UNIVERSITY OF CALIFORNIA

SANTA CRUZ

A Benchmarking System for Mobile Ad Hoc Network Routing Protocols

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

DANIEL HIRANANDANI

June 2012

The Thesis of DANIEL HIRANANDANI
is approved:

_____

Professor Katia Obraczka, Chair

_____

Professor J.J. Garcia-Luna-Aceves

_____

Professor Daniel Camara

_____

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

A Benchmarking System for Mobile Ad Hoc Network Routing Protocols

by

Daniel Hiranandani

Network simulations are heavily used in the networking community to evaluate the performance of computer networks and their protocols. Simulations are often chosen over alternatives such as live experiments due to limited resources in terms of scalability, as well as reproducibility of the experiments. Many of the routing protocols designed for Mobile Ad Hoc Networks (MANETs) are evaluated solely on their performance calculated by these simulations, but the simulation environments the routing protocols are exposed to are often limited in scope. Only certain aspects of the routing protocols are tested, so the protocols are only understood in terms of the fabricated scenarios that they are subjected to.

We first investigate the current best practices in simulation-based multi-hop wireless ad-hoc network (MANET) protocol evaluation to examine how wide-spread this problem is in the networking community. We extend a prior characterization of the settings and parameters used in MANET simulations by studying the papers published in one of the premier mobile networking conferences between 2006 and 2010. We find that there are still several configuration pitfalls which many papers fall victim to, which in turn damages the integrity of the results as well as any research aimed at reproducing and extending these results. We then describe the simulation "design space" of MANET routing in terms of its basic dimensions and corresponding parameters. We then discuss the benchmark infrastructure that was created to provide an easy to use solution for testing these protocols in a wide range of scenarios. The following chapter looks extensively at the realistic scenarios provided with the benchmark that act as sample scenarios to promote modeling simulations after real-world situations, and to show the flexibility in adding new scenarios. We also propose four "auxiliary" metrics to increase simulation integrity. Next, we show results generated by the benchmarking tool and provide our concluding thoughts.

To my mother, Susan Hiranandani. And to my father, Jack Hiranandani.

I would like to thank my family for their love and support through these years. My father, Jack, has always provided me the inspiration and drive to accomplish any obstacles I encounter. My mother, Susan, who kept me on track with all of her careful planning and forward thinking. My sister, Jennifer, and her family, Justin, Hailey, and Laila, who have always been available with their clear-minded advice, love, support, and friendship. And to my wonderful friends for their insight, encouragement, understanding, and great memories throughout the entire journey.

I would like to thank my advisor, Katia Obraczka, for her wisdom and guidance through this project. To Carol Mullane, for helping me countless times and always eager to do it with a smile. To Tracy Larrabee and Gerald Moulds who helped me to understand the importance and absolute pleasure in teaching. To Daniel Camara for his great insight and direction, even when given embarrassingly short amounts of time.

Finally, I would like to thank my labmates, Kerry, Vladi, Bruno, and Marc who I admire for their talent and willingness to help with any problem. I treasure the many days and nights spent with them for the friendships it created.

1 Introduction

Network simulations are extensively used in the design and evaluation of computer networks and their protocols. There are many reasons why network practitioners and researchers turn to simulations either as an alternative– or complement to actual "live" experiments. Some of the main reasons for the popularity of network simulators are ease of design space exploration, experiment reproducibility, and scalability. This reliance on network simulators makes it critical that simulation scenarios accurately and adequately reflect the real environments and conditions under which the network systems being studied will operate.

Unfortunately, as pointed out by some previous surveys on the topic [4][15][38][39], this is often not the case. Not only that, but in a survey we conducted of the papers published in the ACM MobiHoc conferences between 2006 and 2010, we found that most of the papers that used simulations as their experimental platform do not fully disclose the settings and parameters used[20]. This lack of full disclosure calls into question the quality and reproducibility of the experiments: not only it is not possible for a third-party to reliably achieve the same results but also it questions the validity of the conclusions that are based on the simulation results. As research communities thrive on extending the work of others, the lack of full knowledge of the experimental methodology used by previous efforts is a serious inhibiting factor. Even worse, we fear that the researchers *themselves* do not know what parameters they are using. Relying on default values in a simulator will likely produce different results between simulator versions, and will certainly produce different results when comparing different simulators. Furthermore, these parameter values may not reflect the actual environment and conditions under which the network will be operating.

Additionally, quite often the designers of the protocols are the ones designing the tests by which their protocols are evaluated. Consequently, there tends to be a bias where the developer designs the experiments that will highlight the positive features of their protocols. So it is not always the case that the tests thoroughly expose the protocol to the full spectrum of operating conditions.

At this point, it is interesting to look at how some other disciplines perform experi-

mental evaluations. Benchmarking, i.e., running a set of standard tests for relative performance assessment is widely adopted in computer architecture, VLSI, compilers, and databases, to name a few. We thus contend that similar practice should be adopted within the networking community. The overarching goal of this effort is thus to promote benchmarking as the standard best practice when designing and studying the performance of computer networks and their protocols. As a starting point, we focus on routing protocols for wireless multi-hop ad hoc networks (MANETs). MANETs refer to infrastructure-less networks where there is no functional distinction between hosts and routers: all nodes can originate, sink, as well as forward traffic. MANET routing protocols reflect the wide variety of MANETs which can take on many forms ranging from static, dense, and homogeneous networks to highly mobile, sparse, and even connectivity-challenged networks.

In this thesis, we introduce a benchmarking infrastructure which: 1) makes available reliable, reproducible, and rigorous experimental scenarios, 2) enables viewing and re-using as baseline performance results of other protocols without having to recreate simulation scenarios to reproduce them, and 3) facilitates generating relative performance results for new protocols.

2 Related Work

A few efforts have focused on studying the validity of simulation-based protocol evaluation. However, as will become clear in Section 3, the community as a whole has not been following the recommendations provided by previous work.

For example, the work presented in [23] reported important statistics for the simulation-based papers accepted to the ACM MobiHoc conference up to 2006. This paper was a very important milestone as it brought to light the current best practices in simulation-based evaluation of MANET protocols. Our work leverages on this effort and goes a step further: it shows that current practices in simulation-based MANET protocol evaluation are practically unchanged; it then describes the design space of MANET routing protocols in terms of its fundamental parameters as the basis for the evaluation guidelines for these protocols.

In [24], two key auxiliary metrics to provide feedback on the effectiveness of the scenario being used were introduced - average shortest path hop count and average network partitioning. The proposed metrics are periodically measured over the duration of the simulation. Between these two auxiliary metrics and a third one recently introduced by [27], the average neighbor count, a researcher can identify if their simulation scenario has too few or too large of average hop count distances, too little connectivity which leads to network partitioning, and too dense or too sparse of a network. These auxiliary metrics do provide excellent information about a scenario; in this paper, we propose five additional metrics that capture important information about the simulation scenario. These metrics are described in Section 4.

The use of simple models is proposed in [38] which surveyed the papers published in the ACM MobiHoc 2008. They found that 59% of the papers did not run meaningful comparative studies: they either did not compare against "truly competing solution(s)" or did not compare their solution(s) to any other protocol. We argue that we can eliminate this problem with our future work aimed at creating a publicly-available standardized set of routing protocol benchmarks.

The work in [29] reports a survey similar to [23] in which they studied 280 papers on simulations of peer-to-peer systems and found that 71 papers did not even state which simulator was used.

Five principles are presented in [21] which reinforces the importance of having repeatable, rigorous, complete, statistically and empirically valid simulations. It stresses the need for researchers to include all parameters and configurations used in their experiments.

In [11], the implementation differences of IEEE 802.11 was studied by comparing two different simulators as well as the differences in 802.11 within multiple ns-2 versions. This study finds that the results between simulators are quite different (although they also find that the difference is minimal when the ns-2 802.11 MAC is ported into OMNeT++).

The work described is [1] also finds in 2006 that very little has changed in the MANET community in terms of simulation-based evaluation methodology. Almost

90% of papers do not even specify the simulator version, and over half do not specify the number of simulation runs. These omissions make accurately reproducing and re-using previous results near impossible.

The continuum proposed in [7] aims to identify and classify types of mobile ad hoc networks. They focus on node speed and node density for classification.

In the area of CPU benchmarking, the Standard Performance Evaluation Corporation (SPEC) [9] has a standardized suite of tests that evaluate the performance of a processor. This suite of tests is comprised of many smaller tests using various real-world applications including compression, compiling, discrete event simulation, and speech recognition. When new processors are vetted against previous processors, these SPEC tests are run on the processor which produces a single numeric value that can be compared to the other processors to determine a hierarchy of rank. We do not feel that a single value is telling enough to describe a routing protocol's performance, however similar to [16], the idea of running a battery of standard tests on different routing protocols and having measurable as well as comparable metrics is the basis of our benchmark.

There have been numerous projects and studies advocating the use of realistic mobility models and traces [19][13] [22][32] [2]. In our work we go a step further and introduce a benchmarking system that promotes not only the use of realistic simulation scenarios that mimic real applications, but also enables reusing and reproducing results as baseline for comparison among new and existing protocols.

Other MANET routing protocol benchmarks have been explored previously [26], but they do not provide an infrastructure that allows for the sharing and re-use of simulation scenarios and results.

3 Current MANET Simulation Best-Practices

In order to characterize the current state-of-the-art in evaluating MANET protocols, we conducted a survey of the full papers accepted into the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) for the years 2006 through 2010. We chose to study MobiHoc because of the prior surveys which identified the

common problems in papers accepted into previous editions of the same conference. We also focused on MobiHoc as it is regarded as a highly selective and prestigious conference in the MANET community. We consider the results of this survey to be a "best practices" survey in that they represent the configurations used by some of the best papers published in the MANET community. Although we were able to find 12 of the 25 papers submitted to the most recent MobiHoc 2011, we chose to omit all of the MobiHoc 2011 papers from our survey to preserve completeness of each year's worth of papers.

