# Progressive Scaling: Methodology for Tunning and Validating Large Ad Hoc Networks Simulations

Cesar Santivanez
BBN Technologies
10 Moulton St., Cambridge, MA 02138
csantiva@bbn.com

A. Bruce McDonald
Northeastern University
360 Huntington Ave., Boston, MA 02115
mcdonald@ece.neu.edu

## Abstract

Validating large scale ad hoc network simulation is usually infeasible due to the high cost associated with implementing a network with so many nodes. Indeed, typical ad hoc network implementations (testbeds) consist of at most tens of nodes. To complicate matters further, the high cost of high performance simulation platforms and the limited time available for simulation study renders the use of high-fidelity simulations impractical. Thus, large scale networks are studied by means of low fidelity simulations, where the physical and MAC layer behavior is 'abstracted' with non-obvious impact on the quality of the results. In this paper, we present a technique, namely 'Progressive Scaling', that allows evaluating and improving the quality of large scale simulations at a reasonable cost. The use of Progressive Scaling provides the research community with an increased confidence on the validity of conclusions derived through large scale simulations. This technique is illustrated by an example of the authors experience applying it to the study of routing scalability for large ad hoc networks.

## 1 INTRODUCTION

An ad hoc network is a wireless network where the nodes (possibly mobile) communicate (possibly using multiple hops) without the presence of an infrastructure. Ad hoc networks are suitable in situations where the network must be rapidly deployed and function without an infrastructure, such as military communications, disaster relief, campus networks, etc.

Recent years have witnessed a surge in the interest in ad hoc networks. Spurred by ever-decreasing form-factors and cost of wireless transceivers and processors, a multitude of new applications are emerging. These include short-range ad hoc wireless networks for ubiquitous computing, larger range indoor wireless LANs that operate in ad hoc mode, metropolitan area networks (e.g. Metricom [1], Rooftop [2]), and sensor networks [3]. Standards such as Bluetooth, HomeRF, and IEEE 802.11 are giving impetus to the growth in the number of ad hoc communication enabled devices. Indeed, large ad hoc networks are on their way.

Before large ad hoc networks are deployed, however, we need to design efficient protocols to be run over them. Unfortunately, as it was the case for other large scale networks, the use of large-scale real-life testbed is prohibitively expensive. We are, then, forced to use simulation to study and compare different candidate protocols.

Simulation studies of ad hoc networks, however, need to circunvent two main problems. First, that the performance of a mobile network, specially an ad hoc network, is highly dependent on the obscure nature of radio propagation. Radio propagation in turn depends on many dynamic and unpredictable operating conditions, which makes it extremely hard to be accurately modeled in a simulation. Thus, the evaluation of an ad hoc networking system is incomplete without validation using a real life prototype. Second, even if high fidelity models of the physical (as well as MAC) layer were available, employing them will require too much processing power resulting in a high cost in time and money. Thus, researchers are usually forced to relay on simulations employing lower fidelity, abstracted models for studying large ad hoc networks. However, abstracted model of the physical layers may easily bias the researchers results. For example, in [4] the authors showed examples where the non-uniform impact that using different physical layer models has on the resulting performance of the simulated routing protocols caused a difference on the ranking of the routing protocols depending on the physical layer abstraction being used. Thus, the question that we ask ourselves is: *how – short of building a large scale prototype system – can we be sure that the 'abstracted' models used on our simulations did not biased our results.?*

The BBN team developing network-layer protocols in the the Density- and Asymmetry-adaptive Wireless Network (DAWN) project[5] had to confront this problem. The DAWN project is part of the DARPA Global Mobile Information Systems (Glomo) program, and builds upon a system developed as part of the (also DARPA) Multimedia Support for Mobile Wireless Networks (MMWN)[6]. The MMWN/DAWN system (referred to as simply the DAWN system) provides network-layer support for QoS in large (hundred of nodes), dense, mobile ad hoc networks. The DAWN project team faced the challange of validating its scalable techniques beyond the capabilities of DAWN's 10 node testbed[7]. In particular, the main author – as part of the DAWN project team – was responsible for evaluating a family of scalable variants to Link State Routing, namely Fuzzy Sighted Link State (FSLS) Routing, determining the best algorithm in this family.

This paper describes the DAWN project team solution to this problem: a technique we call *Progressive Scaling*. Progressive scaling consists on running our experiments on several different platforms. The first platform is a testbed of actual radios. As many as economically feasible. Each sucesive platform reduce the fidelity of the experiment and therefore it also reduces our confidence on the results. This confidence is regained by validating the results obtained in the lower fidelity platform again the results obtained with the higher fidelity platform. This process is further explained in Section 2.

