stress the importance of providing a detailed parameter list for all of the models. It's also important to specify the version for the commercial simulator, or make the source code available for self-developed simulators. A large part of the problem is that there is no agreed upon network simulator that has been fully validated for use in most studies. Researchers often develop their own models or use different simulation packages. Additionally, it was found that researchers often use their own assumptions and parameters. The lack of a universal simulator reduces the repeatability of experiments. Therefore it is even more important to diligently report on the versions, models, and simulation parameters used.

Andel and Yasinsac also recognized that researchers were employing overly simplistic models. As an example, Shadowing and two-ray path loss models should be used in place of elementary free space path loss models to add necessary realism. As more implementations and experimental testbeds become available, the authors suggest that researchers should tune their physical layer models and abstractions to match actual measurements. More realistic application models should also be used where appropriate. Typically the layers above the transport layer are compressed into a single layer that generates constant-bit-rate (CBR) traffic with some distribution for the frame size and generation rates. Realistic traffic

can be generated from a specific application profile or an actual trace for a given scenario.

A larger problem that damages credibility of simulation results is the lack of validation. As Kurkowski et al. [40] suggest, a large majority of simulation research lacks reference to the removal of bias, the PRNG used, and fail to employ a confidence interval. Andel et al. [3] address these problems by asserting that researchers need to address issues of randomness, and when possible validate the complete simulation against a real-world implementation. When its not possible to validate against a real implementation, a less precise comparison with analytical models and specifications may be used. Results should only be captured after the system reaches steady state, (i.e., removing transients). The authors warn researchers to only use simulation as proof of concept or for general performance trends. Takai et al. [73] deem simulation unfit to compare different protocols against one another, as they showed examples where the underlying assumptions or model parameters produced inconsistent results when comparing several routing protocols. Comparative analysis of protocols are only valid for the particular experiment and should not be presented as the common trend. To better understand the effect of underlying parameters researchers should use sensitivity analysis to a identify a chosen parameter's significance. For example, when comparing two routing protocols, the effect of node speed may be varied to illustrate the effect of mobility on several performance metrics for each protocol.

2.8 Languages & Simulators

There are many well accepted simulation languages and engines that can be readily employed to develop models. Commercial grade simulators such as NS-2 [23], CSIM [4], SSFNET [56], and OPNET [19] have a steep learning curve and are difficult to validate.

However, not all simulators are created equal. As evidence, Cavin et al. [14] observe a significant difference between NS-2, OPNET, and GloMoSim for a simple flooding protocol. The main limitation of many of the available tools are that large simulations consisting of tens of thousands of nodes have been known to scale poorly. In [53], the authors simulate 300 mobile nodes with intensive random traffic connections using NS-2 for 10 seconds of simulation time. The simulation ran 100 times slower than real time, finishing in about 1000 seconds on a modern PC.

An additional problem is that there is a lack of agreement between many simulation packages. Cavin et. al [14] compared the results of a simple and well-understood flooding protocol on Network Simulator (NS-2), OPNET, and Global Mobile Information Systems Simulation Library (GloMoSim). Their study pointed out a significant disparity between wireless simulation packages not only in absolute value, but also in general trends or behavior observed. Their results suggest that simulation alone is insufficient. Moreover, a simulator cannot be trusted until it has been validated against a real implementation or thorough comparison with analytical results.

Our work has led to the development of a wireless simulator coded in C++ and a MPLS simulator implemented in Java. We initially fabricated network models using commercial and open source simulation packages. In particular, we used CSIM for the initial wireless simulator and the Desmo-J simulation library for the initial MPLS simulator. These frameworks permitted rapid development, but at the cost of performance. Both simulators have since been re-designed to improve scalability and performance.

2.8.1 CSIM

CSIM [4] is a proprietary commercial simulator freely available for license. It is a general purpose simulator that was originally developed for hardware and processors, but more recently has been expanded to support network simulation. Implemented as a wrapper on top of the C language, the CSIM package provides a basic graphical user interface, a discrete event simulation core, and many optional packages (e.g. plotting, performance tools). The wireless module provides a framework to simulate wireless links as opposed to stationary links in the standard networking package. Wireless links are implemented as synchronous threads, called syncrons. WAIT and RESUME methods permit a node (thread) to sleep while not communicating. The state of threads abstracts away the details of the discrete event simulator so that synchronous behavior can easily be handled without explicit event creation or modification. The wireless networking models were used to implement prototype wireless models. Later a C++ simulator was developed to gain access to the event scheduler for performance reasons and make use of object oriented design (OOD).

2.8.2 Desmo-J

Desmo-J [57] is a modular, object-oriented, all-purpose discrete event simulation library written in Java. It supports common simulation entities such as stochastic distributions, queues, and data collections. A discrete event scheduler, simulation clock, and event lists are provided as well. An extensible set of classes provides basic functionality. Building models in Desmo-J is reduced to simply adding logic and functionality to the basic models. Desmo-J was originally used to simulate adaptation and preemption protocols.

3. Modeling and Simulation of Mobile Ad Hoc Wireless Networks

Wireless networks are inherently more difficult and computationally expensive to simulate than fixed wired networks. The notion of a fixed link is replaced with an error-prone broadcast channel. Bit errors in wireless networks are orders of magnitudes higher than fixed wired networks and vary with the received SINR. Channel models are used at various levels of complexity to model the path loss a signal experiences from the source to the destination. These models may take into account the terrain, obstacles, inter-nodal interference and other physical characteristics that affect radio wave propagation. Lastly, node mobility produces a rapidly changing network topology.

It is computationally infeasible to model every detail, therefore wireless models exhibit a trade off between fidelity and computational complexity. Additional features increase realism at the expense of runtime and development time. The increased runtime is due in part to the additional parameters and computations. More importantly, complex models incur a higher variance in the results. Complex models require additional simulations to obtain the same confidence level. Using a layered approach for the protocol stack, researchers are permitted to select the models at each layer that best fits the goals of the simulation study.

Ad hoc wireless networking is rapidly evolving technology that has received widespread attention from the research community. Proposal and evaluation of new wireless channel models or network protocols requires testing usually by simulation on various topologies and under realistic operating conditions. This includes a representative movement model, a range of transmission powers, offered loads, and other network parameters. Lastly, there

are several inter-protocol relationships found in wireless networks that need to be considered. As an example, Takai et al. [72] show that observed deviations in the performance comparison of several routing protocols may attributed to settings used in other layers of the protocol stack.

In a project between Drexel University and the U.S. Army, we set out to develop and test the performance of ad hoc networks where nodes employ the IEEE 802.11 MAC protocol IEEE-802.11. A packet-based discrete-event simulator (DES) was developed with several physical and data link layer models. Of primary interest is determining the appropriate level of model detail given imposed runtime constraints. By sweeping a range of values, we conducted a sensitivity analysis for various model parameters. The results of these studies provide insights into how significant a given parameter is for modeling. Finally, removing unnecessary detail or simplifying complex models can then be applied to increase performance.

3.1 Wireless Models

In the sections that follows, various models are considered at each layer of the protocol stack. Implementation in software typically assigns each layer to a class or module and follows a network layered approach. Only the protocols implemented in the C++ version of the wireless simulator are presented.

3.1.1 Movement Models

The need to model realistic node movement has spawned the development of several movement models. A detailed survey of movement models has been conducted by Camp

et al. [13]. In this survey two classes of movement models, trace and synthetic models are identified. Trace models are captures of node movement in a real network environment. Traces provide excellent results for study when taken over long periods of time. In many cases the network mobility pattern may not be obtainable for networks not yet in place, or privacy concerns may prevent such data collection. For these reasons, synthetic models are used to approximate realistic movement patterns. In following sections, several of the more popular synthetic movement models are presented.

The computational burden of the movement models are incurred when computing the current positions for nodes in the network. This involves computing the current x, y, and z coordinates in Euclidean space, given the trajectory of the mobile node and the time interval since the last update. For the simplest of cases, without considering obstacles and complex surfaces for the terrain, the distance traveled is simply the Euclidean distance. More complex models may assign intermediary points (or waypoints) in between the starting and destination coordinates as specified by a path finding algorithm. Physical layer functions make function calls to calculate the instantaneous node positions. Positions may be updated at discrete events when either transmissions occur or a destination is reached. By observing that transmission events occur on a much shorter timescale than significant movement updates, even longer durations which may encompass many transmissions before re-calculating node positions may be used. These longer update intervals are considered a model abstraction that sacrifices precision for the sake of reduced running time (See section 3.1.5).

Random Walk

Random walk is one of the earliest movement models dating back to Einsteins work on Brownian Motion in 1905 [21], and is a commonly used movement model mobile user in cellular networks. In this model, mobile nodes randomly choose their speed and destination uniformly in $[speed_{min}, speed_{max}]$ and $[0, 2\pi]$ respectively. A new speed and trajectory are chosen for a mobile node after traveling for a fixed distance d, or instead may be chosen at a time interval t. Upon reaching the boundary of the simulation space, the mobile node is reflected from the boundary with a new trajectory dependant upon the angle of incidence. Random walk can be extended to 1D, 2D, or N dimensions and may be simplified by assigning to same speed to all nodes. An important property of Random Walk is that it is a memoryless process, as past trajectories and speeds do not influence future selections.

Random Waypoint

Random waypoint is widely used to model movement in ad hoc networks [35, 54, 55]. The random waypoint model [35] is similar to the random walk model but introduces pause times in between trajectory/speed selection. Pause times are chosen uniformly between $pause_{min}$ and $pause_{max}$, and the trajectory is selected within the confines of the simulation space. One movement cycle under this scheme is characterized by the selection of a random destination, traversing to this destination with a random velocity, and then pausing for a random period of time upon reaching the destination. The simplicity of random waypoint and random walk lends itself to tractable performance characterization, but results in an unrealistic movement model, producing sudden stops and sharp turns [13]. Additionally,

the random waypoint model has been shown to skew results as it fails to achieve a steady state when the $speed_{min}$ is chosen near 0 [79]. This transient, termed *speed decay*, is characterized by a gradual decrease in the average speed for the nodes in the simulator. This problem is discussed at length in the following section. In addition, the boundary effect produced a non-uniform node density where nodes are more likely to be found near the center of the simulation arena. A host of variations to the random waypoint model have been proposed to improve realism or start in the steady state distribution [67, 55, 8, 34].

City Section

While the family of random waypoint models are desirable for mathematical simplicity, they fail to model movement of any real network. To address this issue, Markoulidakis et al. [47] introduce a much more complex set of models based on transportation theory [1]. The authors present three models appropriate for a full range of scenarios. In particular, the City Area Model traces user motion at an area zone level. This is the high level model, which specifies the distribution of transfers between area zones, the distribution of each type of node (e.g. automobile, pedestrian, etc.), and the percentage of mobile nodes that are stationary or moving. The Area Zone Model is composed of street unit models and emulates users moving on a street network. The Street Unit Model tracks user motion with an accuracy of a few meters. There are three different types of street unit models: highway, traffic light streets, and high/low priority roads for modeling different traffic patterns. Traffic density and speed may be specified using empirical or other distributions. The City Area Model adds significant realism for urban wireless scenarios. The main improvements come from detailed specification of the simulation population, and specifying a more re-

alistic simulation arena with obstacles such as buildings, along with rules conducive to modeling pedestrian and automobile traffic on city grids.

Speed Decay

Several researchers have identified the problem of speed decay in random movement models [79, 55, 80]. In the majority of movement models the velocity of nodes are chosen uniformly from $(0, V_{max}]$. One might expect that the speed of a typical node at a typical point in time have an average value of $\frac{V_{max}}{2}$. However, nodes following the random waypoint models fail to reach a steady state distribution and instead the average speed decreases as the simulation progresses. Intuitively, this phenomena is accounted by the fact that an average trip increases in duration as more and more nodes get "stuck" traveling at lower speeds. That is, at a typical point in time there are more nodes moving slowly then quickly. Nodes traveling quickly reach their destination sooner and hence choose a new speed. A side effect of boundaries is a non-uniform clustering of nodes, whereby nodes tend to be found near the center of the simulation space.

Mobility is a primary parameter that characterizes network performance. Therefore, it is imperative that a stable mobility model be employed [59, 79, 80]. We say a model is stable when it has a time-stationary distribution. One common solution is to wait a duration of time until the simulation has neared steady state before collecting data. The two primary drawbacks to this approach include the potential for a waste of a large amount of simulation time [10], and it is not clear how to identify when steady state has been reach. Another way to simply reduce the duration of the speed decay is to choose a minimum speed greater than zero, or choose speeds within a percentage of the desired average speed [13]. This

is a particularly limiting solution as it doesn't allow a variety of movement speeds for the same experiment. A more ideal solution, labeled *perfect simulation*, has been invented to remove the transient period and begin simulation drawing from the steady state distribution. Models with a minimum speed greater than zero have a stationary regime. This stationary distribution can be formulated via renewal theory [54, 55, 80, 43] or Palm calculus [41, 11].