In addition to the MobiHoc survey, we extended the survey to include 82 papers which simulated routing protocols from a variety of conferences to get a broader idea of what the "average" MANET routing protocol simulation scenario looks like. Besides broadening the universe of papers surveyed, we extended the survey outside of MobiHoc for a number of other reasons, including: fewer routing protocol papers have been published in MobiHoc in the recent years compared to the previous study; few papers actually specify their configurations and parameters out of the routing papers published in recent MobiHoc years (which further demonstrates the issue at hand).

3.1 MobiHoc Survey

Out of the 159 MobiHoc papers we reviewed, while we noticed that simulation is still an often-used tool to evaluate protocol performance (105 papers or 66.0% used simulations), only 47 out of the 105 simulation-based MobiHoc papers specified which simulator was used (44.8%). Comparing the popularity of individual simulators used in these published papers to a survey of the 2000-2005 MobiHoc papers that used simulation-based evaluations [23], we found that Network Simulator 2 (ns-2) tied with custom simulators (13 of the 47 simulated papers for each, or 27.7% each). This is quite different from the previous study which ranked ns-2 at 43.8% and self-developed simulators at 27.3%. Matlab, however, had the largest increase in declared popularity moving from only 3.8% proclaimed-usage in the previous study to 21.3% in our survey. Figure 1 shows the per-simulator usage breakdown.

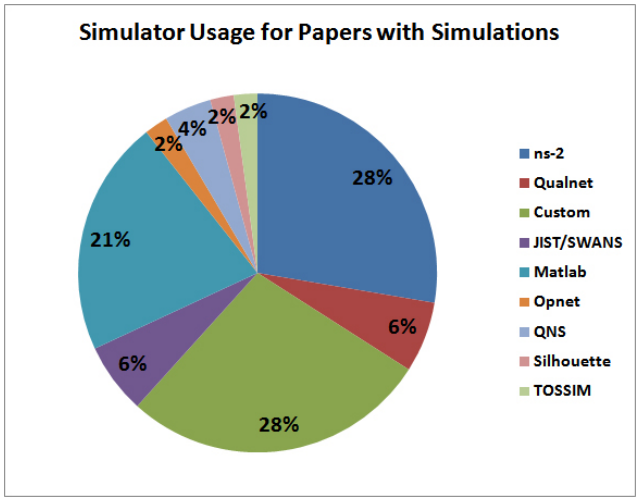

Figure 1: Breakdown of simulators used in MobiHoc papers from 2006-2010.

Some of the more startling results show that out of the 25 papers which identify the mobility model used, 8 papers (32%) use the Random Waypoint Mobility Model which has been shown to exhibit undesirable behavior and thus produce unreliable results [41]. Additionally, of the 15 papers which identify how source and destination pairs were chosen to be the source and sink of a flow, random selection was the most popular at 86.6% which has two major issues. The first is that there is no guarantee on the minimum number of hops for the path between the source and the destination; thus, the source and destination pairs can potentially be next to each other and traffic between them does not require any routing. Secondly, the nodes that are chosen might produce traffic flows that do not overlap. Overlapping flows stress the network's ability to handle multiple flows in terms of queueing and processing power, so scenarios with zero concurrent flows could produce artificially optimistic results.

We also noticed that the papers surveyed tend to run a single scenario multiple times. While we certainly encourage such practice to reduce the effect of outlying results, we need to address the problem of using the same Pseudo Random Number Generator's (PNRG) seed. It was noticed that in ns-2 which uses a fixed Pseudo-Random Number Generator (PRNG) seed of 12345, re-running simulations without ever changing the seed will produce identical results [23]. Nevertheless, this problem is still quite prevalent today. To quantify the gravity of this issue, of the 39 papers

that declared running more than one simulation run, only 5 papers (12.8%) addressed changing the PNRG seed for each run.

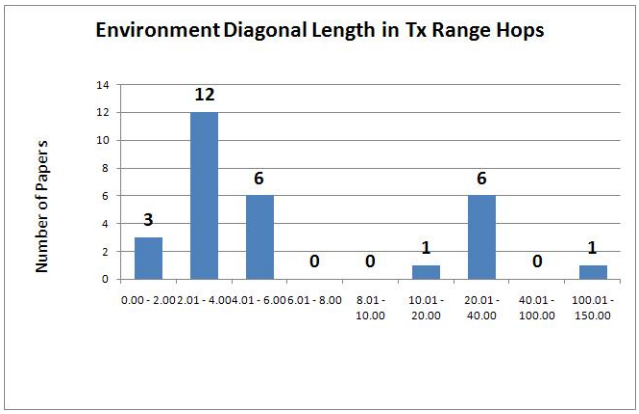

Figure 2: Measuring MobiHoc 2006-2010 simulation scenario environment area in terms of the number of transmission range hops across the area's diagonal.

As a measure of how arbitrary topologies are chosen, we measured the longest distance in an environment (the diagonal of the rectangular environments) in terms of a node's transmission range. This distance spans the longest point-to-point distance in the environment, and roughly shows the maximum possible diameter of the network. Figure 2 shows that the values range from as little as 0.35 hops all the way up to 141 hops.

In order to better characterize the quality of scenarios used, we also calculated one of our proposed auxiliary metrics, the Average Node Density of the network which is described in more detail in Section 7, for the papers that listed the Environment Dimensions, Number of Nodes, and Node Transmission Range. The results are shown in Table 3 sorted by density, and they show a very wide range of values ranging from 400 nodes per cell down to 0.025 nodes per cell.

A fairly common occurrence was for papers to declare that they were unable to provide details on the simulation configurations due to the "constrained space" of the paper. We realize that this is an understandable concern, however it is not acceptable. It is, however, perfectly acceptable to include a URL that links to the research group's or individual's website that contains information about the simulations. Taking this one step further, we also encourage including contact information on the website to give others a way to obtain the code used in the simulations.

As a fitting example, while we lack the space in this paper to provide the full results of the MobiHoc survey, we show a snapshot of the results in Table 1, Table 2, and Table 3. We also invite the readers to visit `http://inrg.cse.ucsc.edu/` to view the more detailed version of these statistics.

Table 1: MobiHoc 2006-2010 Simulation Survey

Totals	Percentage	Description
105 of 159	66.0%	Used simulation for protocol evaluations
47 of 105	44.8%	Specified which simulator was used
13 of 47	27.7%	Used ns-2 as their simulator
3 of 47	6.4%	Used Qualnet as their simulator
13 of 47	27.7%	Used a Custom simulator
3 of 47	6.4%	Used JIST/SWANS as their simulator
10 of 47	21.3%	Used Matlab as their simulator
1 of 47	2.1%	Used Opnet as their simulator
2 of 47	4.3%	Used QNS as their simulator
1 of 47	4.3%	Used TOSSIM as their simulator
1 of 47	2.1%	Used Silhouette as their simulator
5 of 105	5.8%	Stated the simulator version
25 of 105	23.8%	Used mobility
8 of 25	32.0%	Used the Random Waypoint Mobility Model
3 of 25	12.0%	Used the Random Walk Mobility Model
2 of 25	8.0%	Used the Brownian Motion Model
1 of 25	4.0%	Used the Random Direction Mobility Model
4 of 25	16.0%	Used other mobility models
7 of 25	28.0%	Used Mobility Traces
1 of 7	14.3%	Used the UMass DieselNet Trace
2 of 7	28.6%	Used the MIT Reality Trace
1 of 7	14.3%	Used the Intel Labs Trace

Continued on next page

Totals	Percentage	Description
1 of 7	14.3%	Used the Infocom DTN Trace
1 of 7	14.3%	Used the Haggle Traces
2 of 7	28.6%	Used other (unnamed) traces
34 of 105	32.4%	Declared node distribution methods
14 of 34	41.2%	Used Random Distribution
10 of 34	29.4%	Used Uniform Distribution
2 of 34	5.9%	Used Perturbed Grid Distribution
5 of 34	14.7%	Used Poisson Distribution
2 of 34	5.9%	Used Clusters
1 of 34	2.9%	Used Power-law Distribution
9 of 105	8.6%	Stated Packet Rate Distribution
7 of 9	77.8%	Used CBR Flows
2 of 9	22.2%	Used VBR Flows
15 of 105	14.3%	Stated Source-Destination Pair Selection
13 of 15	86.7%	Used Random Selection
2 of 15	13.3%	Used Fixed Selection
51 of 105	48.6%	Stated the Environment Dimensions
37 of 105	35.2%	Stated the Radio Transmission Range
66 of 105	62.9%	Stated the Number of Nodes
38 of 105	36.2%	Stated the Number of Simulation Runs
6 of 105	5.7%	Used multiple PNRG seeds
7 of 105	6.7%	Accounts for Steady-State (Traffic, Mobility, or both)
66 of 105	62.9%	Stated Metrics collected
16 of 66	24.2%	Measured Delivery Ratio
19 of 66	28.9%	Measured Delay
4 of 66	6.1%	Measured Overhead
12 of 66	18.2%	Measured Throughput
5 of 66	7.6%	Measured Average Hop Count

Continued on next page

Totals	Percentage	Description
5 of 66	7.6%	Measured Average Node Degree
4 of 66	6.1%	Measured Energy Consumption
1 of 66	1.5%	Measured Network Diameter

Table 2: MobiHoc 2006-2010 Simulation Parameter Values Survey

Min	Average	Max	Standard Deviation	Description
5	1,027.11	10,000	1,027.11	Number of Sim. Runs
1	138,371	10,000,000	1,095,000	Num. of Nodes
1m	1,035.91m	10,000m	1,682.10m	Environment Width
1m	1,259.49m	20,000m	2,852.32m	Environment Height
1:1	1.31:1	15:1	1.96	Env. Aspect Ratio (H:W)
0.20m	152.91m	600.00m	174.32m	Node Tx Range

3.2 Extended Survey

In this extended survey, we noticed that a few patterns emerged in the values of parameters chosen, and we list them in Table 4. We found that the following is an appropriate "average simulation" in the sense that these were the most-often occurring values of parameters in our study.