The remainder of the paper illustrates the use of Progressive Scaling by describing its application in the study of the algorithms in the FSLS family. Section 3 briefly describes the FSLS family and Section 4 discusses our experience applying Progressive Scaling to determine/validate the best algorithm on this family (FSLS). Finally, Section 5 presents the conclusions of this work.

# 2 PROGRESSIVE SCALING

We now consider a question that is fundamental to the study of network scalability. Given that a real-life prototype has a large hardware cost (nearly $5000 per node in our case), it is prohibitively expensive, at least at the research stage, to construct a large testbed. How then does one evaluate whether the mechanisms are scalable or not? While this is a question that is valid for *any* kind of network, it is doubly important in ad hoc wireless networks since the channel and radio properties are very difficult to capture in a model.

In our work, we have used an approach that may be described as *progressive scaling*. We have three evaluation methods – a real-life testbed of 10 nodes; an emu-

lated network where we run the very same software that runs on the testbed, but replace the radios by a detailed software emulation of the same; and an simulation model in the OPNET simulation tool.

Our approach is illustrated in Figure 1. A set of experiments is done on a 10 node real-life testbed, and repeated for the emulated 10 node network. We compare the results of these experiments and tune the emulation so that the model reflects real-life as closely as possible. We then scale up the emulation to about 50 nodes. Larger numbers of nodes is impractical since this is a very high fidelity emulation. We then conduct a set of experiments on a 50 node OPNET simulation model and compare it with the results from the emulated network, and again, tune the OPNET simulation to synchronize the results. We then conduct 800 node experiments with the OPNET model. This "synch-n-scale" method can be extended further using progressively lower fidelity models if necessary.

We have only been able to follow this approach partially due to lack of resources and the proprietary nature of some of the protocols (e.g., the Nokia MAC protocol). We have also found that modeling the radio and the channel identically in two disparate simulation tools is extremely difficult (it is important to do so, as the performance is very sensitive to this). Nonetheless, such progressive scaling has given us much more confidence in our large-scale simulation results than if had only run a 800 node simulation.

In the reminder of this section, we describe each of the platforms we used.

## 2.1 Radio Testbed

The DAWN testbed is based upon the LR 4000 embedded router from Nokia Wireless Routers (formerly Rooftop Communications) [2]. This is a wireless IP router product that uses the Utilicom Longranger 2050 radio modem, and a Motorola processor with Flash ROM and RAM, all integrated into a single "box". The Utilicom radio is a 2.4 GHz ISM band, spread spectrum radio with programmable data rates upto 1.676 Mbps.

We selected this product over other candidates (e.g. the popular WaveLAN PCMCIA with a notebook) for three main reasons. First, our software could be downloaded to run in embedded mode which obviates the need for a notebook computer and gives better performance. Second, the Utilicom radio was the only commercial radio that provides transmit power control – a key requirement for the DAWN topology control mechanism. Third, it came with a suite of IP-related protocols that made it easy for us to attach to other networks using IP (DAWN is a "subnet" or "link" level system from IP's viewpoint
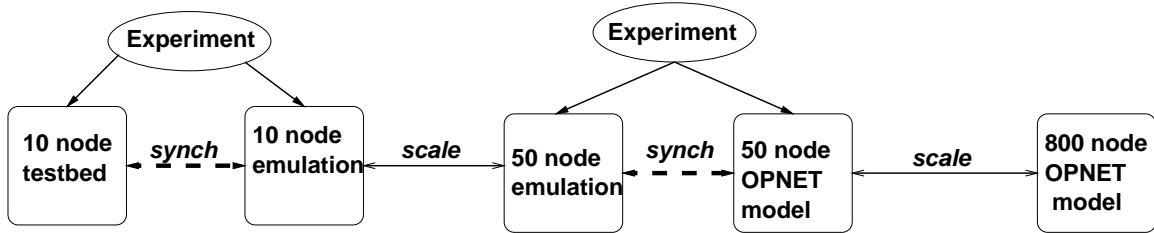
Figure 1: The "synch-n-scale" approach to validating scalability of protocols without compromising fidelity. Results from a real-life testbed, an emulation thereof, and a simulation are used to progressively validate system performance.
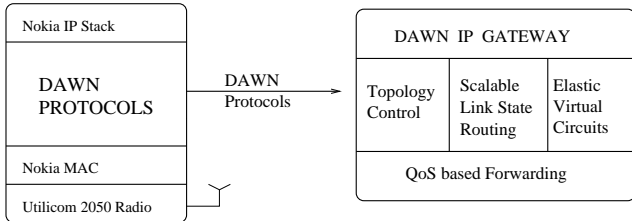


Figure 2: The components of a testbed node, including the modules comprising the DAWN protocols

and runs "below" IP). Fourth, it provided a development environment, the C++ Protocol Toolkit (CPT) that allowed the same software to be downloaded into the radio to be used for high fidelity simulation/emulation, reducing the time from algorithm design/simulation to actual implementation.