3.1.2 Channel Models

One of the most important considerations for modeling and simulation of wireless ad hoc networks is the determination of whether a pair of mobile radios can communicate. A link budget is used to describe the physical relationship between the transmitter and receiver. The maximum allowable path loss expressed below, takes into account the transmitter power P_t, the transmitter and receiver's antenna gains G_t and G_r respectively, and lastly the receiver sensitivity (see Equation 3.2).

$$L_{path} = P_t + G_t + G_r - S_r \tag{3.1}$$

There are several channel models used at the physical layer of the OSI reference model to characterize the performance at the receiver's (S_r). A useful method of presenting these models is through the three main factors that effect signal propagation: path loss attenuation, slow-fading (shadowing), and fast-fading (Rayleigh fading) [71]. These factors vary in significance according to the distance between transmitting and receiving radios and restrict the achievable data rates, range, and reliability of the channel. The extent to which these factors affect the wireless signal depends upon the environment as well as mobility

between the transmitter and receiver. For distances of more than a couple of kilometers, attenuation is the most significant factor, as the received signal power tends to fall off steadily. For distances between 1-2 kilometers, the signal is significantly affected by slow fading. If distances are on the order of several hundred meters, the signal is noticeably affected by fast fading. Several of the factors that influence the path loss, especially the terrain, cannot be used to characterize all transmissions; hence several models are required to describe the variety of transmission environments [52]. In what follows, a description of some of the popular path loss models, preceded with an introduction to wireless effects modeling is presented.

The performance at the receiver (S_r) in equation 3.1 is an important metric which takes into account the effective temperature noise T_0, the noise figure F, the Boltzman's constant k, the losses in the receiver hardware, the modulated symbol rate R_c, and the modulated signal-to-noise ratio (SNR) $\frac{E_c}{N_0}$. We express the receiver sensitivity as

$$S_r = 10\log_{10}\left[L_r k T_0 F R_c \left(\frac{E_c}{N_0}\right)_{min}\right](dB) \tag{3.2}$$

Where the modulated signal to noise ratio is the minimum required for successful communication. Various QoS metrics are commonly used to characterize the performance of wireless networks. Of these outage probability and bit error rate (BER) are most relevant to studies of the physical layer. The outage probability expressed as

$$\mathbb{P}_{outage} = \mathbb{P}\left[\frac{E_c}{N_0} < \beta\right] \tag{3.3}$$

is the probability that the SNR at the receiver falls below some threshold β. In other

words, a transmission is considered successful if the the ratio of the received signal power, (E_c) and the noise at the receiver, (N_0) are in excess of a device specific hardware specific threshold parameter.

If we are interested in computing the area outage probability, then for a circular region (without statistical variations) with radius R, the area outage probability is

$$\mathbb{P}_{outage} = \frac{1}{\pi R^2} \int_0^r \mathbb{P}[\frac{E_c}{N_0} < \beta] 2\pi r dr \tag{3.4}$$

Up until this point we have not considered the affects of interference. Co-channel interference occurs when multiple devices within close proximity to one another transmit at the same time and frequency. Interference is a major limiting factor on the performance of wireless networks. Numerous modulation schemes and higher layer protocols have been proposed to reduce the level of interference, including Time Division Multiple Access (TDMA) [22] and Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [29] to name a few. To account for the addition of co-channel interference the minimum SNR in Equation 3.2 may be replaced with the Signal to Interference and Noise Ratio (SINR)

$$SINR = \frac{P_r}{N_0 + I} \tag{3.5}$$

which is simply the ratio of the received signal power, P_r to the noise power, N_0 plus the aggregate interference, I. The Noise power adds to the distortion of the received signal and is contributed by ambient noise present in the environment. The interference term, I represents the accumulated received power from N other interfering transmitters.

$$I = \sum_{i=1}^{N} P_{r,i}(dB) \tag{3.6}$$

The second limiting factor of performance is the bit error rate (BER). The BER is a result of fast fading signals that introduce smaller but more rapid fluctuations in the signal, that lead to single bit errors. Compared to wired networks with typical bit error rates of 10^{-9} [16], the BER rate is much higher for wireless networks ($10^{-5} - 10^{-3}$) [78]. Error correction can be employed to solves bit errors of sufficiently low levels without making the channel unusable, but this metric can provide a good measure of channel quality [27]. The BER is function of the SNR, but dependant upon the modulation scheme employed. Equation (3.7) provides a sample BER model for a specific modulation scheme, Binary Phase-Shift Keying (BPSK) [63]. The BER under AWGN is

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.7}$$

For binary modulation systems, such as BPSK, this is also the symbol error rate. The packet error probability can be computed from the BER by taking into account any error correction codes that are employed. For the case where error correction is not used in hardware, this value would be 0. The number of errors N incurred during transmission is modeled as a binomial random variable $N \sim B(n, p)$ with n bits and a BER of p.

Free Space Attenuation

The free space attenuation model is a simple line of sight (LOS) path loss model that simulates a signal's decay from the transmitter to the receiver. This model does not include

any reflection or diffraction effects. Moreover the signal's decay rate is assumed to remain constant between the two radios. The path loss for free space is

$$L_p(d) = 10\log_{10}\left[\frac{P_t}{P_r}\right] = -10\log_{10}\left[\frac{G_t G_r \lambda^2}{(4\pi)^2 d^2}\right](dB) \tag{3.8}$$

where the wavelength of the signal is denoted by λ, and d is the distance between the transmitter and receiver. This model is accurate only where there is a LOS component and hence is not appropriate for use in terrestrial mobile ad hoc networks. However, the simplicity of the free space attenuation model makes it a popular choice for analytical analysis.

Path Loss Attenuation

Building upon the simple free space attenuation model, the path loss attenuation model includes the affects of reflection, refraction, and scattering. All of which, allow a signal to reach the destination through multiple paths. The simulation environment is characterized by the density of objects or scatterers which dictates selection of the path loss exponent α. Some typical values for α are provided in table 3.1. The underlying equation driving this model is

$$L_p(d) = L_p(d_0) + 10\alpha\log_{10}\left(\frac{d}{d_0}\right)(dB) \tag{3.9}$$

where $L_p(d_0)$ is the free space path loss at a reference distance. This reference distance is usually measured empirically for specific environments.

There also exists a modified attenuation model that accounts for a signal's changing

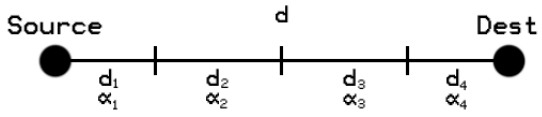

Figure 3.1: Path loss across multiple terrain types

decay rates. The type of environment may not be constant over the distance traversed by the signal, leading to multiple values of the signal's decay rate for each differing region [51]. The modified path loss equation becomes

$$L_p(d_0) = 10\log_{10} L_p d^{-\alpha_0} \prod_{i=1}^{N} \left(1 + \frac{d}{d_i}\right)^{-(\alpha_i - \alpha_{i-1})} (dB) \tag{3.10}$$

where α_i is the path loss exponent in the i^{th} segment, and d_i is the distance from the transmitter to the start of the i^{th} segment, when segments are numbered 0 to N. d_0 denotes the reference distance which must be greater or equal to a unit in length. Figure 3.1 illustrates the assignment of d_i and αi when a transmission traverses three terrain types between a source and destination radio. Implementation in the simulator requires a terrain database to maintain the path loss exponent for each region of simulation arena. The path loss attenuation model is a major improvement upon the free space model, yet the model is still feasible for fast simulation. This makes the path loss attenuation model the most popular channel model for wireless ad hoc networks today.

Environment	Path Loss Exponent, α
Free Space	2
Urban macro cell	3.7-4.3
Shadowed urban	3.5
In building, (same floor)	1.6-3.5
In building, (different floor)	2-6
Home	3

Table 3.1: Path loss exponents for different environments [63]

Log-normal Shadowing

Log-normal fading, also called shadowing or slow fading, denotes a random attenuation caused by large objects affecting the signal. Since the size, shape, and location of objects are not known, a statistical characterization of the environment is used. Fluctuations in power due to shadowing are represented by a zero mean Gaussian random variable X_σ, that can be added to the path loss equations for any of the attenuation models. The standard deviation σ of the random variable produces the variability of the received power seen about the mean. This value reflects the terrain, and is determined from various field measurements (see Table 3.2).

$$L_p(d) = L_p(d) + X_\sigma(dB) \tag{3.11}$$

The accuracy of log normal shadowing is dependent upon identifying the appropriate shadowing parameter for each region of the simulated environment. When the simulation environment contains large objects, the channel model may be improved with the addition of a relatively inexpensive computation.

Environment	Standard Deviation, σ
Urban macro cell	5-12
Shadowed urban	6.5-8.2
In building, (same floor)	5.2-7.0
In building, (different floor)	9.6-14

Table 3.2: Standard deviation for different environments [63]

3.1.3 MAC Protocols

In wireless networks nodes communicate via a shared broadcast channel. Media access control (MAC) protocols at the data link layer are necessary to arbitrate access to the shared medium both fairly and efficiently. The characteristics of wireless networks are completely different from wired medium and introduce new issues for modeling such as node mobility, an error-prone broadcast channel, hidden and exposed terminals, and power constraints. A new set of protocols are necessary to address these issues.

IEEE 802.11 Protocol

IEEE 802.11 is the de facto wireless LAN standard defining the specification for the physical and media access control (MAC) layers of the protocol stack. The primary function of the MAC layer is to arbitrate transmission requests for nodes in a given area that would otherwise interfere with one another and prevent communication. Under 802.11, nodes can communicate in either infrastructure mode called the point coordination function (PCF) or in distributed contention-based mode named the distributed coordination function (DCF). DCF is the primary access method for 802.11 and is based on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

Under CSMA/CA, before a radio may begin transmission of a data frame, it must sense

the channel to assure another radio doesn't interfere. After the channel has been sensed idle for a time period of DIFS (DCF Inter-frame Space) the station may proceed with the transmission. Due to the high bit error rates and the inability to detect collisions in wireless networks, the IEEE 802.11 DCF adopts collision avoidance (CA) instead of conventional methods that merely respond to collisions after they are detected. Collision avoidance is implemented by sensing the channel for a "random" period of time before transmitting a frame.

A successful transmission must be acknowledged by the receiver with an ACK frame. This is necessary because the transmitter is unable to discern whether the frame was received successfully or not simply by listening to the channel. A common problem in wireless networks termed the *hidden terminal problem* arises when another frame, sent from a node not in range of the transmitter, collides with a frame at the receiver. Before sending an ACK frame a radio must wait a time period of SIFS (Short Inter-Frame Space). If an acknowledgement is not received within a time-out period, the frame is assumed to have failed and re-transmission is scheduled. Each station in the network maintains what is referred to as a network allocation vector (NAV) to refrain from transmitting a frame until this timer reaches zero.

The DCF provides an optional contention method referred to as a virtual carrier sense mechanism where a handshaking procedure is used to reserve the medium prior to to data transmission. Virtual carrier sensing is used to reduce contention caused by hidden terminals. Figure 3.2 illustrates a data frame transfer using virtual carrier sensing. A small request to send (RTS) frame, containing the expected duration of the transfer, is sent by the transmitter that has a frame to send. The intended receiver then may respond with a clear to

send (CTS) that also indicates the expected duration of the transfer. All radios in range of the RTS and CTS update their NAV timers and refrain from transmitting until the ongoing transfer completes.

If the medium is sensed to be busy during a time interval while waiting to transmit a data frame or an ACK, the transmitter must defer access to the medium until the end of the ongoing transmission. A binary exponential back-off algorithm is used to select a random time interval. A back-off timer is random integer drawn from a uniform distribution over the interval $[0, CW - 1]$, where CW (Collision Window) is an integer within the range of CW_{min} and CW_{max}. The back-off timer is the number of slots the transmitter must wait before attempting to seize the channel. The value is decremented by one for each idle slot detected. The timer suspends whenever the channel is sensed busy and is resumed after the channel has been idle for a period of DIFS. When the timer reaches zero, but an attempted transmission fails, the value of the congestion window is doubled upon each collision (i.e. $CW_{new} = CW_{old} * 2 - 1$) until it reaches CW_{max}. This method is used to dynamically adjust to the congestion level of the network and reduce collisions.

The 802.11 protocol can be thought of as a state machine and implemented as such. Faithful implementations use three control frames per data frame or roughly three times the number of events. Moreover, maintenance of state information consumes several variables per station.