We can see from Table 4 that the average scenario randomly places nodes in the environment, randomly moves them, and then randomly selects which nodes will communicate with each other. We do believe in adding randomness to simulations to introduce variance, but we feel that randomizing all of these aspects will make the scenario overly synthetic, and thus far from reality.

Table 3: MobiHoc 2006-2010 Avg Densities

#Nodes	Dimensions	Tx Range	Avg Density
10000	1500m x 1500m	150m	400.000
20	300m x 300m	600m	320.000
1000	500m x 500m	120m	230.400
100	1000m x 1000m	600m	144.000
300	500m x 500m	100m	48.000
512	4000m x 4000m	600m	46.080
160	100m x 100m	20m	25.600
50	100m x 100m	32m	20.480
20	500m x 500m	250m	20.000
250	300m x 300m	40m	17.778
100	1250m x 1250m	250m	16.000
1000	4000m x 4000m	250m	15.625
150	600m x 600m	88m	12.907
3000	610m x 610m	20m	12.900
50	1000m x 1000m	250m	12.500
80	600m x 600m	100m	8.889
50	1m x 1m	0.20m	8.000
54	100m x 100m	18m	6.998
64000	1000m x 1000m	5m	6.400
500	200m x 200m	6m	1.800
100	100m x 100m	3m	0.360
100	5000m x 5000m	150m	0.360
10	4000m x 4000m	100m	0.025

Table 4: Extended Simulation Survey "Average Scenario"

Value	Parameter
ns-2	Simulator Used
900 seconds	Simulation Duration
1000m x 1000m, 1500m x 300m	Env. Dimensions
50 or 100	Number of Nodes
250m	Node Transmission Range
802.11	MAC Protocol Used
Random	Initial Node Distribution
Random Waypoint	Node Mobility Model
512 Bytes	Packet Size
4 Packets per second	Packet Send Rate
Random	S-D Pair Selection
20	Number of Traffic Flows

4 Simulation Design Space and Guidelines

In this section, we lay out the basis for specifying a set of guidelines to standardize the evaluation of MANET routing protocols. Such guidelines will enable not only

accurate and unbiased performance assessment, but also sharing of tools and results. We start by identifying the main performance "dimensions" of MANET routing, namely: topology, traffic, and mobility. We have found that many of the values used in simulation studies are largely synthetic, i.e., they are not consistent with real-world scenarios. Since the values of these parameters largely depend on the driving application(s), we provide some example real-world situations along with logical ranges of values that could be used when simulating them.

4.1 Topology

The topology used by simulation experiments describes physically the environment where the simulation takes place. It typically includes: width, height, terrain, channel characteristics (e.g., path loss, fading, etc.), number of nodes, the nodes' transmission ranges, and the method for determining where to place the nodes (node distribution/placement).

These parameters are often modified individually to produce "new" simulation scenarios. However, some parameters may have a "collective" effect. This is the case, for example, of number of nodes, size of the area, and transmission range, all of which affect the density of the network being simulated, which, in turn, can have significant impact on the performance of MANET routing.

Examples have shown how easy it is to create two scenarios that are effectively measuring the same things when changing the environment size, average node speed, and transmission range, and how important it is to identify the environment size and node speed in terms of the transmission range of the node [23]. We propose that this should be taken one step further to include the number of nodes in these measures as well. Since the environment size, node speed, and transmission range effectively describe the node density of a scenario, it is logical to see that the number of nodes plays a significant role in the density as well. When designing a scenario, careful thought should be put into these four parameters because they are not independent of each other - they all affect the network density.

4.2 Traffic

Modeling traffic includes parameters such as when nodes start and stop sending data, the number of traffic flows, the selection process for determining source and destination nodes, and the packet rate distribution. While values assigned to these parameters should reflect the driving application(s), there are a few guidelines that should be followed when selecting values for these parameters.

As pointed out in Section 3, the most common method for choosing traffic source-destination pairs is random selection. Therefore, it is important to note some of the adverse effects this methodology can introduce. For example, selecting source-destination pairs at random may cause, as side-effect, the number of hops in the path to be abnormally small or large. It may also mean that no node might ever have to route for more than one flow at a time. However, it is necessary to subject network protocols in general, and routing in particular to heavier as well as non-uniform traffic loads. Therefore, other traffic source-destination selection policies in addition to uniform selection need to be employed. We also propose an additional auxiliary metric that will measure the average number of concurrent flows in the nodes along the routed paths. We recognize that this auxiliary metric will provide a coarse-grain analysis of the problem, however it will address the issue of not even knowing whether scenarios contain any overlapping flows at all.

4.3 Mobility

Mobility models, like topology and traffic, are very application dependent, and while some mobility models are more popular than others, they are not necessarily the best models to use. Take for example Random Waypoint Mobility: it is still the most used mobility model, but it does not produce realistic movement for applications such as human walks [27]. Therefore, using mobility models such as Self-similar Least Action Walk (SLAW) would be preferred instead. Additionally, there are traces such as MIT Reality [12], UMass DieselNet [17], and others available through CRAWDAD [8] which can provide realistic node mobility.

5 Benchmark Infrastructure

The foundation of this benchmark relies on evaluating the performance of routing protocols by exposing them to many different types of scenarios. Taking the entire parameter design space into consideration, which we discuss in [20], we initiated this benchmark with a select few sample scenarios that exemplify different facets of the design space. It is far too common to get a one-sided perspective of a routing protocol's performance by only subjecting it to one, or several similar scenarios, so this infrastructure allows us to paint a clearer picture of a protocol's response to a wide variety of environments.

There were several goals that we needed to meet in the creation of this benchmark infrastructure. First and foremost since it relies on having many scenarios stressing different aspects of a protocol, it must be easy to add new scenarios. Secondly and just as important as the first it must be easy to run the simulations as well as to gather metrics. Our third objective was to be able to directly compare results between different routing protocols which also brought about a fourth goal of eliminating the need for researchers to implement other researchers' previously created scenario environments. Lastly, to easily facilitate the distribution of the new scenarios and their results.

The prototype version of this infrastructure was built inside of the Network Simulator 3 (ns-3) simulator, but there is no reason that it could not be implemented in any other simulator. Due to the differences inherent in different simulators, however, we foresee direct comparisons as only being possible within the same simulator due to varying implementations of protocols and environmental models.

This benchmark contains several example scenarios (Smart Energy Grid, MASE Seismic Monitoring, Campus DTN, and SF Taxi VANET), but these scenarios are intended to act as reference points – not as an exhaustive performance evaluation suite. The purpose of this work is to facilitate the ease of adding new benchmark scenarios so that researchers can add their own scenarios so that the entire parameter design space can eventually be tested for each protocol.

We provide a full download of the source code which can be found at

`http://inrg.cse.ucsc.edu/`.

5.1 Code Structure

The structure of this benchmark is broken up into several pieces: a run script, scenario-common code, scenario-specific code, scenario-specific scripts, and processing scripts.

The run script, also known as the Main Benchmark Script, is what invokes the benchmark. This script is responsible for calling all other scripts to set parameters, build scenarios, run scenarios, calculate results, and for generating plots.

The Main Benchmark Script first calls the scenario-specific scripts which iterates through any dynamic parameters (parameters that are desired to change between runs of the simulation such as traffic rate) and calls the ns-3 build/run system. ns-3 is built for that scenario using its scenario-specific code along with the current dynamic parameters. Since there are many parameters and algorithms that are shared between scenarios, a scenario-common code file is used to reduce the amount of duplicate code.

The scenario-common code contains code that is needed by all scenarios. This primarily includes code for setting up and initializing ns-3 in terms of the simulator itself, common MAC/PHY parameters, source-destination pair selection algorithms, as well as metrics and auxiliary metrics calculating functions. This file also contains the *main* function which in turn calls the functions that set scenario-specific parameters specified in the scenario-specific code files.

The scenario-specific code files contain functions that set parameters for a specific scenario such as the number of nodes, number of sources, environment dimensions, node transmission range, etc. They also contain mobility model/trace information as well as settings for the traffic. These functions and parameters need to be set for each scenario, but since the values will be different between scenarios, they need to be stored in files that are only intended to be used by that particular scenario.

After ns-3 has finished running a simulation, the scenario-specific script calls a processing script which renames files according to their dynamic parameters, stores

them into folders organized by the scenario name and PNRG seed, runs a preliminary metrics-gathering script, and removes files that are no longer needed.

The preliminary metrics-gathering script processes the FlowMonitor file that is generated by ns-3 and it produces results such as Throughput, Delivery Ratio, Delay, and more. These statistics are stored in a file that is uniquely named by the dynamic parameter combination, and is processed again after all simulations have been run. All of the statistics files are processed after ns-3 concludes its simulations because this allows us to perform an analysis on all of the results at once. The statistics post-processing script finds all of the statistics files related to a scenario's dynamic parameter combination, and for each PNRG seed that was used, averages the results as well as record the min/max values for each metric. These values are output into plot files which are processed by a plotting script to produce visual graphs of the averages of the data as well as error bars to indicate the variance across simulation runs. These plot files are created by starting with preset "base files". These base files, one for each metric, contain information such as the title of the graph and axis labels and are copied into the plot directory when the statistics post-processing script is first run. The plot files are then appended with the averaged statistics as they are calculated, and an error bars file is simultaneously maintained which holds information for creating the error bars on the plots using the min/max data. After the statistics post-processing script has added the averaged values for each scenario, the entire error bars file is appended to the plot file which is then read by the plotting script.

The final plotting script [3] is built on top of gnuplot and was used in this benchmark because it allows for easy plotting of clusters. The goal of this benchmark is to compare the performance of routing protocols, and this script produces exactly that - graphs where the scenarios are clustered so that the routing protocols can be compared side-by-side.

5.2 FlowMonitor

FlowMonitor is a module in ns-3 that is used to collect commonly used performance metrics [5]. The implementation of this benchmark in ns-3 uses FlowMonitor extensively to compute its metrics. FlowMonitor captures information about "flows", which we usually take to mean layer 4 end-to-end connections. FlowMonitor is less exclusive in its definition of a flow, and while it usually takes on the layer 4 end-to-end meaning, it can also represent other types of "flows" such as layer 2 hop-by-hop flows.