The protocol modules comprising the DAWN embedded system are illustrated in Figure 2, along with the layers above and below them. In this paper we will discuss our experiments with the Scalable Link State Routing module. More information about our testbed and the experiments run over it can be found in [7].

## 2.2   CPT Emulation

As mentioned before, the DAWN code running embedded in the LR4000 was developed using the C++ Protocol Toolkit (CPT) provided by Nokia Wireless Routers. CPT provides a number of commonly used functions such as timer and packet header manipulations. Additionally, the CPT toolkit already included a library of protocols to choose from, so we need not spend time developing one.

The CPT toolkit also included a high fidelity software model of the radio (utilicom 2050) and the wireless channel (assuming $1/r^4$ propagation characteristics). Thus, with CPT the **same code** used in real life can be used on simulation by simple replacing the radio and the channel by their software models, significantly reducing the time

required for assembly and debugging.

One immediate advantage of using the CPT toolkit, was our ability to expand the limits of our experiments far beyond the 10-node limit of our testbed. On the other hand, the proprietary nature of the radio and channel models provided prevented us from modifying/improving these models to better match our experiences or the models used on the OPNET platform.

We run our CPT emulations on a FreeBSD system. The hard-coded limit on the number of nodes was of 100 nodes. However, before reaching this number the running times due to the high fidelity modeling became too long. The actual maximum network size we used in our experiments was of 80 nodes.

## 2.3   OPNET

Above the tens of nodes, the use of CPT high fidelity emulations was not practical. For evaluating our protocols on larger size networks we rely on simulations obtained using OPNET version 6 running on Solaris workstations.

The OPNET simulations used on our Scalable Link State Routing studies employed high fidelity models of the routing and MAC algorithms. No 'abstraction' was allowed in these modules. No 'backdoor' communication was allowed: nodes only used locally available information obtained through packet exchange eith neighboring nodes. Mechanisms were not assumed to 'work properly' but were actually implemented.

The radio and channel modules used simplified abstract models based on the default OPNET's wireless radio pipeline stages. These stages, however, had to be modified to increase fidelity as follows:

- The receiver power decay with distance $r$ was set to $r^4$ instead of $r^2$.

- A packet with extremely low signal is no longer cause for the *carrier busy* signal to be triggered. This feature made sense in a wireless LAN where anyone can hear to anyone, but in a multihop net-

work some nodes transmissions are just too weak to be perceived.

- Since several low power packets may combine into a significant interference, we needed to keep track of these packets and the cumulative interference they cause. This was not done in OPNET by default since its model assume that the channel would lock to the first signal that arrived (regardless of its power) and treated all remaining signals as noise. Note that not keeping track of interference signal caused OPNET to declare the channel free after completing reception of a weak signal even if a stronger signal was in the middle of reception.

- The carrier sensing (signal lock) signal is declared ACTIVE only if the combined received energy (background noise, interference from other packets, and received power of the currently processed packet - if any) exceeds a configurable threshold. This better models what happens in the radio (synchronization stage). So, not only a weak signal won't cause the carrier sensing to trigger, but the carrier busy signal won't disappear upon completion of a packet reception if the combined interference is still above the threshold.

The above changes inceases OPNET processing burden, since packets that were treated as 'NOISE' (less processing required) are now tracked as 'VALID' in order to properly calculate the interference power. To alleviate in part this increase in processing requirements, we modified the closure stage so that it only processes packets whose receive power is above a threshold. So, packets whose energy is so low that not even combined may trigger the carrier busy signal in an actual radio are no processed at the receiver, reducing the processing burden.