3.1.4 Problem Statement

Meaningful simulation of ad hoc wireless networks is a time-intensive task. Reducing the execution time is crucial for studying networks where the desired scale can range from

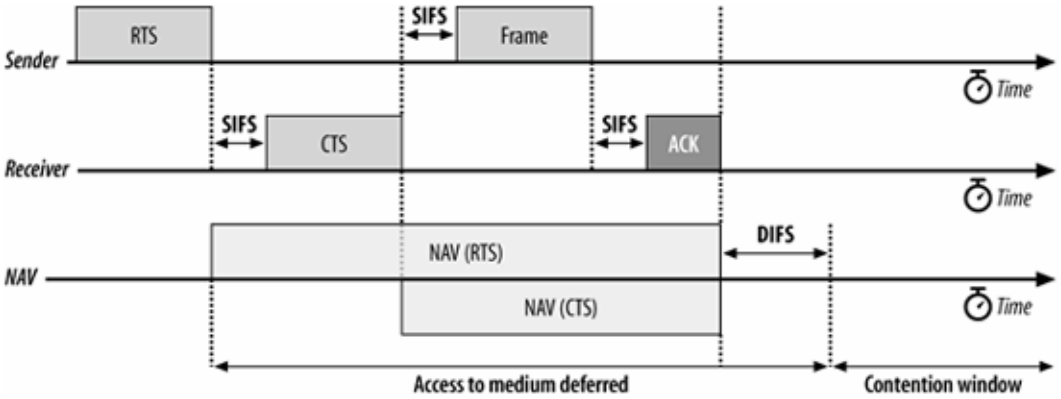

Figure 3.2: Basic access method of 802.11 MAC protocol [25]

a few hundred to thousands of radios. For simulation of large ad hoc networks, the most computation-intensive tasks are computing interference and determining which receivers are in range of a transmitter. Both cases scale poorly as $O(N^2)$ physical layer calculations are required for a wireless system of N nodes. The goal of our research is to develop abstractions for simulation of data link layer protocols in wireless ad hoc networks.

For our project, we began with a high fidelity 802.11 implementation, then looked for techniques to reduce either the number of events or the complexity of the protocol. We set out to develop a simulator containing several abstraction models of varying complexity for the 802.11 model. The original simulator was built in CSIM, but was later ported to C++ for modularity and performance gains. Modular design lends itself naturally to model selection. We model the protocol stack, whereby each layer contains several models. Depending on the running time constraints or accuracy level required, our multi-fidelity simulator can use the most suitable model. Our work does not consider algorithms for model selection, but rather focuses on the relevant affect of simulation parameters on common wireless protocol performance metrics.

3.1.5 Related Work

One of the earliest studies of protocol abstraction is the work of Bajaj, et al. on the VINT project [5]. The authors present several abstractions they developed for the NS-2 simulator. Centralized routing is one abstraction they present that reduces the computational cost and memory usage of distributed routing algorithms. This replaces routing messages with a centralized computation, and removes redundant state information at the cost of slightly different routes. A second abstraction replaces hop-by-hop packet flow with session-level packet forwarding [32]. The cost of this abstraction is the loss of queuing dynamics as propagation delays are precomputed. Lastly, algorithmic routing may be used in place of shortest-path routing algorithms such as Dijkstra's all pairs shortest path routing or Hierarchical routing algorithms. Algorithmic routing reduces the memory requirements from $O(n^2)$ for Dijkstra's algorithm and $O(n \log n)$ for Hierarchical routing to just $O(n)$. Dijkstra's algorithm runs in $O(n^3)$ and Hierarchical routing runs in $O(n\sqrt[3]{n})$ while algorithmic routing only has an $O(n)$ running time complexity. Huang and Heidemann [33] show that algorithmic routing has only about a 10% difference in the set of shortest-path routes.

Simulation of wireless networks has recently drawn considerable attention in the area of abstraction. Complex channel models and a constantly changing topology increase the complexity dramatically over that of wired simulators. Heidemann et al. [31] and Cavin et al. [14] discuss issues relating to the accuracy of mobile ad hoc network simulators. Both show how some metrics can vary rather drastically with low level model detail, while other higher layer metrics are relatively insensitive. Blum et al. [30] modify GloMoSim to support two modes of operation. At first the simulator runs in a high fidelity mode, and collects statistics on the RTS/CTS/ACK success rates and propagation delays for the

MACA protocol [37]. After a confidence level is reached, the simulator switches to an abstraction mode, whereby the data link and physical layers are replaced by probabilistically determining packet success and delay from the empirical distribution. The simulator may then switch back to the high fidelity mode after a set duration. Depending on the parameter selection it was shown that a great deal of running time could be saved with minimal deviance in the performance of the metric of interest. In this case, the authors were interested in comparing the end-to-end delay performance metric. Extending upon this work Liu, et al. [45] develop a simpler model of 802.11 MAC layer that reduces the number of events and eliminates physical layer calculations altogether. Instead they use a queue to maintain all frames awaiting transmission at the MAC layer, messages that arrive to a full queue will be dropped. The end-to-end delay and the time the queue is free to send another message are a function of the channel state. Under the channel model for this work, a packet is considered successful if the utilization of the channel utilization is below 75% capacity. The results showed that such a simplification had significant differences in the end-to-end delay and throughput for smaller networks, but as the network size increased agreement improved.

There are several proposals to improve the $O(n^2)$ running time necessary to calculate the inter-nodal interference in wireless networks. Perrone and Nicol [60] apply the Barnes-Hut (N-body) algorithm [7] to the problem by observing that the gravitational pull between objects in astrophysical models is similar to signal attenuation models of wireless models in that both decay polynomially with distance. Intuitively, nearby radios are most affected by signal interference whereas radio's farther away have a negligible affect on the received signal. Their results demonstrated a marked decrease in the number of pairwise calculations

required compared to the brute force approach. Naoumov and Gross [53] improve upon NS-2 by introducing a 3D array of pointers to nodes which are ordered by position. The simulation area is then divided into cells. The search space to compute the inter-nodal interference power is effectively reduced to a geographic subset of the simulation area as opposed to considering all nodes. This technique exploits the probability that nodes will be spread throughout the simulation area and not all contained within the same cell.

3.1.6 Data Link Layer Protocol Abstraction for Mobile Ad Hoc Networks

Our abstraction models are closely related to the approach of Naoumov and Gross [53], and can be seen as complementary to their solution. Their approach focuses on reducing the number of potential interferer's while our research looks to reduce the number of potential receivers for Multiple Access, Collision Avoidance for Wireless (MACAW) protocols such as IEEE 802.11. In the following sections, we present three abstractions that improve scalability and decrease the running time of an IEEE 802.11 wireless model (described in section [51]) while achieving comparable results to an industry standard OPNET model. Given the running time constraints or desired granularity, a multi-fidelity simulator could make use of an algorithm or heuristic to select from the full-fledged implementation or one of these abstraction models. Physical carrier sensing (via beaconing) is not simulated, but virtual carrier sensing is accomplished by the RTS/CTS mechanism.

Modeling and simulation of the full IEEE 802.11 model adds significant overhead to the simulator. In addition to physical effects, the data link layer must schedule each data frame reception which can spawn three or more discrete events. The first event occurs when stations determines it has a data frame to transmit. A request-to-send frame is first

scheduled, if this is successfully received a clear-to-send frame must be scheduled. Next the data frame is scheduled and lastly the ACK frame is scheduled. In a discrete event simulator this translates into three added events for each data frame assuming all frames are received on the first attempt. Failures of one of any of those frames (RTS-CTS-Data-ACK) result in retransmissions that increase the expected number of events per packet above the minimum of four events. The amortized cost to sort the event-queue is $O(\log n)$. In addition to the size of the event queue, another scalability issue is the number of physical layer calculations also increases by a minimum factor of four.

Model #1: Event-Compression

The event-compression model is our most faithful abstraction model. In recognition that the RTS, CTS, and ACK add three times the number of events per data packet transfer, the event-compression model uses what we call a collapsed RTS/CTS and an implicit acknowledgement. The data frames are assumed to be much larger compared to the control frames and consume the bulk of the transmission time. The RTS and CTS are transmitted instantaneously, and are not modeled by separate events. Acknowledgements are implied by simply notifying the transmitter of the status of the frame. Moreover, control frames do not interfere with data packets.

When a station's NAV indicates the medium is free, the next DATA frame is taken from the buffer and the reach of the RTS frame is considered for all other possible radios in the network. All stations within range update their NAV timers accordingly. Similarly, if the intended receiver of the DATA frame is within the list of stations affected by the RTS, then the reach of CTS is calculated. If the CTS is successfully received by the original

transmitter the DATA frame is transmitted. Note that an event is not explicitly scheduled for frame reception. Reception of packets are marked as successful or not in a future RTS event after the transmission duration has past. The total savings of this model is a three fold reduction in the number of events scheduled for each data transmission.

Model #2: Neighbor List

While reducing the number of events improves performance, its important to reduce the computation required per event. The second abstraction modifies the method in which the reach of control frames are calculated. Specifically, each node maintains a list of 'reachable' nodes or neighbors which are updated periodically. The set of reachable nodes are within the threshold distance (d_{max}) required to successfully receive an interference-free transmission. The 'neighbor distance' d_{max} from the receiver is determined by the free space attenuation as expressed in equation 3.8. P_r assumes the value of the receiver sensitivity to produce the threshold distance for communication.

Let

$$\pi = \{\overrightarrow{x_i}\} \tag{3.12}$$

be the location of the mobile nodes at some point in time. Then $\pi(B)$ is the set of nodes in A that lie in area B, and

$$\mathbb{E}[|\pi(B)|] = N\frac{|B|}{|A|} \quad N\frac{\pi d_{max}^2}{A} \tag{3.13}$$

When $B = b(0, d_{max})$ then $|B| = \pi d_{max}^2$ is the average number of 'reachable' neighbors,

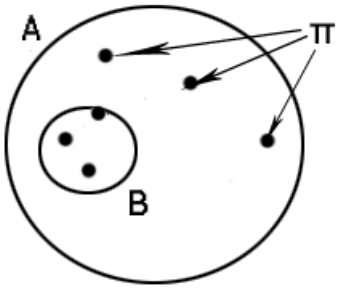

Figure 3.3: The simulation area and the neighbor list

for a network consisting of N nodes. Figure 3.3 illustrates the simulation space (A) with a subset of potential receivers (B) for a a given transmitter.

The update interval for the neighbor lists may be assigned as a function of node velocity v:

$$t_{int} = \frac{\frac{1}{2}d_{max}}{|v|}(seconds) \tag{3.14}$$

t_{int} is the the time it takes a node to travel half the neighbor distance. For a transmitter centered at d in Figure 3.4, the set of receivers are all those neighboring nodes which are enclosed by the radius d_{max}. The value of the update interval, t_{int} is chosen as a heuristic for when it is likely that the set of receivers for the transmitter has changed.

The motivation for using the neighbor list stems from two observations. First, only a potentially small fraction of all the nodes are likely to be able to detect a transmission. These nodes lie within a disk of radius d_{max} centered at the transmitter. In the worst case which is highly unlikely, all nodes are in range of each other and there is no improvement. It is more efficient to restrict the set of nodes to only those that could be affected. The second

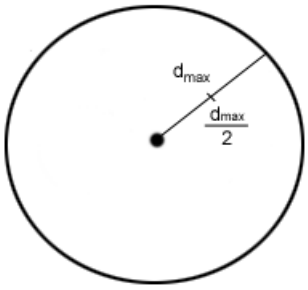

Figure 3.4: The neighbor list coverage area

observation is that the topology remains effectively constant for appropriately chosen time intervals. In other words, the positions of nodes typically change at a much lower time scale than frame transmissions.

Calculation of the neighbor-list and the update interval are proposed heuristics for reducing the number of physical layer calculations. Under this model, a transmission of a data frame invokes an RTS and possibly a CTS in a single event. Rather than checking every node, physical layer calculations are only required for each neighbor in the circular area specified in 3.13. The benefits of this abstraction are directly proportional to the update interval which in turn is a function of the transmit power as well as the speed of the nodes.

Model #3: Simplified Neighbor List

The third abstraction simplifies the transmission of control frames even further. Building upon the previous model, a neighbor list is maintained for each node in the network. The calculation for reception of the RTS or CTS frames differs in that only the intended receiver is considered. All other stations in the neighbor list are assumed to have received

the control frame successfully. For one RTS or CTS frame, physical layer calculations are only required for the sender-receiver pair. This returns an even bigger savings over the Event-Compression abstraction. Because of the single calculation for each the RTS and CTS control frames, there is less of a dependency on the transmission power.

3.1.7 Results

A packet based discrete event simulator was developed for the purposes of this study. Work thus far has produced models for the physical, data link, and applications layers. Nodes are confined by a simulation space of one square kilometer, and movement is governed by the random waypoint model without pauses. The application layer consists of a basic source which generates packets with an inter-packet arrival time modeled as a Poisson process. Packet sizes are chosen from an exponential distribution with a mean of 128 bytes. Destination nodes are chosen uniformly from the set of all nodes. Transmission success is determined using an attenuation model with a path loss constant of $\alpha = 4$. The bit error rate (BER) is evaluated for a given packet assuming a differential phase shift keying modulation (DPSK) scheme. Error correction is not supported, therefore the probability of successfully receiving a frame is the likelihood that there are no bit errors.

The simulator maintains a list called the packet queue that contains all outstanding transmissions in the network. When a new transmission begins, it is added to the packet queue, and outdated transmissions are removed. Whenever a transmission is added, the BER is recalculated for every transmission on the list to reflect the new interference conditions. The BER is then used to predict the number of errors over the length of that transmission. If any errors are predicted, the transmission attempt corresponding to the

frame is marked as failed. Lastly, the transmission times do not include propagation delay.