This became a challenge over the course of the project because FlowMonitor was counting both end-to-end and hop-by-hop flows in its output file. The scripts in this benchmark initially did not distinguish between the two types of flows, so many of the metrics showed very strange results. There were many more hop-by-hop flows, the number of packets sent in those flows far outnumbered the number of packets sent in the end-to-end flows, so the results were heavily skewed. In order to overcome this issue, the benchmark scripts were modified to construct a list of "relevant flows" before processing the data in the FlowMonitor file by discriminating based on the destination port of the flows.

In ns-3, when traffic sinks are declared, a port is specified for that node to receive traffic on. FlowMonitor was showing in its output file that the end-to-end flows were indeed using the same port as was specified in ns-3, but the hop-by-hop flows used an entirely different port. Since the use of different ports is what distinguishes a "relevant flow" from a non-relevant one, the FlowMonitor output file is first scanned to remember the IDs of the flows that are relevant. Once this list has been constructed, the rest of the data in the FlowMonitor output file is processed while only counting the statistics measured for those relevant flows.

5.3 Running Simulations

The process that this benchmark system uses is shown in Figure 3. Running the benchmark is performed by executing the Main Benchmark Script which holds a list of pseudo-random number generator seed values to use, a list of the names

of the routing protocols implemented in the simulator, automatically finds all of the scenarios to run, and calls the scenario-specific scripts which contain dynamic parameters that are iterated through for successive simulation runs such as packet size and traffic rate.

New parameters can be added either as static parameters (ones that will stay constant for each time the benchmark is run), or as dynamic parameters (ones that require the simulation to run for each new dynamic parameter combination).

Since static parameters are contained within the scenario-specific file which contains code to set the scenario-specific parameters, the process to add new static parameters is the same as adding new parameters to any C++ file.

Dynamic parameters hook into the ns-3 command-line argument system, so only a few things need to be modified to add new dynamic parameters. First, the parameter should be added to the *manet-routing-benchmark.cc* file which holds code common to all scenario files including metric and auxiliary metric functions. The parameter needs to be added to this file with the *cmd.AddValue()* function which is run on an instance of *CommandLine* to let ns-3 know that is it a valid command-line argument. Next, the dynamic parameter should be added in the scenario's shell script. If the dynamic parameter needs to take on different values for each run, then it should be added in a loop such that the ns-3 build/run system is called for each value of the parameter.

As an example, with packet sizes of 512 and 1024Bytes, and traffic rates of 16384bps and 32768bps, there would be four combinations of packet sizes and traffic rates: 512-16384, 512-32768, 1024-16384, and 1024-32768. Each of these packet size/traffic rate combinations would be run 10 times for each scenario, once for each seed specified, in order to produce reliable results.

There are 10 seeds stored in the Main Benchmark Script so that each scenario is run 10 times for each combination of dynamic parameters. We do not use randomly generated seeds for the sake of reproducibility - running the benchmark multiple times should and will produce the same results as prior runs.

This script calls the ns-3 build/run system which links the scenario-specific param-

eters and setup code with the common parameters and metric-producing code. The dynamic parameters are passed to the ns-3 run system through command-line arguments, and the scenario is run. There are several files that are generated in running the simulation such as the Average Node Connectivity, Average Node Density, and FlowMonitor files, and these files are moved to the respective output directory for that scenario after the simulation has finished running. Files generated by this benchmark are named accordingly to indicate the scenario, routing protocol, packet size, and traffic rate used. The process is then repeated until all dynamic parameter combinations have been evaluated. Lastly, the raw output files are run through a metrics processing script which produces human-readable results.

The benchmark will automatically run any simulations that have not previously been run in order to reduce processing time. It will also run through the scenarios using all of the routing protocols that are specified in the main benchmark script.

5.4 Gathering Metrics

Some metrics are generated directly by the benchmark itself and are output into their own files (Average Node Connectivity and Average Node Density), while others are calculated in post-processing by a script. This script analyzes the FlowMonitor file that is created by ns-3, and running this script outputs the Average Number of Concurrent Flows, Number of Dropped Packets, a breakdown of the reason codes for those dropped packets, the Total Delivery Ratio, Total Average Throughput, and Total Average Delay.

These statistics are stored in a separate file which is read by an additional statistics aggregation script. The reason these statistics are stored separately is because this allows the benchmark to evaluate all of the scenarios for all of the routing protocols before going back and computing the results for each individual scenario. This aggregation script reads all of the statistics files for a scenario with a particular packet size and traffic rate combination and averages the values over the different runs of that scenario. In other words, for each PNRG seed that was used in running a scenario, this script will average the metrics together to produce files which are

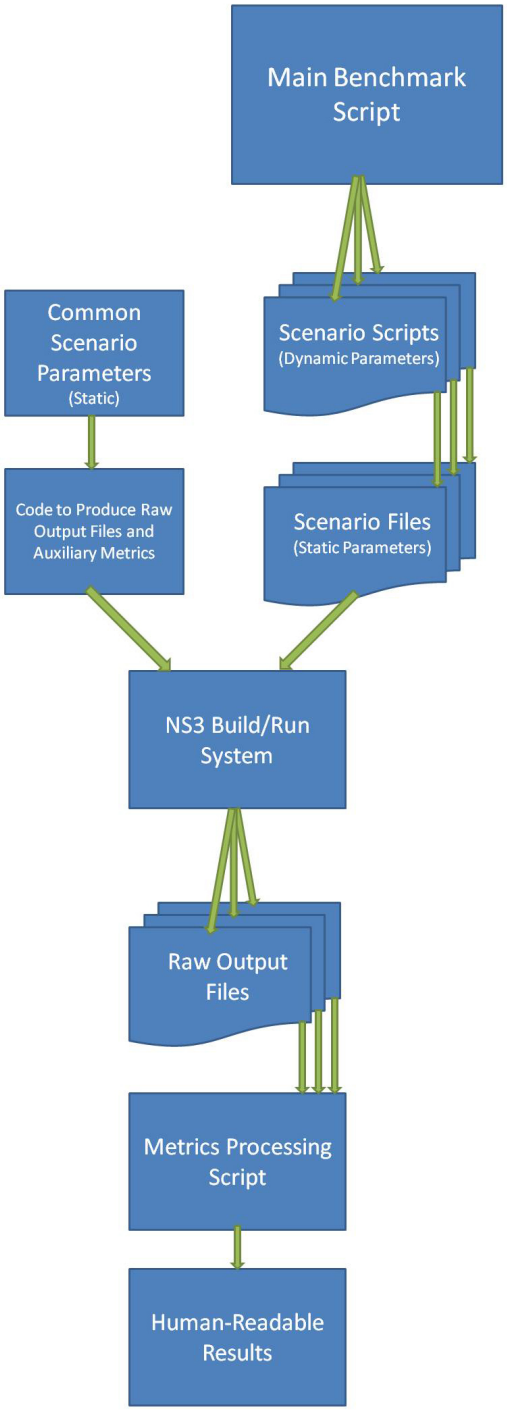

Figure 3: The order of events that occur in this benchmark system.

able to be plotted automatically.

5.5 Adding New Scenarios

New scenarios can be created using the sample scenarios as guides, and all that need to be modified are: the environment, traffic, and mobility parameters. Scenario files also require a supporting shell script which allows for parameters to be specified that should change on each run of the scenario, such as packet size. Mobility models and mobility traces are supported, and all files associated with the scenario (scenario file, scenario script, and mobility trace) should be placed in the same directory with the rest of the example scenarios so that the main benchmark script can automatically find them.

This benchmark has support for selecting random Source-Destination pairs with the guarantee that the destination will not be the same node as the source, along with manually specifying your own Source-Destination pairs. The random algorithm selects $numSource$ nodes to act as sources, and then randomly chooses $numSinks$ sink nodes for each source node. For the reason that a sink node cannot be a source node, the most number of sinks that can be chosen for a source node is $numNodes-1$. After a new scenario has been added (the file containing ns-3 code which sets the static parameters, the shell script which includes the dynamic parameters, as well as any additional files such as a mobility trace), only the *wscript* file in the scenario directory needs to be modified to evaluate the scenario. This file contains information about which files and modules the scenario is dependent on in terms of compiling and linking, so a new entry will need to be added for each new scenario. After *wscript* has been updated, the Main Benchmark Script will find the new scenario the next time the benchmark is run, and will automatically test each routing protocol against it.

5.6 Adding New Routing Protocols

There are only two places in this benchmark that reference the routing protocol - in the main benchmark script which specifies the name of the protocol, and in the

core benchmark code that sets up and configures the routing protocol through the routing protocol helper. All that is required to add a new routing protocol is to add the name of the routing protocol to the Main Benchmark Script as a "supported protocol", and then to add the routing protocol helper to the SetupRoutingProtocol function inside of the core benchmark code. This function maps the name of the routing protocol, which is passed in as a command line argument, to the routing protocol helper, so it is necessary that these names match. Once these two updates are made, the routing protocol will automatically be run against all of the existing scenarios the next time the benchmark script is called.

5.7 Sharing Scenarios and Results

Because this infrastructure is built into the examples that are provided in the standard distribution of ns-3, it will be very easy to share new scenarios that are added to the benchmark. Code that is added to the distribution, including these new scenarios, must be peer reviewed and checked into the ns-3 codebase where they will be stored with the existing set of benchmark test scenarios. Sharing results is something that can also be accomplished in the same way that new scenarios are shared, or in an effort to save memory, they can be omitted from the codebase and require the user to run the simulations for those scenarios. Due to the design of this benchmark system, the benchmark can be run by any researcher who downloads it and will produce the same results as if they were included with the ns-3 distribution.