Thus, the OPNET radio model employed was a high fidelity one, if compared with typical OPNET simulations. However, it differ from the real-life radio in two main aspects. First, our modeling of the packet synchronization and capture process was simple: a threshold comparator with a binary output instead of − at least − an stochastic process with a probability distribution dependent on the signal-to-noise ratio at a given time. Second, the pathloss was assumed dependent only on distance. A decay exponent of 4 was used to capture the effect of multipath and other physical layer phenomenae. Large pathloss changes due to small displacements − presents even on broadband system − were ignored. The effect of these variations, though, were somehow recovered by the random nature of OPNET BER computation, which made possible that two consecutive packets be received with a different number of bits in errors.

Overall, our OPNET models provided high quality simulations at a reasonable simualtion time. However, when we increased the network size to 800 nodes, the simulation time become too long. If bigger networks are to be studied, a decrease in the simulation fidelity would be necessary (of course, using the progressive scaling technique to validate teh models).

# 3 FSLS ALGORITHMS

Standard Link State (SLS) routing, where each node sends a Link State Update (LSU) immediately after a link status change (e.g. a link goes up or down), does not scale well with network size. The control overhead per node it induces increases linearly with network size, eventually consuming the node entire available bandwidth. A reduction in the control overhead, however, may be obtained both in space (limiting who the link state update is transmited to) and time (limiting the time between successive link status information dissemination). This observation motivates the study of the family of Fuzzy Sighted Link State (FSLS) algorithms.

In a highly mobile environment, an FSLS approach will transmit link status updates at particular time instant that are multiple of $t_e$ seconds. Thus, several link changes are 'collected' and transmitted every $t_e$ seconds. The *Time To Live* (TTL) field of the link status update (LSU) packet is set to a value $s_i$ which determines that the LSU packets is transmitted to a distance of $s_i$ hops from the node where the LSU is originated (see Figure 3, where a vertical arrow represents that a LSU is sent at that time, and the number above the arrow represents the number of hops that the LSU will travel). The above approach guarantees that a node that is $s_i$ hops away from a tagged node will learn about the tagged node's link status change at most after $2^{i-1}t_e$ seconds.

Different approaches may be implemented with different choices of the $s_i$ function. For example, if $s_i = \infty$, the Discretized Link State (DLS) approach is obtained. DLS is similar to SLS and only differs in that DLS do not send a LSU immediately after a link status change is detected by it waits for the $t_e$ interval to be completed to collect several link status changes in one LSU. DLS is an improvement of SLS that tries to scale with mobility.

Similarly, if $s_i = k$ for $i < p$ and $s_i = \infty$ the Near Sighted Link State approach (NSLS) is obtained. NSLS uses the 'memory' of past links to forward packets in the direction of the former destination location. As the packet gets to a node that is on the 'sight' (i.e. less than $k$ hops away) of the destination, this node will know how to forward the packet to the destination. NSLS behavior
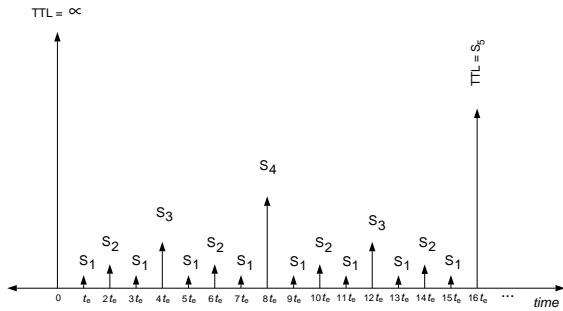
Figure 3: General LSU generation process for the family of Hazy Sighted Link State Algorithms

is similar to a 2 level cluster approach and is best suited for scenarios where the traffic is localized , that is, the most likely destinations are the nodes inside the 'sight' area.

FSLS is based in the observation that nodes that are far away do not need to have complete topological information in order to make a good next hop decision, thus propagating every link status change over the network may not be necessary. FSLS reduction on control overhead, however, comes with the cost of reducing the quality/availability of the source-destination routes. The impact of such sub-optimal routing is dependent on the level and diversity of traffic that the network must carry. Under certain assumptions (most notably the uniformity of the traffic) the best algorithm on the FSLS was found to be the Hazy Sighted Link State (HSLS) algorithm, in which the $s_i = 2^i$. The interested reader is referred to [8] for details of the analysis.

In this paper we describe our experience using *Progressive Scaling* in the validation of our theoretical results stating that HSLS outperfroms SLS, DLS, and NSLS.

# 4 APPLYNG PROGRESSIVE SCALING

This section illustrates the process followed to validate our large scale simulations of algorithms in the FSLS family.