In this study, we compare the proposed 802.11 abstraction models to a faithful implementation in OPNET. OPNET's physical layer calculations are evaluated differently than our simple abstraction model. However, the best attempt was made to match parameters with the default values in OPNET modeler. Two metrics were evaluated in this study, goodput ratio and end-to-end (ETE) delay. The goodput ratio is defined as the fraction of successful data rate transmitted over the attempted traffic rate, or offered load. Goodput, in kbps, is the rate that bits are pushed up from the data link layer at the receiver and is dependant upon the packet error probability. Offered load, also measured in kbps, is the rate that bits created at the application layer. ETE delay measures the time it takes for a single-hop transmission to travel between the source and destination node. This includes, queuing delays, medium contention, and transmission delays.

We begin by scaling the transmission power to observe the affect it has on the goodput ratio. With a fixed packet generation rate of 1.6 seconds per node, and a network size of 30 nodes, we scaled the transmission power from 0.125 mW to 1024 mW. Figure 3.5 illustrates that all of the models demonstrate the same trend, including the OPNET validation model. The concave growth in the goodput suggests that frames are more likely to reach their intended destination as the transmission power increases. Figure 3.6 reveals negligible effects of network size on the goodput ratio for model #1 (the Event-Compression Model). This is representative of the other models as well. The low value for the goodput ratio is attributed two factors. Intended receivers at the application layer were chosen uniformly from set of all nodes which is likely to select receivers outside of the transmission radius. Second, the lack of error correction codes inflates the probability that frame is dropped to

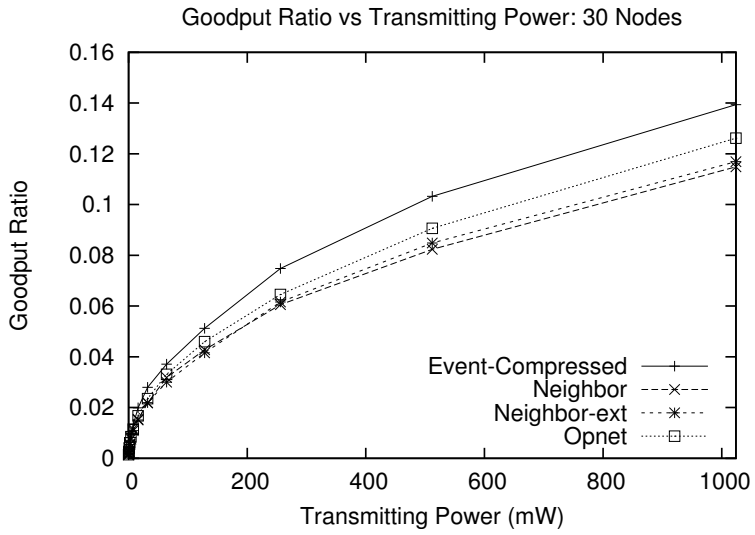

Figure 3.5: Goodput ratio vs transmit power

equal the probability that a bit is received in error.

Figure 3.7 depicts the ETE delay dependency on the normalized offered load. For a fixed transmit power of 100 mW, the offered load was scaled by modulating the inter-arrival time for packets (packet inter-generation mean = $\{3.2, 1.6, 0.8, 0.4, 0.2\}$ seconds). In the three abstracted models, ETE delay noticeably increased with the offered load per node. This is expected as higher offered loads increase queuing delays and contention. The ETE delay for the abstraction models elicit a different trend than the abstraction models, yet the results are on the same magnitude. The results suggest that the abstraction models may not be suitable when delay is a metric of interest. The general trend for how the ETE delay behaves as a function of the network size is shown in Figure 3.8. Only the Event-Compression Model is shown, but it is representative of all of the models.

Lastly, Figure 3.9 displays how the runtime scaled with the network size. The transmis-

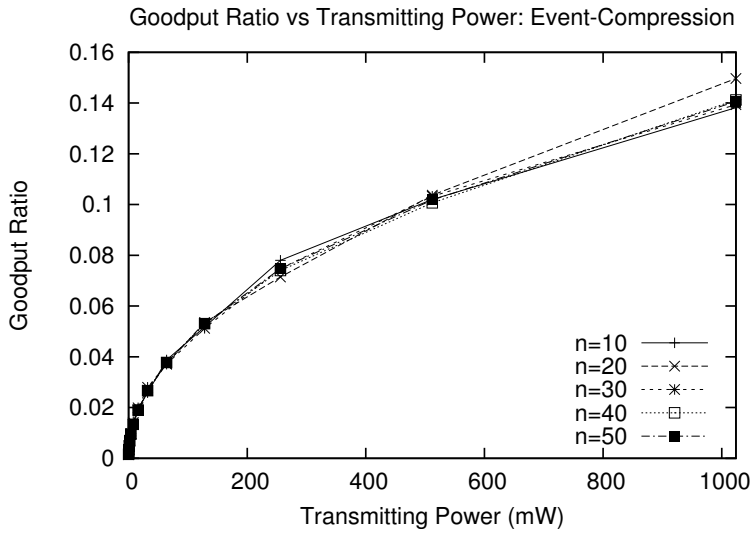

Figure 3.6: Network size vs goodput ratio

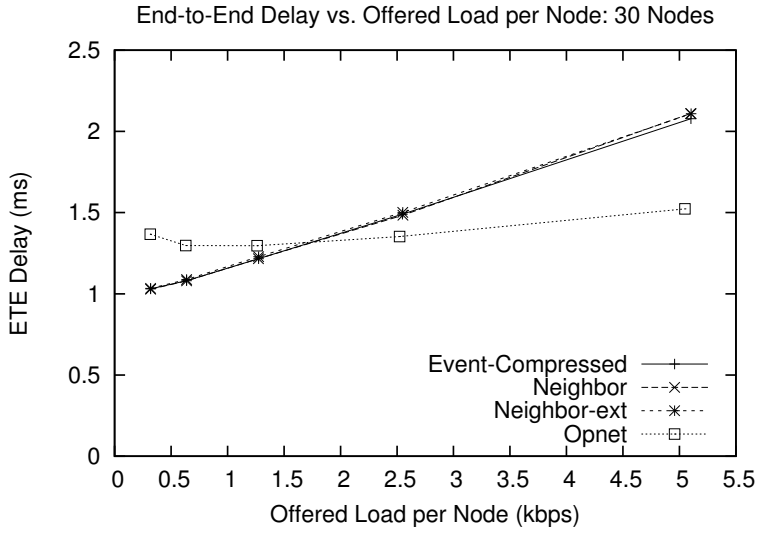

Figure 3.7: End-to-end delay vs offered load

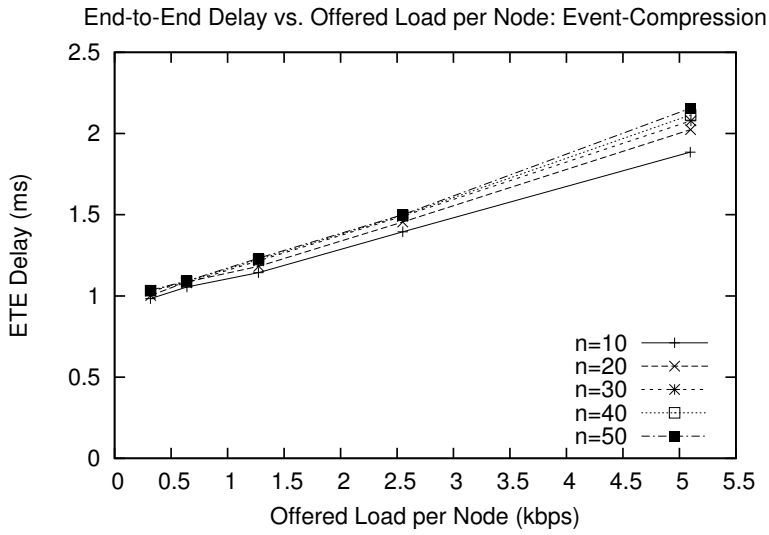

Figure 3.8: End-to-end delay vs network size

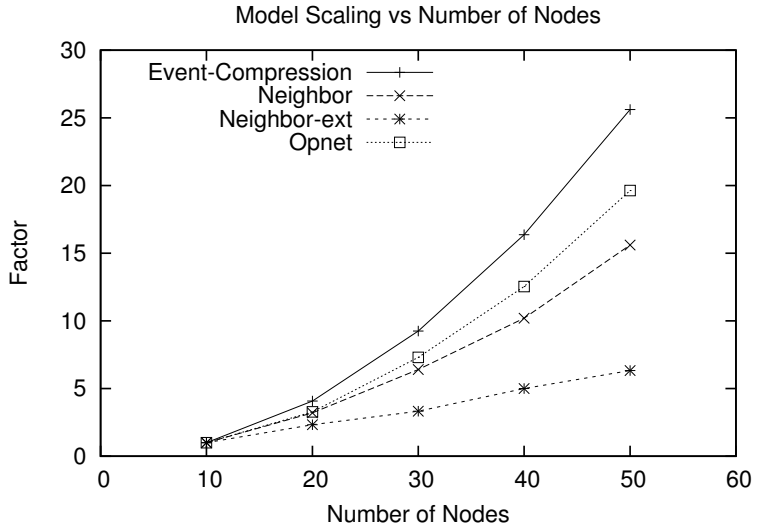

Figure 3.9: Model runtime vs network size

sion power was set to 100 mW and the packet inter-generation time was set to 3.2 seconds. The running times for the models are normalized by their respective 10-node run times to allow relative runtime growth factor benchmarking between the OPNET and AWSIM simulators which used a dissimilar runtime environment and model representation. The OPNET model scaled better than the Event-Compression model, yet its absolute runtime was about twice as high. A progression can be observed from the Event-Compression model with quadratic scaling, down to the Simplified Neighbor List, which elicits linear scaling in the network size.

In summary, the goodput ratio for the three abstraction models have been validated using OPNET. There is however a discrepancy in the end-to-end delay measurements. Each abstraction presented improves upon the rate at which the running time increases in the network size. These results demonstrate a significant improvement, reducing the runtime growth from quadratic to linear.

3.1.8 Performance Programming

To model wireless networks, initially Lockheed Martin's CSIM [4] discrete event simulator was chosen. CSIM provides a ready framework for wireless networking, including models for wireless radios, and is composed of a library with the basic simulation building blocks in C. One of the building blocks is an element called a synchron. Synchrons were used to model the synchronous communication between radios which are modeled as threads. Radios start out initially in a "sleep state"; waiting to receive a transmission. A packet transfer is accomplished when another radio wakes up a nearby radio in a sleep state, sending a pointer to the destination radio variable which represents the packet. Delays are

then used to simulate the duration a station is in the send or receive state.

Initially, multiple syncrons were used at several layers of the protocol stack. However, we were able to reduce the number of threads down to one for the movement and another for communication. A migration from CSIM to C++ was then undertaken to gain access to the event scheduler, in addition to taking advantage of an object oriented design. A simple single-process discrete-event simulator was proposed to alleviate the memory and per-node overhead of using threads to maintain state. However, additional approaches such as worker threads and threading pools could be used to further reduce memory consumption and improve upon baseline performance.

Figure 3.10 shows a constant time improvement for the C++ simulator. There are several differences in the implementation of both simulators, and therefore performance results are considered crude at best. The two scenario's were normalized with a stopping criterion of 20,000 transmission attempts. The CSIM version simulated the full 802.11 model, while the closest C++ model used event compression. The relevant parameters for the simulation trials are provided in Table 3.3.

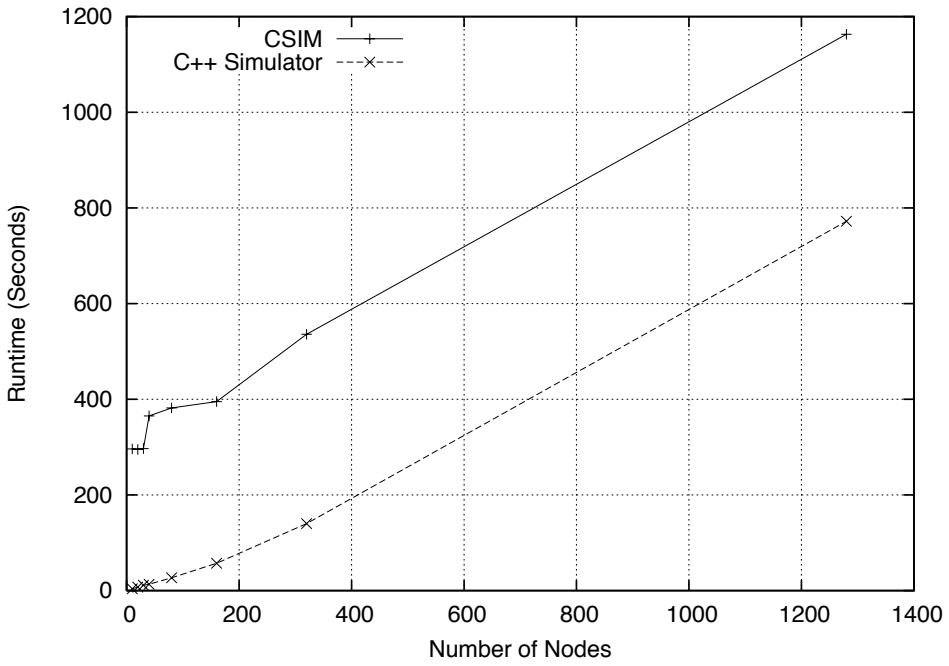

Figure 3.10: Running time vs network size

Parameter	**CSIM Version**	**C++ Version**
Simulation Area	$1km^2$	$1km^2$
Movement Model	Random Waypoint	Random Waypoint
Channel Model	Rayleigh	Rayleigh
Datalink Model	IEEE 802.11	IEEE 802.11 Event Compression
Frame TX Attempt	4	4
802.11b Timeslot	20μs	20μs
802.11b SIFS	10μs	10μs
802.11b DIFS	50μs	50μs
802.11b CWMIN	31	31
802.11b CWMAX	1023	1023
802.11b Max Frame Size	1500 bytes	1500 bytes
Threshold Power	$31.0 * 10^{-9}$ Watts	$31.0 * 10^{-9}$ Watts
Path Loss Exponent	4.0	4.0
Shadowing Variance	8.0	8.0
Stopping Criterion	10,000 arrivals	10,000 arrivals

Table 3.3: Variables used for simulation runtime comparison between CSIM and C++ versions.