6 Example Scenarios

We provide four example scenarios - two static and two mobile - in this section with the intent of showing that it is possible to create scenarios that are applicable to real-world situations. Instead of arbitrarily choosing parameter values, as have been shown in the previous surveys [20], we urge the importance of evaluating protocols with meaningful simulations. Our previous work described the parameter design space in which parameters are selected from to create scenarios, and we chose these four scenarios to represent different areas within the design space. These scenarios

alone are in no way meant to be a comprehensive evaluation of a routing protocol, but they will be a part of the comprehensive test as new scenarios that stress different aspects of a protocol are added to the benchmark.
A full listing of the parameters chosen can be found at
`http://inrg.cse.ucsc.edu/`.

6.1 Static Scenarios

Static scenarios typically give benefit to proactive routing protocols, because the positions of the nodes will not change over time. This allows for a higher amount of traffic to be used early on in the simulation to determine the best possible routes that can be used for the duration of the simulation, provided all nodes stay connected.

6.1.1 Smart Energy Grid

The first scenario, whose parameters are shown in Table 5, is a section consisting of four city blocks (2x2) equipped with smart energy meters that are able to wirelessly communicate energy usage data back to a central data collector unit [18]. The density of the environment will depend on how many houses or apartment units are built on a city block, but we can estimate 15 housing units per acre [6], and we can estimate a medium-sized city block as 125m x 125m [37] which gives us approximately 60 housing units per city block. The energy data sent back to the hub is quite small at 512-1024 Bytes for commands and meter registers [25], and it is also fairly infrequent seeing as the meters would not need to update more than a few times per hour. Since multiple meters are capable of sending data at the same time, the possibility of concurrent flows increases for the nodes closer to the central data collector unit. Since there is a central unit acting as a data sink, the destination node is always fixed, but the source nodes can be chosen at an estimated one every hour.
This scenario tests a protocol's ability to determine multi-hop routes in a very dense network. Since the amount of data transferred is quite low, but done by many nodes, we expect to see a significant amount of signaling packets being sent in this scenario.

If a routing protocol is excessive in how it handles its network overhead, then it will perform poorly in this test.

Table 5: Parameters Selected for Smart Energy Grid Scenario

Value	Parameter
3600s	Simulation Duration
250m x 250m	Environment Dimensions
240	Number of Nodes
240	Number of Sources
4	Number of Destinations
All Sources, Fixed Sinks	Source-Dest Pair Selection
Fixed	Initial Node Distribution
N/A	Node Mobility Model
25m	Node Transmission Range
512-1024 Bytes	Packet Size
5 Packets per Hour	Per Node Packet Send Rate

6.1.2 Earthquake Monitoring

This scenario, as illustrated in Table 6, is modeled after the Middle America Subduction Experiment (MASE) [31][36]. MASE is a sensor network that monitors seismic activity in Mexico, and reports the data back to the collaborating research labs for processing. This network consists of 100 nodes spanning 550km from Acapulco, Mexico to Tampico, Mexico, each of which is equipped with an 802.11 radio [30]. Looking at a map of the locations of the nodes, we can estimate that the width of this environment is approximately 50km since the nodes are laid across a fairly straight line. We were unable to find specific data regarding the traffic these nodes send, but we are able to estimate the values based on reports of 20-40MB of bandwidth per node per day [30] with an estimated 5 minutes of transmit time per hour when in low power mode [10].

This scenario is similar to the Smart Energy Grid scenario in that it is has static nodes, however it is an extreme situation where all of the flows are concurrent and overlapping. Each of the source nodes sends its data to a single sink node at the end of the line of nodes, so the nth node's traffic flows through n-1 nodes. This scenario

will also favor proactive protocols since the positions will remain static throughout the duration of the simulation.

Table 6: Parameters Selected for MASE Seismic Monitoring Scenario

Value	Parameter
900s	Simulation Duration
50km x 550km	Environment Dimensions
100	Number of Nodes
100	Number of Sources
1	Number of Destinations
All Sources, Single Fixed Sink	Source-Dest Pair Selection
Fixed	Initial Node Distribution
N/A	Node Mobility Model
6500m	Node Transmission Range
512-1024 Bytes	Packet Size
8 Packets Per Second	Per Node Packet Send Rate

6.2 Mobile Scenarios

Unlike static scenarios, mobile scenarios expose weaknesses in proactive routing protocols because often times the routes that proactive protocols find become stale and obsolete very quickly. Reactive routing protocols, on the other hand, will determine a route as data needs to be sent, so they are much more flexible in terms of adapting to these changing environments.

6.2.1 College Campus DTN

This scenario uses the CRAWDAD KAIST mobility trace that were collected by recording the movement of students on the KAIST college campus [34]. The mobility trace is a collection of position recordings for 61 nodes over the course of over 22 hours and in a 3900m x 8700m area. Due to this being a mobility trace monitoring real people, not all of the 61 nodes were active over the course of the entire simulation, so we used a snapshot of the trace that represented the time with the highest amount of node movement. To select this section of time, we created a script to first discretize time into buckets, then count the number of movements made by unique nodes,

and then select the sequence of buckets with the highest number of movements for unique nodes. This limited the scope of the trace to a section of one hour in length which is sufficient for this study because we intend to evaluate how routing protocols operate in Disruption Tolerant Networks (DTNs), not specifically to gather results about the entirety of this mobility trace.

Since this scenario is modeling communication between students on a college campus, we set the traffic parameters, which are shown in Table 7, to describe each node sending data to every other node - such as in an emergency type of situation. We equip the nodes with a transmission range of up to 200m which reflects the maximum transmission range of Bluetooth [28]. We configure the nodes to send messages that are 160Bytes, which is approximately the size of a GSM SMS text message [14].

Table 7: Parameters Selected for Campus DTN Scenario

Value	Parameter
3600s	Simulation Duration
1000m x 1000m	Environment Dimensions
19	Number of Nodes
19	Number of Sources
18	Number of Destinations
All Sources, All Sinks	Source-Dest Pair Selection
Fixed	Initial Node Distribution
Mobility Trace	Node Mobility Model
250m	Node Transmission Range
160 Bytes	Packet Size
2 Packets Per Minute	Per Node Packet Send Rate

6.2.2 San Francisco Taxi

This scenario uses the CRAWDAD mobility traces which captured the positions of San Francisco taxis while they were operating [33]. The full trace contains coordinates of 536 nodes spanning a 3600km x 4200km area over 575 hours. This amount of mobility data not only is significantly more data than we need for this test, but we also suspect at least a few of the coordinates were stray positions that are intended to be filtered out since the size of the entire is many times larger than the San Francisco peninsula, as shown in Figure 4. To restrict the trace data, we targeted nodes

in just the San Francisco peninsula, as shown in Figure 5, which resulted in a 9500m x 6230m area. We also used a similar process like was done with the KAIST trace to reduce the amount of the trace to 900 seconds from the full 575 hours, and this limited the number of active nodes in the area to 283. The period of 900s that was selected was a sequence of time that had a large amount of unique node movement so as to keep the number of active nodes high.

The traffic generation parameters are based on an experiment modeling the VANET traffic in downtown Malaga, Spain [40]. The number of Source-Destination pairs chosen were half the number of the nodes used, with CBR traffic ranging from 33Kbps to 1Mbps data rates and 512Byte packets. The transmission range was chosen as a stable value based on the results in [35].

Table 8: Parameters Selected for San Francisco Taxi VANET Scenario

Value	Parameter
900s	Simulation Duration
9500m x 6230m	Environment Dimensions
283	Number of Nodes
283	Number of Sources
10	Number of Destinations
Random Sources, Random Fixed Sinks	Source-Dest Pair Selection
Fixed	Initial Node Distribution
Mobility Trace	Node Mobility Model
750m	Node Transmission Range
512 Bytes	Packet Size
33, 66, 100, 333, 666, 1000Kbps	Per Node Packet Send Rate

7 Metrics

7.1 Metrics Collected

We make a distinction between metrics and auxiliary metrics. We view metrics as being used to evaluate the performance of a protocol whereas auxiliary metrics are used to evaluate the effectiveness of a simulation scenario. There is a danger in some of the auxiliary metrics because there are *similar* metrics that are more common to

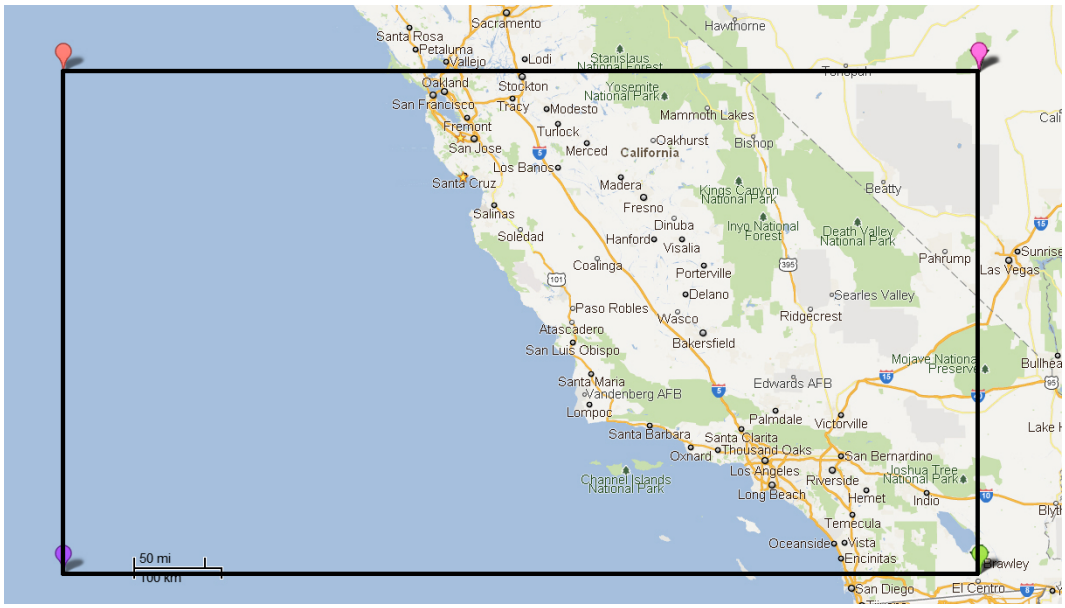

Figure 4: Maximum area of coordinates found in San Francisco Taxi VANET trace files.