The first step was to sanity-check our software. To this end we set up a multi-hop topology using our radio testbed. Setting this multihop topology in a reduced-size setting turned out to be more difficult than we expected. We need to find the right separation and location of nodes so that they can not listen to each other directly. We learnt that obstructions has greater effect than distance in pathloss. We observed packets being

dropped when, for example, a car passed by one of the nodes. We noticed link flappings, and a higher than expected number of unidirectional links. We tune our neighbor discovery module to reduce the effect of link flapping and designed our algorithms to be resilient to unidirectional links.

Our system was put to the final test when we experiment with the 10-node network in a realistic setting: the McKenna test site in Ft. Benning, Georgia. The site has a number of "mock" buildings, simulating a small town. Soldiers with backpacks containing DAWN nodes played out military scenarios with traffic generators simulating real-life voice and message exchanges. It was extremely instructive to observe the vast difference in propagation characteristic between an open environment and one with buildings. As nodes moved inside or amongst building the connectivity started to degrade rapidly. The Ft. Benning experiment took a large amount of time and effor to orchestrate and analyze, but it provided invaluable insight into the characteristics of the physical medium our system should be run over: rapidly changing connectivity with a fair amount of unidirectional links.

Our radio testbed experiments confirmed that our algorithms were mature enough to run on real life scenarios: there were not unexpected physical layer dynamics that would affect its performance to the point of render it ineffective. With this validation we proceed to run several emulation with different network sizes. The smaller network experiments where compared against our radio testbed experience. At his point, an exact match can not be found due mainly to the difficulty of matching channel parameters such as instantaneous pathloss and the proprietary nature of Nokia's CPT's channel model. However, we verify that in both cases (testbed and emulation) the network layer algorithms (routing in this case) experience similar stress (in terms of link flappings and presence of unidirectional links).

For small networks, the emulation results for different algorithms was similar. However, when we increase the network size we could observe that algorithms in the FSLS family outperformed the Standard Link State (SLS) approach. In figure 4 (left) we can see the result of an experiment comparing the 4 algorithms under study in 60-node networks of different densities. The results are typical at this network size: all FSLS algorithms (DLS, NSLS, HSLS) outperform SLS, but there is no clear winner among them. To decide which is the most scalable algorithm in this class we need to increase the network size beyond the 100-node allowed by our CPT emulator.

Larger network size experiments were conducted in OPNET. Figure 4 (right) shows the results obtained for

| Algorithm | Throughput |
|-----------|------------|
| NSLS | 0.3516 |
| HSLS | 0.446 |

Table 1: Throughput for a 800-node network

network sizes from 100 to 400. Note that SLS is no longer simulated since it is already clear that it is outperformed by te FSLS algorithms and SLS simulations take much more time. We notice that NSLS and HSLS – which behave similarly when the netowrk radius is small – outperform DLS. But, could this difference be a product of a unfair biased of OPNET models againt DLS??. To answer this question we used our CPT emulation platform. Since in CPT emulation the performance of DLS and HSLS (NSLS) was too close and OPNET results at 100-node networks were not conclusive either, we decide to reduce the transmission rate of our CPT emulations so that the network is stressed similarly as OPNET's 400 node network case. To this end, we used the Utilicom 2020 emulation (similarly to Utilicom2050 but with a lower transmission rate of 298 kbps). The densities were also adjusted to match the longer transmission range at this lower speed, matching OPNET average node degree. The results of this experiments are shown in Figure 4 (center), where the throughput of a 80-node network for different node speeds is shown. Since NSLS behaves similarly as HSLS at this size, only HSLS is ploted. SLS is also plotted for completeness (and because at this size it does not take so much to run). This results validate OPNET results, that HSLS (and NSLS) outperforms DLS.

It should be noted that CPT's and OPNET's matching was not straightforward, since different channel models (Nokia's proprietary model apparently presented lower fidelity than OPNET's) and even differences in the quality of the random number generated in both platforms (i.g. correlation among consecutive random numbers) influenced the results. However, even before a careful analysis was underwent to minimize the impact of this diferences in the network layer protocols, the relative advantage of HSLS (and NSLS) over DLS was evident.

With this reassurance of our OPNET model, we used it to compare NSLS and HSLS at higher network sizes. At 400-nodes we already observed a trend of HSLS starting to outperform NSLS. When we increase the network size to 800 nodes we obtained the results shown in Table 1, where HSLS scalability superiority is evident.

Thus, the confidence in our OPNET simulations gained through the use of the progressive scaling tech-

nique allowed us to determine the best algorithm in the FSLS family and at the same time identify those simulation artifacts that may bias our results, allowing us to mitigate their effect. Furthermore, the identification of these artifacts clear us of the uncertainty about results not affeced by them, and therefore we can make a more extensive and efficient use of the data collected in our simulations.