4. Traffic Engineering for the Next Generation Internet

High bandwidth applications such as streaming media are increasing in popularity and have the potential to dominate future Internet traffic consumption. Forester Research [38, 28] estimate that 46 millions homes in the United States alone will have broadband access. In another study conducted by Vision Consulting [20], there were 60 million people watching streaming media worldwide. In addition, the Vision group found that over 6000 hours of streaming media content is being published per week. As of recent, cable companies and telecommunications companies have begun offering streaming television and voice over IP (voIP) services. To handle this increase in demand for real-time traffic, bandwidth scaling protocols will be highly attractive for Internet service providers (ISPs).

In this section, we present two congestion response mechanisms for use in Multiprotocol Label Switching (MPLS) networks that seek to maximize the performance of real-time applications. MPLS network backbones combine data link and network layer information to simplify and improve IP-packet exchange. The MPLS protocol may be leveraged by ISP's within the confines of their network, also referred to as an autonomous system (AS), to re-route traffic around congestion and link failures. Edge routers assign a label to traffic connections based on service classes. Using these labels, providers can employ a technique known as preemption to make service class-based routing decisions. This approach aligns high priority traffic on the shortest or least cost routes by displacing low priority traffic. The criteria used to select which streams are moved is referred to as the preemption algorithm.

While preemption yields a degree of flexibility to service providers, it is unable to ef-

fectively throttle congestion when the network is over-provisioned. There are two general approaches to this problem, congestion control and admission control. An admission control algorithm rejects additional connections that would exceed the available capacity of a link on the network. However admission control is not supported by the adopted best-effort Internet protocol standard. An alternative we have elected to study is that of rate adaptation. Adaptation addresses the fundamental problem that clients in the network need to be sensitive to the congestion around them. For the same reason a car is not permitted to drive at the desired speed limit during rush hour congestion, streaming media may not always transmit at the desired bit-rate. The algorithm that determines when and who should increase or decrease their stream rate is known as the adaptation algorithm. As congestion increases, stream rates are reduced to make room for additional traffic which result in an increase of capacity.

4.1 Rate Adaptation for Multimedia Streams

High bandwidth, streaming media applications are pervading the Internet. Rate adaptation is a congestion response technique well suited for inelastic traffic (e.g., streaming media applications). Akin to the congestion response mechanism of TCP, using adaptation, UDP streaming media clients may toggle subscriptions (i.e., instantaneous bit-rate) in order to adjust to perceived congestion levels. Consider a media file encoded at two bit rates of high and low quality. Not all clients will be able to subscribe to the high quality stream due to bandwidth limitations or congestion. Using adaptation clients may reduce their subscription to the low quality stream in times of congestion, but are also free to switch to the high quality bit rate when traffic levels subside. An algorithm for distributed adaptation,

Receiver Driven Layered Multicast (RLM), was originally proposed by McCanne [50]. Based on the premise that no one rate is right for all clients, RLM takes a client based approach to accommodate heterogeneity. The client and server may share information about the transfer, but the burden of rate selection is placed on the client and not the source.

Rate adaptive receiver-driven protocols have been widely accepted, and utilized in many streaming media software applications [66, 64, 6]. The Congestion Manager framework (CM) is an IETF standard adopting most of the key concepts of the RML protocol. This framework is used in several streaming media applications. The RLM algorithm proposed by McCanne provides a prescription for selecting the client's subscription level based on the level of congestion experienced in the network. When a client first joins a multicast group, it starts out receiving only the base layer of the lowest quality. After a random period of time, the client tests the bandwidth of the connection by subscribing to the next layer of higher quality. If the client experiences excessive packet loss it drops that layer. The time between join experiments increases after each failure using an exponential back-off algorithm. Clients wait an exponential period of time based on the number of failed packets that occur while attempting to subscribe or "up-shift" to the next subscription level. If, after a certain period of time the client has not experienced loss, the algorithm will repeat again until it reaches the highest subscription layer.

Figure 4.1 illustrates the exponential backoff strategy for a sample realization for an RLM client that receives four layers of increasing quality. The first series of join experiments are successful until the client reaches the maximum subscription level. Upon subscribing to the maximum layer, the client perceives congestion (C), reduces it's subscription rate, and must wait an exponentially longer period of time before re-attempting

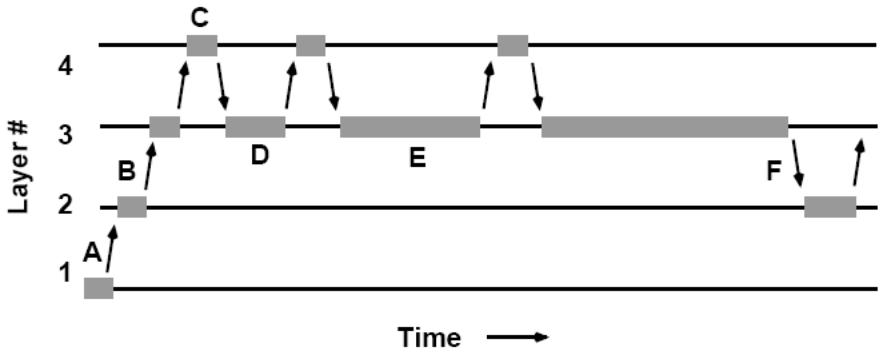

Figure 4.1: An RLM "sample realization" [50]

another join experiment from layer 3 (D). In RLM, a client maintains an exponential timer for each layer which provides a learning mechanism whereby clients converge towards the optimal subscription level over time.

4.2 Preemption on MPLS Networks

Preemption is a congestion control mechanism for removing streams off of a congested path to make room for higher priority arriving streams. Streams may be selected based on application type, bandwidth, customer class, or other criteria. Streams that are removed off of the congested route may be placed on a less desirable route, or the removed streams may become blocked from the system if an alternative route cannot be found. Preemption is also used to reroute traffic when link failures occur. Preemption results in a decreased blocking probability and improved alignment on shortest paths for high priority traffic at the expense of low priority traffic.

4.3 Related Work

Current research has focused on the areas of optimal adaptation policies [68, 76], as well the combined use of adaptation with admission control [15, 76, 75]. Bocheck et al. [9] propose and evaluate a content aware utility based adaptation system for delivery of MPEG-4 content. In [65], adaptation is used as a course grain response to congestion that is combined with a TCP-friendly congestion control scheme for very short time scales. Smart use of buffering at the receiver is used to absorb the differences in time scale.

The work in [68] investigates optimal policies for streams to dynamically adapt the fraction of their available bandwidth given to base and enhancement layers. The authors in [75] identify the asymptotic bounds of the optimal static adaptation policy using multi-class admission control and show that the expected subscription level approaches that of the optimal dynamic admission policy on large links, both mathematically and via simulation. The major drawbacks of the optimal adaptation policy are that it requires centralized knowledge and control of the network, and for high arrival rates, there is a large rate of adaptation for streams near the volume threshold. In addition, it is shown in [76] that a static volume based admission policy consisting of only two subscription rates can achieve near optimal QoS. Under this policy, at the time of admission, streams of short duration subscribe to the maximum rate, whereas streams of long duration subscribe to the minimum rate. For optimal performance, this requires accurate information of the network state at the time of admission.

Preemption is an attractive vehicle for implementing priority access policies, as well as granting favorable paths to high priority traffic. A large proportion of the literature has

focused on the policy used for deciding which (if any) LSP should be preempted. In 1981, Calabrese characterized the performance of a preemptive two class telephone network with two different preemption policies that differ only in the consideration of alternative routes. Garay and Gopal have shown that selecting the optimal stream which least "disturbs" the network is NP complete [24]. In response, the authors present a few simple heuristics for use in a centralized system.

In [61], two decentralized preemption algorithms are introduced. These algorithms optimize three fixed criteria constrained by the order of importance: number of connections, bandwidth, and priority. De Oliveira et al. [18] extend this work by allowing the service provider to specify the balance between the number of connections, bandwidth, and priority in the preemption policy. An adaptive preemption policy is proposed whereby lower priority LSP's reduce their rate if able to make room for the arriving high priority LSP. This allows for additional clients and minimizes the degree of re-routing. A simple and fast heuristic is proposed for the LSP selection in both algorithms and is shown to be a close approximation of the optimal solution. In [70], the authors consider an admission control policy for adaptive hierarchical streams. In this scheme, enhancement layers may be preempted in favor of new arrivals when the demand exceeds the network capacity. Two serious drawbacks are the lack of comparison to a baseline model and no mathematical expressions of performance.

4.4 Preemption and Adaptation Combined

In an effort to combat congestion on high bandwidth next generation networks, we investigate the combined use of preemption and adaptation. Adaptation can be thought of

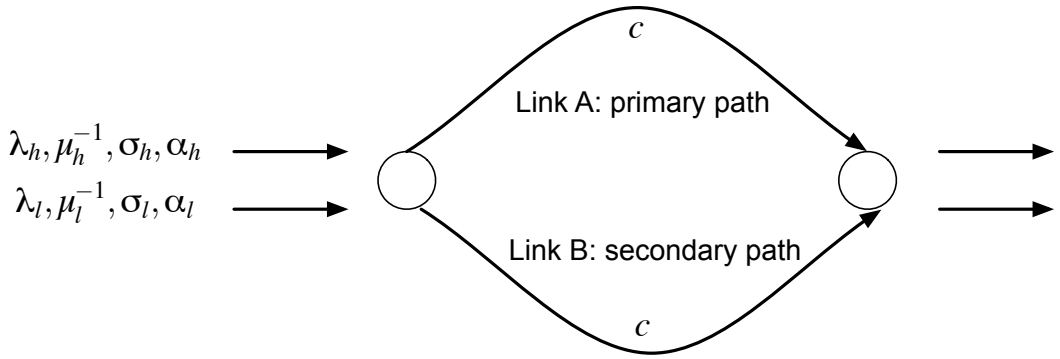

Figure 4.2: A two link network

as a preliminary measure to counter congestion. Adaptation has the affect of increasing the network capacity by reducing individual stream rates, yet adaptation is unable to make use of routing information to find free capacity along additional routes. Preemption is seen as a secondary measure to align priority traffic and re-route flows onto under-utilized routes. The combined use of adaptation and preemption yields an overall improvement in the blocking probability, quality of service, and traffic alignment.

In our study, a simple two-route network is constructed to readily demonstrate the affects of adaptation and preemption. The capacity of both links are fixed at 100 Mbps. One of the links (Link A) has a lower delay and is labeled as the primary path, see Figure 4.2. We consider streams that have two priority levels: High Priority(HP) and Low Priority (LP). In addition, streams have two subscription levels: minimum and maximum. Admission into the two-link network is governed by Algorithm 4. Arriving streams attempt to enter the network at the maximum rate on the primary route. When congestion occurs, we first consider adaptation to permit additional streams onto the primary path. When adaptation can no longer accommodate the arriving stream we then attempt to preempt lower priority

Parameter	Symbol	Value
Sim Duration	T	10,000 Arrivals
Link Capacity	c	100 Mb
Arrival Rate	λ	1-50
Mean Stream Duration	μ^{-1}	10.0 seconds
Max. Stream Size	σ_{max}	2 Mbps
Min. Stream Size	σ_{min}	1 Mbps
Random Number Generator		Mersenne Twister
Routing Protocol		CSPF

Table 4.1: Variables used for two-link simulations (C++)

streams onto the secondary path (Link B). Lastly, the arriving stream is blocked if neither mechanism can find sufficient space. When space on the congested route becomes available streams may be selected to increase their rate to make use of the available bandwidth. All of the relevant variables for the experiments are provided in Table 4.1.

We consider two traffic engineering metrics to measure performance:

- **Blocking Probability:** Streams are blocked when there is not sufficient room for admission. Preemption and adaptation may be used to free up space on the congested route. When adaptation is employed, an arriving stream is blocked after all the residing streams have been adapted, yet there still is insufficient room to fit the arriving stream. Using preemption, streams will be blocked from the route when there are no lower priority streams to preempt, meaning there is only high priority traffic on the available routes.