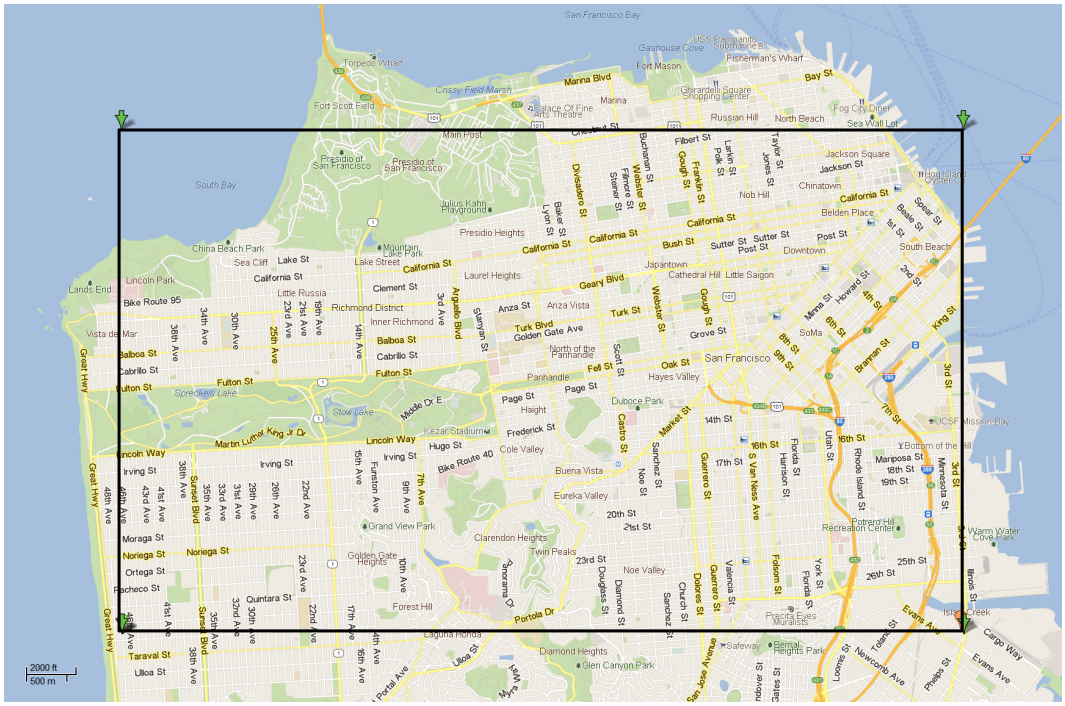

Figure 5: Area used in San Francisco Taxi VANET scenario.

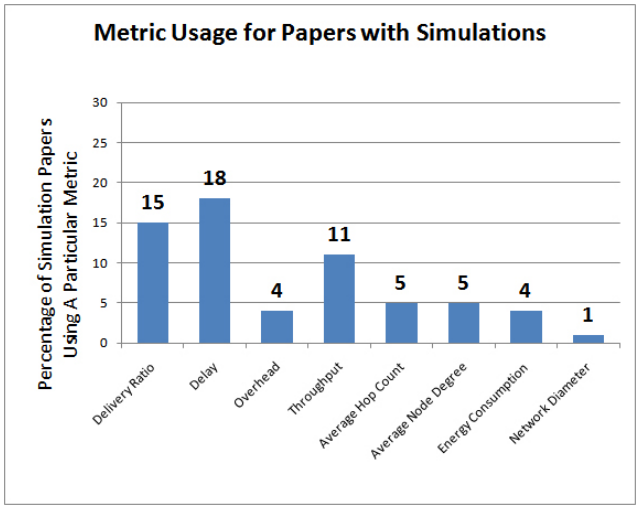

Figure 6: Percentage of papers found in MobiHoc 2006-2010 that used the indicated metrics.

use (such as average hop count), but they don't provide as valuable of information (compared to average shortest path hop count).

In our extended survey of MANET routing protocol papers, we found that there were several de facto metrics for evaluating the performance of a protocol [20]. These metrics are shown is Figure 6, and include Delivery Ratio (number of packets received divided by the number of packets sent), Average Delay (Average over all received packets' arrival time minus departure time), Throughput (Total number of bytes sent divided by the total amount of time data was being sent), Routing Overhead (The number of routing packets sent per data packet sent), and Energy Consumption, while the most common auxiliary metric was Average Node Degree (Average Number of Node Neighbors). Some metrics were combined for comparisons because there are several variations of the metrics that can alternatively be used (such as throughput can be alternatively measured as goodput - the total number of bytes received divided by the amount of time data was being sent).

Several auxiliary metrics have been proposed such as the Average Shortest Path Hop Count and Average Network Partitioning which measure the average shortest path number of hops to any destination from any source, and the average percentage of nodes that are reachable by any source to determine the connectivity of the network [24]. Since both of these auxiliary metrics measure information about *potential*

paths for data (not necessarily ones that are traveled by sent data), they must be measured at small intervals of time over the life of the simulation as opposed to measured based on the packets sent over the duration of the simulation.

We propose that there are five more important auxiliary metrics that should be accounted for in determining the effectiveness of a simulation scenario: Average Node Density, the Average Number of Concurrent Flows, the Number of Unroutable Packets, the Total Average Connectivity, and the Average Source-Destination Distance in Transmission Range Hops.

7.1.1 Average Node Density

Average Node Density is a coarse measure of how dense or sparse a network is. In order to calculate the Average Node Density of a network, we assume a grid distribution of the nodes across the environment, and we also approximate a node's transmission area as a square with sides of length $2 * T_r$ where T_r is the Transmission Range of the node. We first divide the environment area ($width * height$) into cells of the same size as the node's transmission area, and then we divide the number of nodes by the number of cells which gives us the number of nodes per cell. This auxiliary metric is very easy to calculate, and can give valuable insight into the scenario long before the scenario is simulated. It is important to note that since the density of the network will change over time in a mobile network, so in order to more accurately calculate the density of a dynamic network, measurements are taken periodically throughout the simulation to capture a sequence of instantaneous node densities.

Our implementation of the average node density auxiliary metric also produces a visual node density map. This map displays the environment discretized into cells of size $2 * T_r$ in varying intensities of color based on the percentage of nodes that occupied the cells. A higher intensity cell indicates that there were many nodes populating that cell for a significant period of time over the course of the simulation. An example of the density map is shown in Figure 7.

7.1.2 Average Number of Concurrent Flows

The Average Number of Concurrent Flows is a measure of traffic flows that overlap in time. This metric will allow researchers to design simulations with higher numbers of concurrent flows to not only ensure that the network experiences higher congestion which will in turn demonstrate the worst-case performance of a protocol, it will also expose synthetic scenarios. Nodes routing multiple flows simultaneously will experience a higher level of stress at all layers: contention and collisions at the PHY/MAC layers, queueing in the routing layer, and QoS in the application. Real-world situations tend to have at least a few nodes in which flows converge upon such as fixed data sinks in smart energy meter networks [18] or emergency response situations where local clusters communicate internally before forwarding data to upstream clusters, so the lack of simulating flows converging upon certain points in the network will lead to unrealistic results.

Our implementation of the average number of concurrent flows uses the FlowMonitor class which is distributed with ns-3. FlowMonitor records information about traffic flowing between particular source and destination nodes including the first and last packet sent/receive times, the total amount of delay, the total amount of jitter, total number of packets sent/received, number of lost packets, and reasons for packets being lost. To determine the average number of concurrent flows, we first determine the length of time of a flow based on the first and last packets sent for that flow. Next, we construct an array that is the length of the amount of time that traffic is flowing in the simulation, and increment the counter in each cells of the array for all cells that fall between the time that the first packet is sent in the flow, and the last packet received in the flow. We then take this array which shows us how many flows are active at any point in time, and reduce it to a single average value.

We also considered implementing the related, and more powerful auxiliary metric, Average Number of Overlapping Flows. This metric gives insight into the number of concurrent flows, as well as detailing how the flows interact with each other. It is entirely possible to have a high number of concurrent flows in a network where none of the flows overlap, so the network would not be as stressed as it would

first appear. Counting the number of overlapping flows, however, would provide information regarding how many flows were being routed by each intermediate node in the path. Nodes having to simultaneously route for many flows would experience a higher load, and would thus produce more meaningful results when trying to stress the network. The drawbacks in implementing this auxiliary metric lie in needing to expose the path a packet will take to determine which nodes are routing for that flow.

Since paths are determined by the routing protocol, and those paths can change due to mobility, network load, available energy on the node, etc., this metric would need to keep a mapping between a flow and the path(s) it takes at each instance of time that the network is probed. Since some paths are pre-determined, it might be necessary to look inside the routing tables as packets are being generated which reduces the ability to calculate this metric without requiring specific code for each routing protocol. A better approach would be to have nodes record packet sequence numbers when they are sent or received so that the path can be determined and mapped back to the flow that the packets belong to in post-processing. The drawback of this approach is that is requires a significant amount of memory and post-processing power, so we leave this metric to the future work as an improvement upon the Average Number of Concurrent Flows.

7.1.3 Number of Unroutable Packets

The Number of Unroutable Packets is a measure of the packets dropped due to the routing protocol which includes no route existing between the source and destination nodes, and the TTL being exceeded on a packet. While Delivery Ratio includes one aspect of this auxiliary metric since it counts the number of packets that *were* able to be delivered, the packets that were not able to be delivered can be due to effects of the PHY, MAC, or routing layers - they are all grouped together to count towards dropped packets. In evaluating a routing protocol, it is important to know how many packets were dropped due to the physical and MAC layers, however those dropped packets should not be attributed to the routing protocol.