## 5  CONCLUSIONS

Progressive scaling is a powerful technique that helps to validate/improve the confidence on our simulations.

Although perfect matching among platforms is not always possible, identifying and mitigating the simulation artifacts (such as imperfect channel models, optimistic assumption about slow varying channels and link qualities, dependence on random numbers used on different nodes, etc.) that bias performance improve the quality of our results.

Finally, the confidence an insight gained through the process of progressive scaling allows to extract more useful data and gain more knowledge out of our simulation logs.

This technique was successfully applied by the authors in a comparative study of scalable routing algorithm for ad hoc networks.

## 6  ACKNOWLEDGEMENTS

## References

[1] http://www.metricom.com

[2] http://www.rooftop.com

[3] G. Pottie and W. Kaiser, "Wireless Sensor Networks." *Communications of the ACM, 2000.*

[4] M. Takai;J. Martin;R. Bagrodia. 2001. "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks." In *Proceedings of MobiHoc'01* (Long Beach, CA, Oct. 4-5). ACM, New York, NY, 87-94.

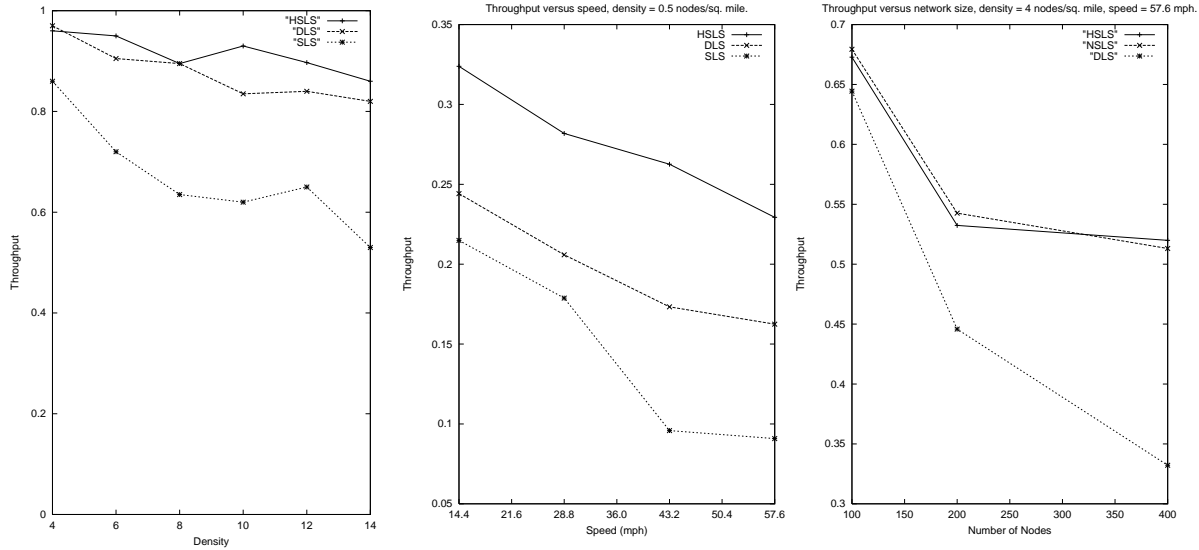[5] "Density and Asymmetry-adaptive Wireless Network (DAWN)." BBN Technologies. http://www.ir.bbn.com/projects/dawn/dawn-index.html

Figure 4: Throughput results for different network sizes and platforms. **(left)** 60-node network results obtained using CPT. **(middle)** 80-node network results obtained using a low-rate CPT emulation. **(right)** 100- through 400-node network obtained using OPNET.

[6] S. Ramanathan; M. Steenstrup. "Hierarchically-organized, Multihop Mobile Networks for Multimedia Support." *ACM/Baltzer Mobile Networks and Applications*, Vol. 3, No. 1, pp 101-119.

[7] R. Ramanathan and R. Hain. 2000. "An ad hoc wireless testbed for scalable, adaptive quality-of-service support." In *Proc. IEEE Wireless Communication and Networking Conference (WCNC)*, September 2000.

[8] C. Santivanez; S. Ramanathan; I. Stavrakakis. 2001. "Making Link State Routing Scale for Ad Hoc Networks.", In *Proceedings of MobiHOC'2001* (Long Beach, CA, Oct. 4-5). ACM, New York, NY, 22-32.