  For the two-route topology in Figure 4.2, blocking only occurs when both links are at full capacity. Using Little's Law we can predict the per-class arrival rate λ in which blocking begins to occur. Let n_h, n_l indicate the mean number high and low

Algorithm 4 Admission algorithm for two-link network

Admit HP arriving stream under the following conditions:

if HP arrival at full rate plus the aggregate full rate load on link A is less than the capacity of link A
then Admit the LP stream at full rate on link A
else if Adaptation is enabled and the HP arrival at adapted rate plus the aggregate adapted load on link A is less than the capacity of link A
then Admit HP arriving stream at adapted rate on link A
else if Preemption alone is enabled and enough LP streams can be moved from Link A to B
then Admit HP arriving stream at full rate on link A; Preempt lower priority streams to link B
else if Adaptation and Preemption are enabled, and the aggregate adapted rate LP load on link A exceeds the size of an adapted rate LP stream
then Admit HP arriving stream at adapted rate on link A; Preempt lower priority streams to link B
else if The aggregate full rate load on link B plus the full rate of a HP stream is less than the link B capacity
then Admit HP arriving stream at full rate on link B
else if The aggregate adapted rate load on link B plus the adapted rate of a HP stream is less than the link B capacity, and adaptation is enabled
then Admit HP arriving stream at the adapted rate on link B
else if Preemption alone is enabled and the aggregate full rate LP load on link B exceeds the size of a full rate HP stream
then Admit HP arriving stream at the full rate on link A; Preempt lower priority traffic to link B
else if Both adaptation and Preemption are enabled, and the aggregate adapted rate LP load on link B exceeds the size of and adapted LP stream
then Admit HP arriving stream at the adapted rate on link B; Preempt lower priority streams out of the system
else
Block the HP stream
end if

Admit LP arriving stream under the following conditions:

if The aggregate full rate load on link A plus the full rate of a LP stream is less than the link A capacity
then Admit the LP stream at full rate on link A
else if Adaptation is enabled and the aggregate adapted rate load on link A plus the adapted rate of a LP stream is less than the link A capacity
then Admit the LP stream at the adapted rate on link A
else if The aggregate full rate load on link B plus the full rate of a LP stream is less than the link B capacity
then Admit the LP stream at the adapted rate on link B
else if If adaptation is enabled and the aggregate adapted rate load on link B plus the adapted rate of a LP stream is less than the link B capacity
then Admit the LP stream at the adapted rate on link B
else
Block the LP stream
end if

priority streams in the system at steady state. Let λ_h, λ_l signify the high and low priority arrival rates respectively. In the absence of blocking we obtain the following relationship:

$$n_h = \lambda_h \mu_h^{-1}, \quad n_l = \lambda_l \mu_l^{-1} \tag{4.1}$$

Without adaptation or preemption for the two-route network, we calculate that the links will begin to fill to capacity at arrival rate λ = 10 for both classes (see equation 4.2). Applying adaptation alone effectively doubles the network capacity. Using equation 4.2 with σ_{min} in place of σ_{max} results in an arrival rate of λ = 20 for both classes. Preemption will allow twice as many HP streams in the system as high priority traffic will see only n_h, while the low priority arrival rate will be affected by $n_h + n_l$. As a result the arrival rates for the low and high priority classes are λ_l = 10 and λ_h = 20 respectively. Finally, using both adaptation and preemption permit twice as many streams than the preemption case for both classes.

$$n_h + n_l = n_{maxfull} \quad \rightarrow \quad \lambda \quad = \quad \lfloor \frac{2c}{\sigma_{max}} \rfloor \quad = \quad \lfloor \frac{200}{2 \times 10} \rfloor \quad = \quad 10 \tag{4.2}$$

Figures 4.3 and 4.4 illustrate the blocking probability for the low and high priority classes as a function of the arrival rate. These figures demonstrate the improved blocking probability for the high priority class when using either preemption or adaptation. When adaptation is used, the arrival rate at which congestion begins or con-

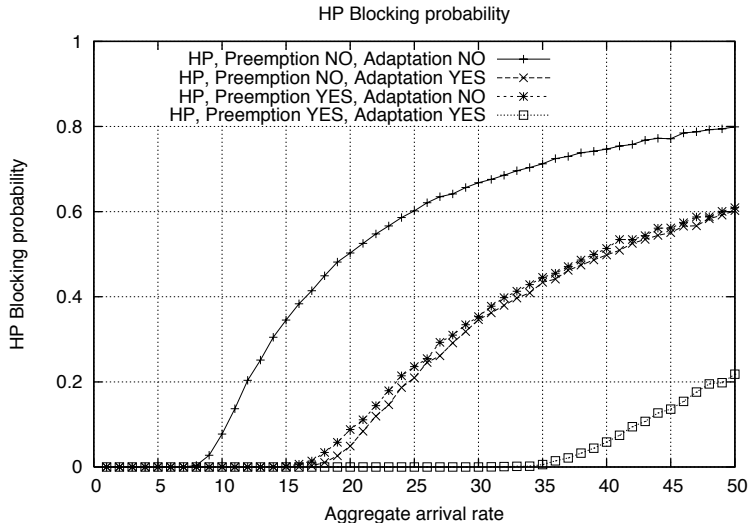

Figure 4.3: High priority blocking probability

gestion arrival rate is doubled. The use of preemption decreases the high priority congestion arrival rate two fold.

- **Traffic Alignment:** We measure the affect of preemption by computing the fraction of time a stream is on the primary path. The primary path is the path a stream would be placed on by the routing protocol in a non-congested regime. Figure 4.5 shows the fraction of time HP streams reside on the primary path. Considering the system begins to fill at $\lambda \approx 10$ with no adaptation or preemption, we expect HP traffic to start traveling along the secondary route at $\lambda \approx 5$. In Figure 4.6 it is shown that the LP fraction of time on Link A is the same. When either adaptation or preemption are used, we see that the HP traffic is able to stay on the primary route until $\lambda \approx 10$. The low priority fraction of time on the shortest route is significantly reduced when using preemption. In Figures 4.5 and 4.6 we see that the combined use of adaptation and

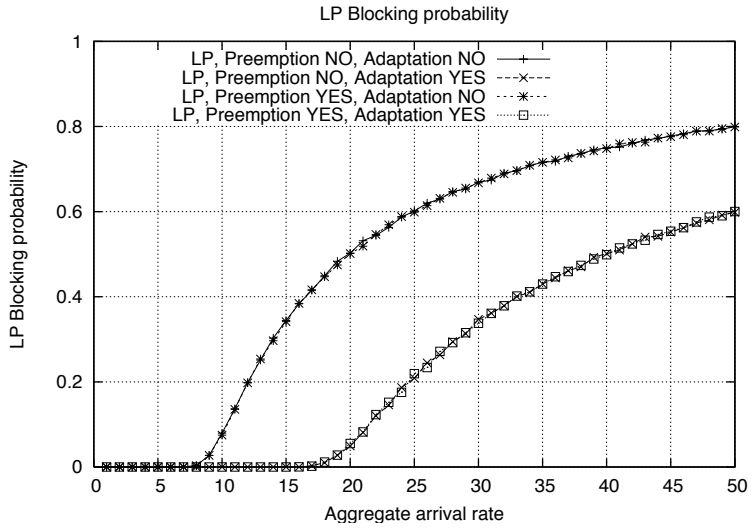

Figure 4.4: Low priority blocking probability

preemption yields a four fold increase in the number of high priority streams on the primary path.

The costs associated with preemption and adaptation can be expressed in three metrics. Adaptation decreases the quality of the stream and incurs some visual distortions with each rate change. While not measured here, each preemption in practice results in a non-negligible delay for the affected streams. Therefore, it is natural to consider each adaptation and preemption as a unit cost.

- **Rate of Adaptation (RoA):** Each rate change incurs a non-negligible decrease in stream quality. To capture this cost, the rate of adaptation is defined as the number of times a stream changes its bit-rate over the streams duration. The rate of adaptation is reported as a client average over all of the streams. Figure 4.7 depicts the rate of adaptation for the two-route network. For λ small, the RoA is zero as no adaptation

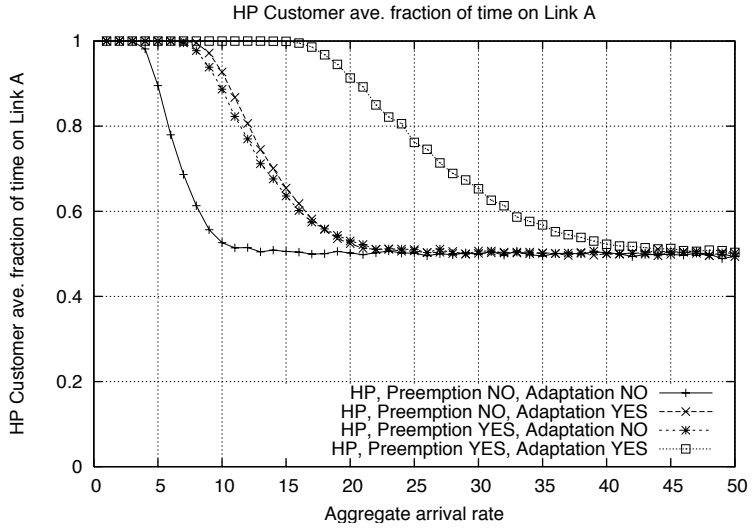

Figure 4.5: High priority customer average fraction of time spent on primary path

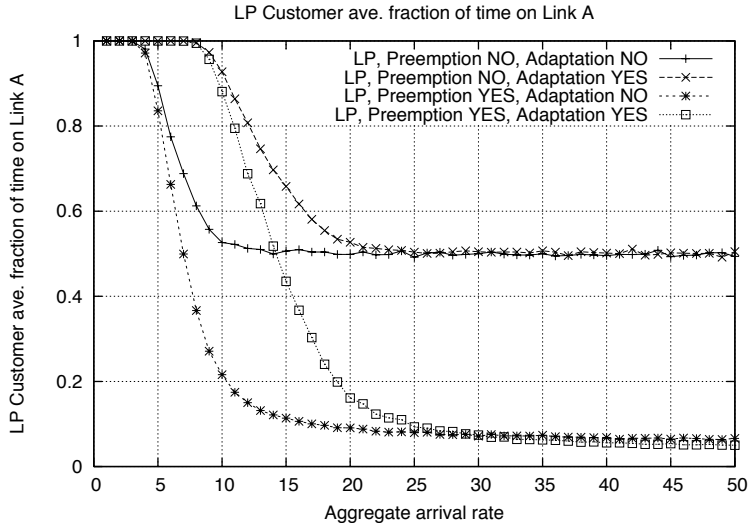

Figure 4.6: Low priority customer average fraction of time spent on primary path

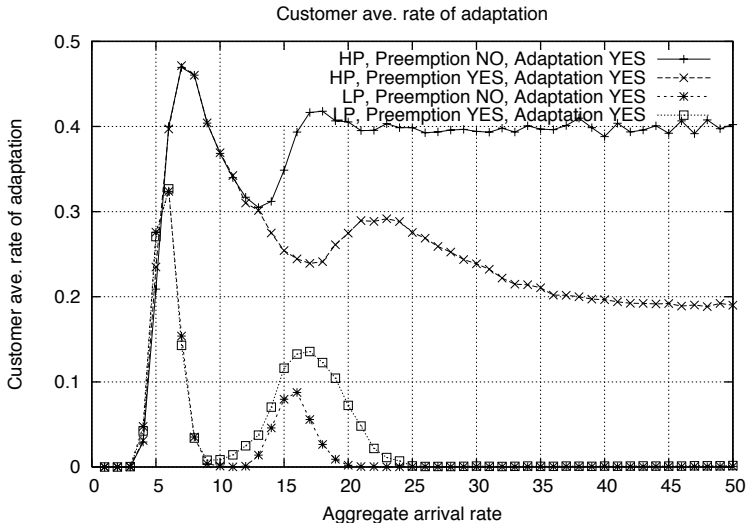

Figure 4.7: Customer average rate of adaptation

is necessary. Low priority traffic approaches zero as λ gets large because LP stream rates are the first to be reduced and last to be increased. When the network is in steady state HP departures are immediately followed by an adaptation. Then when the next arrival follows immediately after, an additional adaptation is required. This results in an average of two adaptation events for each HP stream for the adaptation and preemption case. Adaptation alone permits a mix of LP and HP traffic, and each HP stream is equally likely to adapt following a departure and arrival. The set of streams is effectively reduced in half with an adaptivity of a half, therefore the probability of being selected for adaptation is doubled. This results in an average of four adaptation events for each HP stream.