In this benchmark, we again used the the FlowMonitor class which is distributed with ns-3 to count the number of unroutable packets. While it currently already exists in ns-3, we still advocate the use of this auxiliary metric and would like to see it implemented in the other network simulators as well. FlowMonitor categorizes dropped packets under several labels - No Route, TTL Expire, Bad Checksum, Queue Overflow, Interface Down, Route Error, Fragment Timeout - so to construct the Number of Unroutable Packets, we combine the No Route, TTL Expire, and Route Error error codes, where Route Errors are generated by AODV indicating that a link is no longer valid.

To complete focus on the packets dropped due to the routing protocol, we investigated the implementation of an Ideal MAC. This MAC layer would guarantee delivery of all packets sent so that only the routing protocol would be responsible for dropping packets. While this solution provides a clearer picture of the routing protocol, it does move away from the effort of producing realistic simulations.

7.1.4 Total Average Connectivity

In addition to the above proposed auxiliary metrics, we also implemented a measure of the Total Average Connectivity of the network which is the inverse implementation of the Average Network Partitioning proposed in [24]. This metric is calculated by first constructing a one-hop neighborhood matrix for each node based on their current positions and transmission ranges, and then running the Floyd-Warshall algorithm to determine multi-hop paths. These matrices are generated periodically over the course of the simulation, and then aggregated when the simulation terminates to calculate the total average connectivity based on how many how many instances of time node S had any path to node D. Currently this implementation reduces the number of hops gathered at each instance of time into a binary counter of how many instances of time there was any length of a path to the destination, but it could easily be adapted to calculate the average minimum hop count based on transmission ranges.

7.1.5 Average Number of Source-Destination Distances in Tx Range Hops

The Average Number of Source-Destination Distance in Transmission Range Hops is a measure of the physical distance between source and destination nodes, which in turn determines the minimum number of hops a routing protocol could utilize to send data between the nodes. This metric takes the difference between the two nodes and divides it by the node's transmission range, T_r, to determine the line-of-sight minimum number of hops.

8 Results

In order to quantify the value in this benchmark system, we ran the routing protocols that are built into ns-3 (AODV, OLSR, and DSDV) against our sample scenarios and collected the metrics that we described in the previous section. The results and analysis are in the sections that follow.

Two of the auxiliary metrics produced results that are very similar across all three protocols tested, because they measure scenario features as opposed to the routing protocol. The measure of connectivity as shown in Figure 13 shows that the connectedness of a scenario does not change when testing the different protocols. Variation comes into play with random mobility models such as Random Waypoint, which can be seen in the slight variation of connectivity in the Average scenario. The other scenarios use static placement or mobility traces, so node positions in those scenarios are identical for each protocol that is tested.

The other auxiliary metric that produced nearly identical results across all runs within a scenario is the Average Source-Destination Distance in Tx Range Hops. This auxiliary metric can be seen in Figure 16. The reason it is nearly identical within each scenario is because it relies on three things - which nodes are randomly selected as sources/sinks, their positions with respect to time, and the transmission range of the nodes. Since the transmission range of a node does not change within a scenario, and each PNRG seed used is tested against each routing protocol so that

the same source-destination pairs are used, it comes down to the node mobility. As discussed above, the only scenario in this benchmark example that uses a random mobility model is the Average scenario, so that is why there is a slight variation in the average distance between source and destination nodes. The other scenarios either do not move the nodes, or they move them in identical ways for each run due to the mobility trace, so the distances result in the same values within a scenario.

8.1 Average Scenario

As a comparison, we also took the most commonly used parameters that we found in conducting a survey of the current best practices, and used them to construct an "Average Scenario" in order to see how a completely synthetic scenario would perform when compared directly to these realistic environments. The parameters used in this Average Scenario are shown in Table 9.

This scenario has a maximum of 20 possible concurrent flows, which all three of the protocols are able to maintain as shown in Figure 12. The connectivity shown in Figure13 is also approximately the same for each of the protocols, although it is low at 12-13%. The low connectivity results in low delivery ratios in Figure{refdeliveryRatioResults, however we can see that OLSR outperforms AODV and DSDV by about 20% and 25% respectively. While at first glance we see that this is an odd result, looking at throughput in Figure 17, we can see that OLSR sends significantly fewer data packets than AODV or DSDV. We can also see in Figure 14 that for those packets that are sent, the delay is much less than AODV or DSDV which allows them to be sent with higher success.

With respect to the total average node density which can be seen in Figure 7, the average scenario shows a fairly well distributed occupancy of the cells over the duration of the simulation which we attribute to the Random Waypoint mobility model. While at first glance this seems desirable, it does not reflect realistic movement.

A positive aspect of this scenario can be seen in the average source-destination distance in Tx range hops. The average scenario produces an average of 3.56 hops between source and destination nodes which means that the routing protocol will

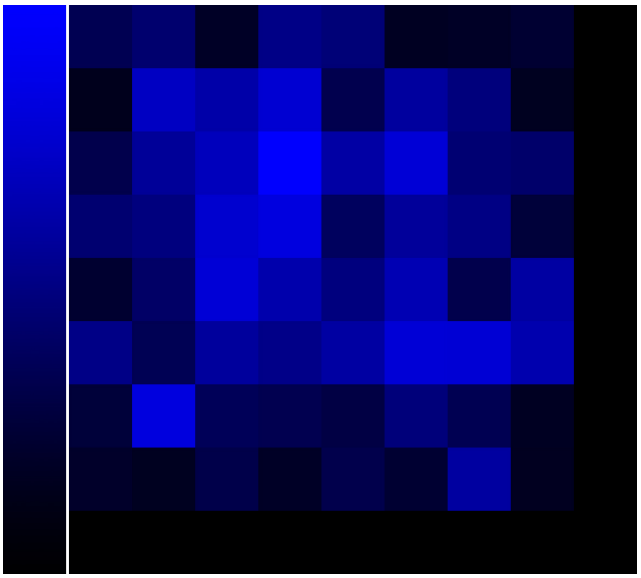

Figure 7: Average Scenario Node Density Map

need to produce effective multi-hop routes.

We can also see in Figure 19 that out of the packets dropped, OLSR produces a higher number of unroutable packets which we attribute to stale routes in a mobile environment.

Table 9: Parameters Selected for Synthetic Average Scenario

Average Scenario	
Value	Parameter
900s	Simulation Duration
1000m x 1000m	Environment Dimensions
100	Number of Nodes
20	Number of Sources
20	Number of Destinations
Random Sources, Random Fixed Sinks	Source-Dest Pair Selection
Random	Initial Node Distribution
Random Waypoint	Node Mobility Model
125m	Node Transmission Range
512 Bytes	Packet Size
4 Packets per Second	Per Node Packet Send Rate

Figure 8: Earthquake Monitoring Density Map

8.2 MASE Seismic Monitoring

We can see very clearly from the connectivity results shown in Figure 13 that this scenario constructs a network with 100% connectivity. The delivery ratios in Figure 15 indicate that all three protocols deliver close to 100% of the packets sent which is expected for such a well connected scenario. The average number of concurrent flows (Figure 12) is approximately 95 flows for each of the protocols which also falls in line with the characteristics of this scenario. We can see that AODV suffers slightly in packet delay (Figure 14) compared to the table-driven protocols which we attribute to it generating routes on demand as opposed to proactively maintaining the routes. Figure 17 shows that while OLSR and DSDV have similar data throughput rates, AODV performs worse which we attribute to the higher delays.

Given that this scenario is static and defined to place the nodes linearly, it is not surprising to see that the density map shown in Figure 8 depicts exactly that. It does, however, put into perspective the number of hops packets are traveling to get to the single destination for all of the traffic flows. Also, Figure 19 shows that none of the protocols produce a noticeable number of unroutable packets which indicates that drops in this scenario are not at the fault of the routing protocol.

8.3 Energy Grid

This scenario also being static, we expected somewhat similar results to the MASE Seismic Monitoring scenario. We notice similar trends due to their static node placement and high connectivity (although this scenario falls to 81% connectivity compared to the MASE scenario's 100% connectivity). An indication that there is a difference between the two scenarios can be seen by looking at the delivery ratio. We can see across all protocols that fewer packets are successfully delivered (Figure 15) which we attribute to the difference in connectivity. This scenario also produces similar numbers of concurrent flows in the three protocols (Figure 12) with

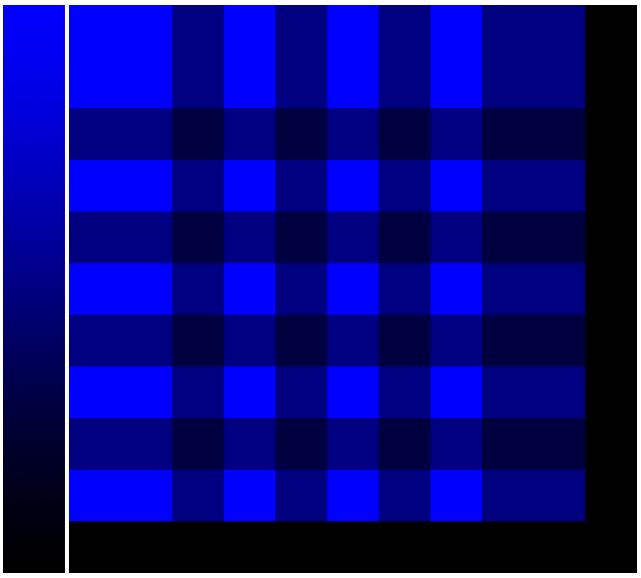

Figure 9: Smart Energy Grid Node Density Map

DSDV maintainng a slightly higher number of flows. We can see in Figure 14 that, similar to the MASE Seismic scenario, AODV has a higher delay than OLSR and DSDV which negatively impacts the throughput. Again, we attribute this to AODV generating routes on-demand compared to the proactive OLSR and DSDV protocol. Looking at throughput, we notice that this scenario has a very low throughput overall (Figure 17) but this due to the traffic model which sends few data packets infrequently.