- **Quality of Service (QoS):** While quantifying the quality of streaming media is difficult to enumerate mathematically, it is reasonable to express the QoS for a stream

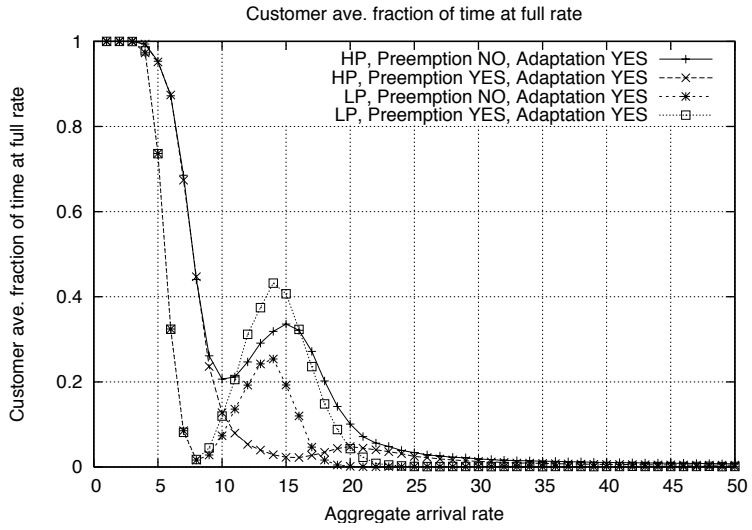

Figure 4.8: Customer average fraction of time spent at full rate

as the fraction of time the stream is at the maximum rate. Results for this metric are expressed as a client average for all streams in the network. In Figure 4.8 we plot the customer average fraction of time a stream is subscribed to the maximum stream rate. As the primary path begins to fill up from $\lambda \approx 5$ to $\lambda \approx 10$ streams begin to adapt to make room for additional traffic. From $\lambda \approx 10$ to $\lambda \approx 15$, adaptation briefly is able to reclaim unused bandwidth after streams depart. After $\lambda \approx 20$, the second path has saturated and no streams are ever at their max rate.

- **Rate of Preemption:** In practice, preemption stops transmission of the preempted stream, and transfers it to a new route. This delay is substantial, therefore we consider the rate of preemption as a cost metric. The rate of preemption is reported as the time average number of preemptions throughout the simulation duration. Figure 4.9 shows the time average rate of preemption for low priority traffic. The rate of preemption

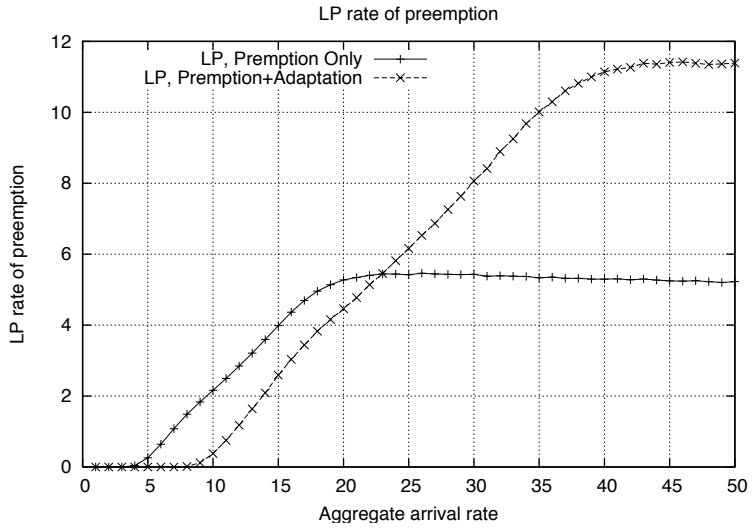

Figure 4.9: Time average rate of preemption

for the preemption only case increases until the network is saturated at $\lambda \approx 20$. The adaptation and preemption case have twice as high asymptotic rate of preemption. This is attributed to adaptation increasing the number of streams on the links so that twice as many preemption events occur.

In Figure 4.10 we analyze the runtime of each regime and find that the execution time increases with the arrival rate until the network begins to saturate. After $\lambda \approx 20$, a gradual decrease occurs as there are fewer adaptations and preemption events. Adaptation has the affect of greatly extending the runtime as the effective capacity is doubled. Figure 4.10 was generated using the same parameters of Table 4.1 with the exception of the stopping criterion which was increased to 100,000 arrivals for each data point.

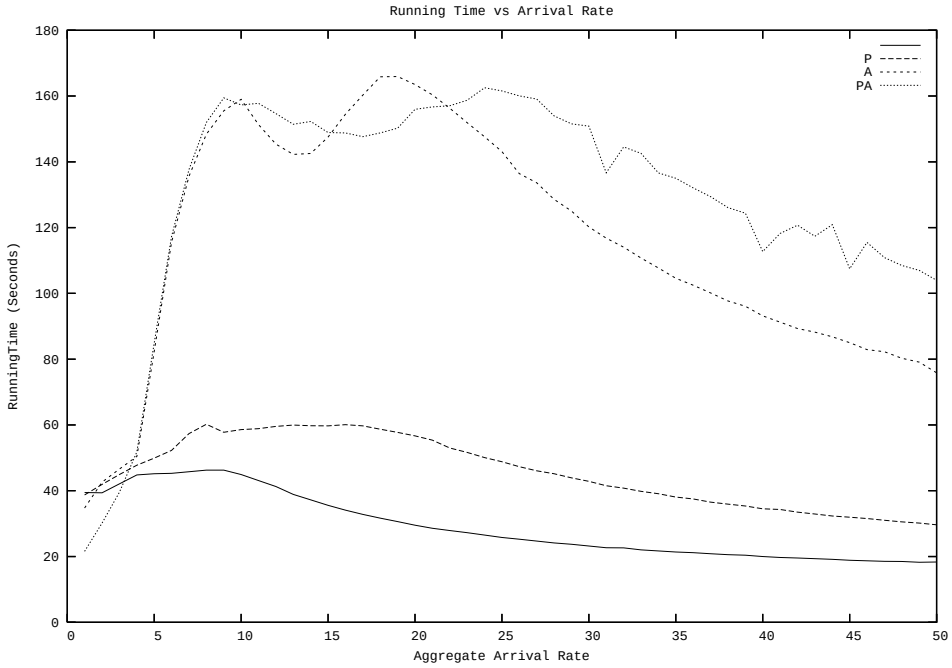

Figure 4.10: Running time of the preemption & adaptation simulations

4.5 Performance Improvements

At first glance, implementation of these algorithms might appear relatively straightforward, however a simulator that takes an arbitrary network graph as an input is fairly complex. Classes were used to represent the objects of the simulator such as nodes, links, routes, streams, and algorithms. All of these objects interact to form the network. However, the object oriented world was not entirely suitable for the implementation of certain routing algorithms. The representation of the network was converted from pointer based objects to linear arrays. Several tables were maintained, updated, and used by the routing and preemption algorithms. Ensuring proper maintenance of the tables for all of the algorithms and special cases proved to be the most time intensive and error-prone tasks during development. There are several problems that arise in the routing and re-routing of paths on the network. Consider a stream traversing multiple links, that is preempted onto a secondary link, however there is insufficient room on the new route unless another stream is adapted. Adapting this stream might trigger another stream on another route to increase it's stream rate as room is now available. Complex network interactions such as these are difficult to validate.

A simple lightweight discrete event simulator package called *Desmo-J* [57] was utilized to abstract away the scheduling of events and allowed for rapid development of the MPLS simulator. In addition, *JFreePlot* [26] was incorporated for data capture and plotting capabilities. Both tools for Java have been used in various other projects, and at first glance appeared to be suitable for the needs of the project. Other commercial simulators were not considered due to concerns of scalability, unnecessary complexity, and the high learn-

ing curve often accompanying these tools. For the purposes of studying adaptation and preemption, a stream-based abstraction was used in place of a high fidelity packet-based simulator. After the simulator was built, it was found that it did not scale well at all in the number of streams. Initial profiling of the simulator showed that memory usage exceeded system capacity causing a tremendous slowdown. The culprit for a large part of memory usage was *JFreePlot*, which was storing large amounts of data for each stream. After disabling the capture of data, memory was still being exceeded. Desmo-J used a producer and consumer paradigm for modeling of arrivals and departures in queuing systems. The problem was each customer was modeled with a thread that was put to sleep for its duration in the system, and then killed on departure. At or around 1,000 customers the system would come to a stand still, most likely caused by thrashing. Each thread uses a minimum amount of memory and so this presented a scalability problem as well.

In order to determine the performance difference between switching from a thread-based simulator to a single process discrete-event simulator, we measure the simulation running time versus the simulation duration and link capacity. The capacity can be calculated as the link capacity is fixed and the maximum size of each stream is 2 Mbps and adapts down to 1 Mbps. In the figures below, a huge decrease can be observed in the running time for a simple single link experiment by using the single-process simulator. Each data point represents the average execution time for 4 scenario cases: adaptation alone, preemption alone, no adaptation or preemption, and both adaptation and preemption. For each of these case the offered load to the link was scaled from 1 to 50 in increments of 1 for the simulation duration. These two metrics illustrate how the running time is effected by the number of streams in the network, and the simulation duration. The rate at which the

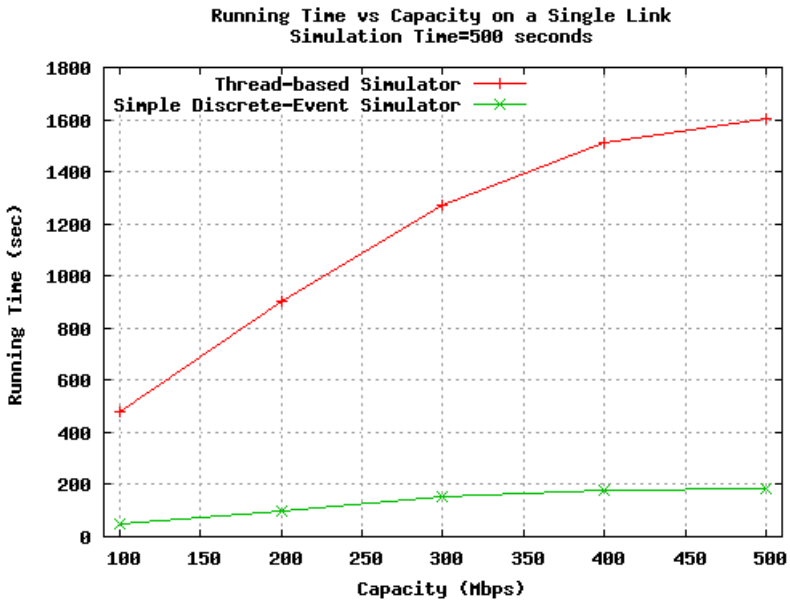

Figure 4.11: Running time vs system capacity on a single link

running time scales with the simulation time is much greater for the thread-based simulator compared with the single process simulator (see Figure 4.12). In Figure 4.11, we see as the effect of increasing the capacity of the link or maximum number of streams aggregated on the link. The thread-based simulator was found to be a memory hog as the number of threads grew short of the 1,000 mark and in part was also due to holding plot information in memory with *JFreePlot*. There is a clear improvement in the running time and the slope is higher for the thread-based simulator.

The main benefit for using a thread-based model is the simple implementation of finite state machines allowing for rapid development. Compared with an event-based model, state transitions are easily defined. An example illustrates this point best, consider two wireless radios who are transmitting a packet for a duration of 5 ms, if a third radio interferes with

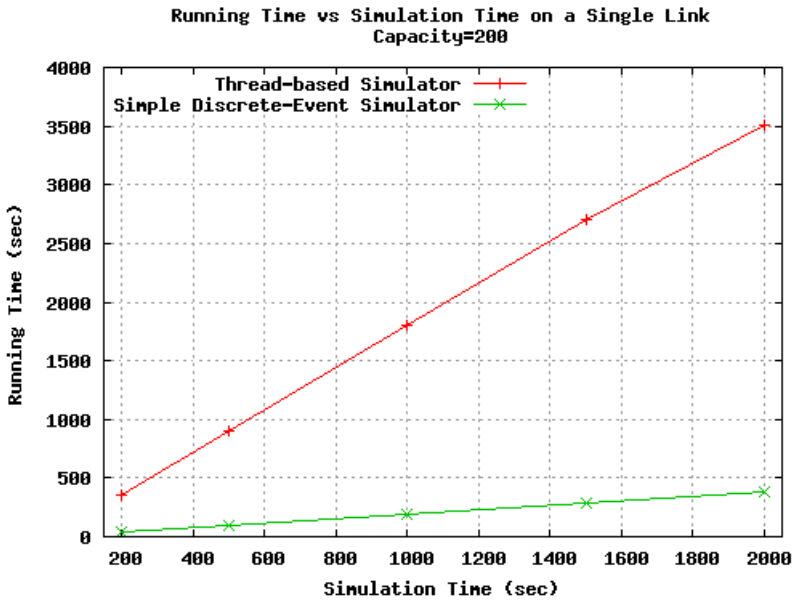

Figure 4.12: Running time vs simulation time on a single link

the ongoing transmission 1 ms into the first transmission, stations may need to change state perhaps by ending the transmission early, or higher level protocol states may need to modified as well. In a discrete-event simulator this event, would require removal of future events(packet reception in this example) planned for the future in place of an event that should occur at the present. In a thread-based based simulator, it's much easier to handle these synchronous issues where an interrupt is simply used to wake threads that are sleeping. The main problem with thread-based simulation models is the scalability in the number of threads. The two problems are the memory usage per thread and the thrashing which occurs when the number of threads becomes large.

Many architectures are unable to efficiently support more than a few thousand threads concurrently. The solution calls for a single process simulation engine, which does away with thrashing and memory usage issues associated with the thread-based implementation.

5. Future Work and Conclusions

A large part of our work has been focused on improving the efficiency of network simulation. We presented a protocol abstraction which can be applied to MACA based data link protocols on wireless ad hoc networks. It would be interesting to apply our approach in conjunction with the work of Naoumov and Gross [53]. Using a two dimensional data structure, the authors reduce the set of receivers considered for interference calculations. Applying this approach, we could thereby reduce the number of nodes considered at each updated interval. An additional avenue to take is to compare the performance of protocol abstractions in relation to different metrics. Our research suggested that some abstractions may be better suited for different metrics.

Future work on the combined study of adaptation and preemption might include an implementation of both algorithms in OPNET for validation purposes. Now that we have a simulator of scale, a plan is currently underway to experiment on larger topologies than the two link scenario. Our assumption that all streams are CBR does not hold in reality; instead UDP traffic must compete with variable bit-rate TCP traffic. We are interested in studying the extent to which the combined use of adaptation and preemption will help congestion on realistic backbone networks where these dynamics are at work.

Bibliography

[1] K. G. Abakoukmin. Design of transportation systems (in greek), 1986.

[2] JongSuk Ahn and SeungHyun Oh. Dynamic Calendar Queue. *Simulation Synopsium*, 00:20, 1999.

[3] T.R Andel and A. Yasinac. On the credibility of manet simulations. *Computer*, 39(7), July 2006.

[4] Lockheed Martin ATL. CSIM - Performance Simulator. http://www.atl.lmco.com/projects/csim/, 2006.

[5] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving Simulation for Network Research. Technical Report 99-702b, University of Southern California, March 1999.

[6] Hari Balakrishnan and Michael Goraczko. Oxygen TV. http://nms.csail.mit.edu/projects/oxygentv/, 2007.

[7] J.E. Barnes and P. Hut. A Hierarchical O(N Log N) Force Calculation Algorithm. *Nature*, 324(4):446–449, December 1986.

[8] Christian Bettstetter. Smooth is better than sharp: A random mobility model for simulation of wireless networks. *ACM MSWiM 2001*, July 2001.

[9] P. Bocheck, A. Campbell, S.F. Chang, and R. Lio. Utility-based Network Adaptation for MPEG-4 Systems. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSS-DAV)*, June 1999.

[10] J. Le Boudec and M. Vojnovi. Perfect Simulation and Stationarity of a Class of Mobility Models. In *Proceedings of IEEE Infocom*, 2005.

[11] Jean-Yves Le Boudec. Understanding the Simulation of Mobility Models with Palm Calculus. Technical report, Elsevier Science Publishers B. V., 2005.

[12] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in Network Simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[13] Tracy Camp, Jeff Boleng, and Vanessa Davies. A Survey of Mobility Models for Ad Hoc Network Research. In *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, volume 2, pages 483–502, 2002.

[14] David Cavin, Yoan Sasson, and Andre Schiper. On the Accuracy of MANET Simulators. In *Proceedings of ACM Workshop on Princ. Mobile Computing (POMC'02)*, pages 38–43, October 2002.

[15] Chun-Ting Chou and Kang Shin. Analysis of Combined Adaptive Bandwidth Allocation and Admission Control in Wireless Networks. In *Proceedings of IEEE Infocom*, 2002.

[16] Broadata Communications. Data sheet - t1/e1 560e fiber optic data modem. www.broadatacom.com/site/PDF/560e.pdf, 2006.

[17] S. Das, C. Perkins, and E. Royer. Performance comparision of two on-demand routing protocols for ad hoc networks. In *INFOCOM'2000*, 2000.

[18] J. de Oliveira, C. Scoglio, I. Akyildiz, and G. Uhl. A new preemption policy for diffserv-aware traffic engineering to minimize rerouting. *In Proceedings of IEEE INFOCOM02*, June 2002.

[19] F. Desbrandes, S. Bertolotti, and L. Dunand. OPNET 2.4: An Environment for Communication Network Modeling and Simulation. In *Proceedings of the European Simulation Symposium*, pages 609–614, Delft, Netherlands, October 1993. Society for Computer Simulation.

[20] Barclay Dutson, Claudia Dutson, and Stephen Drayson. New Opportunities in Streaming Media Report. In *Vision Consultancy Group*, September 2000.

[21] Albert Einstein. Uuml;ber die von der molekularkinetishchen theorie der warme gefordete bewegung von in ruhenden flussigkeiten suspendierten teilchen. *Annalen der Physik*, 17:549–560, 1905.

[22] David Falconer, Fumiyuki Adachi, and Bjorn Gudmundson. Time Division Multiple Access Methods for Wireless Personal Communications. *IEEE Communications Magazine*, 33:50–57, January 1995.

[23] Kevin Fall. NS Notes and Documentation. Technical report, The VINT Project, 2000.

[24] Juan A. Garay and Inder S. Gopal. Call preemption in communication networks. In *IEEE INFOCOM '92: Proceedings of the eleventh annual joint conference of the IEEE computer and communications societies on One world through communications (Vol. 3)*, pages 1043–1050. IEEE Computer Society Press, 1992.

[25] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, 2005.

[26] David Gilbert. JFreeChart Project. http://www.jfree.org/jfreechart/, 2007.

[27] Andrea Goldsmith. *Wireless Communication*. Cambridge University Press, New York, New York, 2005.

[28] Gregory J. Scaffidi and Mark Zohar. Consumer Broadband Hits Hypergrowth in 2001. In *Forrester Research*, October 2000.

[29] S. Hara and R. Prasad. Overview of multicarrier cdma. In *IEEE Communication Magazine*, pages 126–133, December 1997.

[30] T. He, B. Blum, Y. Pointurier, C. Lu, J.A. Stankovic, and S.H. Son. MAC Layer Abstraction for Simulation Scalability Improvements in Large-scale Sensor Networks. *Third International Conference on Networked Sensing Systems (INSS'06)*, May 2006.

[31] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. Effects of detail in wireless network simulation. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 1–11, January 2001.

[32] Polly Huang, Deborah Estrin, and John S. Heidemann. Enabling Large-Scale Simulations: Selective Abstraction Approach to the Study of Multicast Protocols. In *MASCOTS*, pages 241–248, 1998.

[33] Polly Huang and John Heidemann. Minimizing routing state for light-weight network simulation. In *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Cincinnati, Ohio, USA, August 2001. IEEE.

[34] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. *ACM Mobicom 2003*, September 2003.

[35] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[36] Douglas W. Jones. An empirical comparison of priority-queue and event-set implementations. *Communications ACM*, 29(4):300–311, 1986.

[37] P. Karn. A New Channel Access Method for Packet Radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Confrence*, 1990.

[38] Bruce Kasrel, Josh Bernoff, and Meredith Gerson. Broadband Content Splits. In *Forrester Research*, October 2000.

[39] W. Kreutzer. *Systems Simulation - Programming Styles and Languages*. Addison-Wesley, 1986.

[40] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET Simulation Studies: The Incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, 2005.

[41] Jean-Yves Le-Boudec and Milan Vojnovic. Perfect Simulation and Stationarity of a Class of Mobility Models. In *IEEE INFOCOM*, 2005.

[42] Pierre L'Ecuyer. Software for Uniform Random Number Generation: Distinguishing the Good and the Bad. In *Proceedings of the 33nd Conference on Winter Simulation*, pages 95–105, 2001.

[43] G. Lin, G. Noubir, and R. Rajamaran. Mobility Models for Ad-hoc Network Simulation. In *Proceedings of Infocom 2004*, 2004.

[44] C. Liu, M. A. Orgun, and K. Zhang. A parallel execution model for Chronolog. *International Journal of Computer Systems Science and Engineering*, 16(4), 2001.

[45] Jason Liu, David M. Nicol, L. Felipe Perrone, and Michael Liljenstam. Towards High Performance Modeling Of The 802.11 Wireless Protocol. In *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[46] Xiaowen Liu. *Improvements in Conservative Parallel Simulation of Large-Scale Models*. PhD thesis, Dartmouth College, 2001.

[47] J. G. Markoulidakis, G. L. Lyberopoulos, and M. E. Anagnostou. Traffic model for third generation cellular mobile telecommunication systems. *Wireless Networks*, 4(5), 1998.

[48] George Marsaglia. Diehard battery of tests of randomness v0.2 beta. http://www.csis.hku.hk/ diehard/, 2006.

[49] Makoto Matsumoto and Takuji Nishimura. Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudorandom Number Generator. *ACM Transactions on Modeling and Computer Simulations*, 1998.

[50] Steve McCanne. Scalable Compression and Transmission of Internet Multicast Video, 1995. Ph.D. thesis, University of California at Berkeley.

[51] Earl McCune and Kamilo Feher. Closed-Form Propagation Model Combining One or More Propagation Constant Segments. In *Proceedings of the 47th IEEE Vendor Technology Confrence*, volume 2, pages 1108–1112, May 1997.

[52] C.S.R. Murthy and B.S. Manoj. *Ad Hoc Wireless Networks - Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2004.

[53] Valeri Naoumov and Thomas Gross. Simulation of Large Ad Hoc Networks. *In ACM MSWiM*, pages 50–57, 2003.

[54] William Navidi and Tracy Camp. Stationary distributions for the random waypoint model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, 2004.

[55] William Navidi, Tracy Camp, and Nick Bauer. Improving the accuracy of random waypoint simulations through steady-state initialization. In *Proceedings of the 15th International Conference on Modeling and Simulation (MS)*, 2004.

[56] SSF Research Network. SSFNET - Scalable Simulation Framework. http://www.ssfnet.org/homePage.html, 2002.

[57] University of Hamburg. DESMO-J Simulation Framework. http://www.desmoj.de/, 2007.

[58] K. Pawlikowski, J. Jeong, and R. Lee. Letters to the editor. *IEEE Communications Magazine*, pages 132–139, 2002.

[59] D. D. Perkins, H. D. Hughes, and C. B. Brown. Factors Affecting the Performance of Mobile Ad Hoc Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2002.

[60] L. Perrone and D. Nicol. Using n-body algorthims for interference computation in wireless celular simulations. In *MASCOTS 2000 Intl. Workshop Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 49–56, 2000.

[61] M. Peyravian and A.D. Kshemkalyani. Connection preemption: issues, algorithms, and a simulation study. In *Proceedings of INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 143–151, April 1997.

[62] James Gary Propp and David Bruce Wilson. Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics. *Random Structures and Algorithms*, 9(1&2):223–252, 1996.

[63] T. S. Rappaport. *Wireless Communications: Principles and Practice, Second Edition*. Prentice Hall, 2002.

[64] Inc. RealNetworks. Real Media - Helix DNA Open Source Platform. https://helixcommunity.org/, 2007.

[65] Reza Rejaie, Mark Handley, and Deborah Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *SIGCOMM*, pages 189–200, 1999.

[66] Larry Rowe and Steven McCanne. The OpenMash Consortium. http://www.jfree.org/jfreechart/, 2006.

[67] E. Royer, P. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Proceedings of the IEEE International Confrence on Communications (ICC)*, June 2001.

[68] Despina Saparalla and Keith Ross. Optimal streaming of layered video. In *Proceedings of IEEE Infocom*, 2000.

[69] Thomas J. Schriber and Daniel T. Brunner. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2005 Winter Simulation Confrence*, volume 1, pages 167–177, 2005.

[70] N. Shacham. Preemption-based admission control in multimedia multiparty communications. *INFOCOM*, 02:827, 1995.

[71] P. M. Shankar. *Introduction to Wireless Systems*. John Wiley & Sons, Inc., New York, 2002.

[72] M. Takai, R. Bagrodia, A. Lee, and M. Gerla. Impact of Channel Models on Simulation of Large-Scale Wireless Networks. In *Proceedings of ACM MSWiM'99*, 1999.

[73] M. Takai, J. Martin, and R. Bagrodia. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks. In *Proceedings of MobiHoc*, pages 87–94, 2001.

[74] Steven Weber. Notes: Confidence intervals. www.ece.drexel.edu/faculty/sweber, 2006.

[75] Steven Weber and Gustavo de Veciana. Asymptotic analysis of optimal adaptation of rate adaptive multimedia streams. *In Telecommunications Network Design and Management, G. Anandalingam and S. Raghaven, Eds. Kluwer Academic Publishers*, 2003.

[76] Steven Weber and Gustavo de Veciana. Network design for rate adaptive multimedia streams. In *IEEE Infocom*, 2003.

[77] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[78] J.H. Winters and J. Salz. Upper Bounds on the Bit-Error Rate of Optimum Combining in Wireless Systems. *IEEE Transactions on Communications*, 46:1619–1624, December 1998.

[79] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *Proceedings of INFOCOM. IEEE*, 2003.

[80] J. Yoon, M. Liu, and B. Noble. Sound Mobility Models. In *Proceedings of ACM MobiCom*, pages 205–216, September 2003.

[81] Christopher W. Zobel and K. Preston White. Determining a Warm-up Period for a Telephone Network Routing Simulation. In *Winter Simulation Conference*, pages 662–665, 1999.