The density map for this scenario shown in Figure 8 is quite symmetric which is expected given that this static scenario purposefully has nodes placed in a grid. Also, we can see in Figure 19 that there are a very small number of unroutable packets generated by AODV and OLSR, but the number is quite insignificant. This follows our expectation that that drops in this scenario are not due to the routing protocol, but rather due to queue overflows or lower layer interference.

8.4 Campus DTN

This scenario shows a surprising delivery ratio highly in favor of OLSR (Figure 15) which is unexpected due to the mobility trace used. We can see in the same figure that AODV and DSDV have approximately the same delivery ratio at around 17%.

While the throughput is very low for all of the protocols due to the traffic model used sending low numbers of very small data packets (Figure 17), we can see in the throughput that OLSR transmits far fewer data packets than both AODV and DSDV (Figure 17). We attribute this to being a byproduct of the number of concurrent flows that the protocols are able to maintain, because we can see in Figure 12 that OLSR is only able to achieve approximatley 50 flows while AODV and DSDV are able to maintain approximatley 340 each.

Each measured flow contains only the data that is sent end-to-end, so we attribute the discrepancy in the number of flows that OLSR achieves to be a byproduct of the routing overhead that is generated by the protocol. Since a flow is defined by the data packets that are sent–a flow only exists if it has had at least one successful end-to-end data transmission–we believe that the flows that don't exist in OLSR are due to their inability to send at least one successful data packet end-to-end. With the nodes in this scenario being mobile, route updates are more frequent for OLSR which attempts to proactively maintain a table of routes which puts a higher strain on the network with the routing overhead required. We can see in Figure 18 that OLSR loads the network heavily with overhead packets which looks to prevent the unsuccessful flows from sending any end-to-end data packets which in turn reduces the number of concurrent flows as well as the delivery ratio.

Lastly, Figure 14 shows that OLSR has a much smaller delay than AODV or DSDV. The few packets that are sent by OLSR are able to be successfully delivered because it is only maintaining a fraction of the number of flows that AODV and DSDV are achieving, and they are able to reach their destinations very quickly which reduces the likelihood of the route breaking during a transmission. The combination of these events are what lead to a deceivingly high delivery ratio.

The density map for this scenario (Figure 10) reveals that although this network is quite disconnected as it, on average, has a few clusters that are separated from each other, it does have several hotspots where the nodes congregate. This keeps the connectivity at a non-existent level.

We can see a measurable number of unroutable packets in this scenario in Figure 19

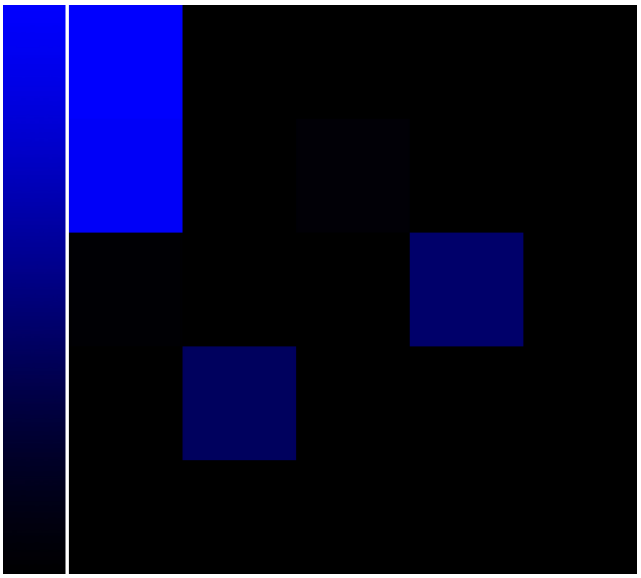

Figure 10: Campus DTN Node Density Map

which we attribute to the highly mobile environment which frequently breaks links. The figure shows that the table-driven protocols incur a slightly higher number of unroutable packets compared to the on-demand AODV.

8.5 San Francisco Taxi VANET

This scenario shows similar characteristics in the delivery ratio (Figure 15) to the Campus DTN scenario. It shows a surprisingly high delivery ratio for OLSR for having very low connectivity whereas AODV and DSDV show delivery ratio trends that are consistent the low connectivity. We caution the take-away of this single metric, however, because we see again in Figure 17 that OLSR sends significantly fewer packets than AODV and DSDV, it achieves a much lower delay for the packets that are sent (Figure 14), and these are because it only maintains a fraction of the number of concurrent flows that AODV and DSDV are supporting (Figure 12). This looks to be, again, due to the routing overhead shown in Figure 18 which indicates that OLSR is sending a significant amount more of overhead packets than AODV and DSDV. The influx of overhead packets puts a much higher strain on the network which limits the amount of data that the protocol is able to send, thus resulting in fewer concurrent flows.

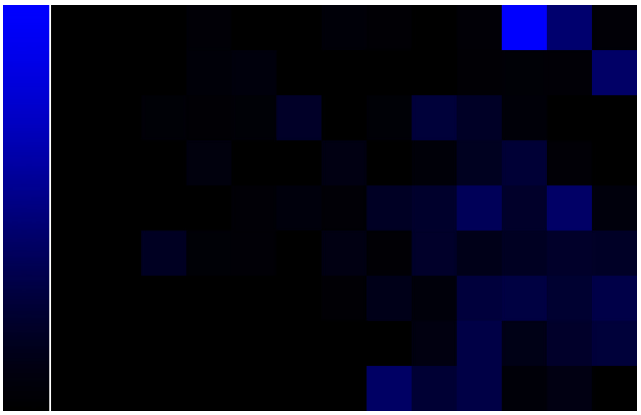

Figure 11: SF Taxi VANET Node Density Map

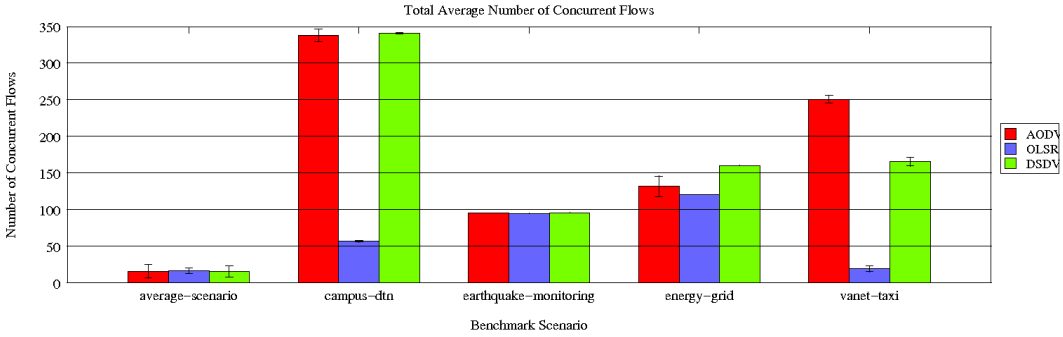

Figure 12: Average Number of Concurrent Flows results

The density map in Figure 11 indicates a fairly sparse network with a few hotspots of connectedness where higher number of nodes are more active.

This scenario also performs similarly to the Campus DTN scenario in terms of unroutable packets (Figure 19). Since this scenario is also highly mobile, we expect a number of the drops to be due to routing protocols generating stale or broken paths. We see again that the table-driven protocols show a slightly higher number of unroutable packets compared to AODV which generates its routes on-demand.

8.6 Validation of the Benchmark

In order to validate the correctness of the results produced by this benchmark, several comparisons were made. Ideally, we wanted to take an existing paper with well-known results and show that this benchmark produces the same trends. Unfortunately, it is extremely difficult to reproduce the results from someone else's scenario based on the information they provide in a paper or even with an extended

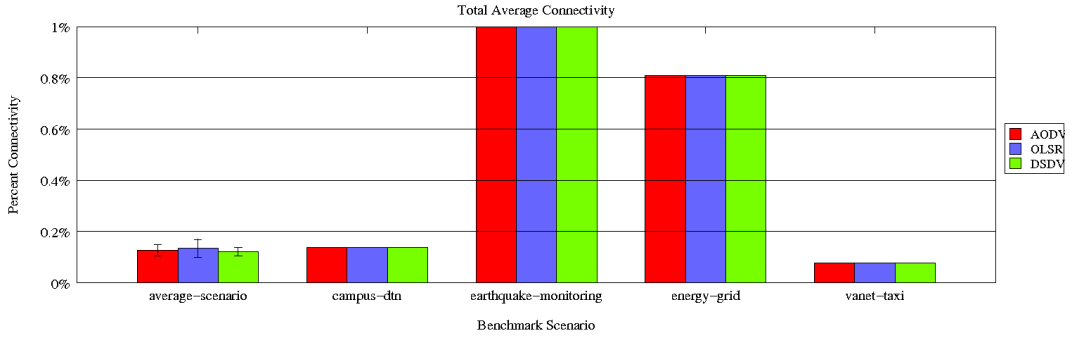

Figure 13: Total Average Connectivity results

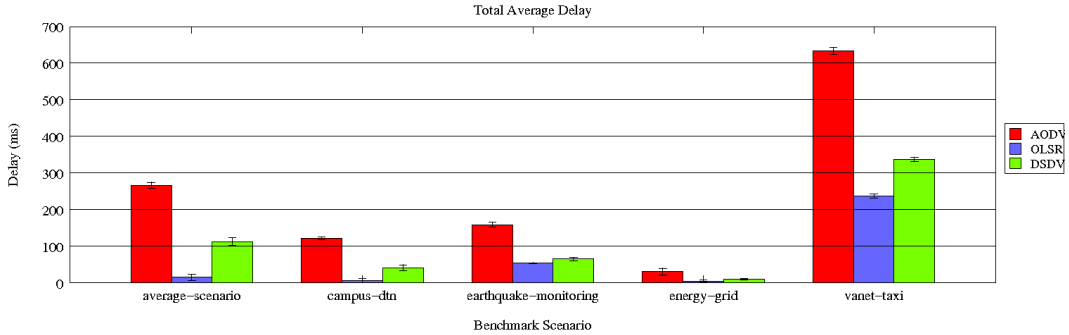

Figure 14: Total Average Delay results

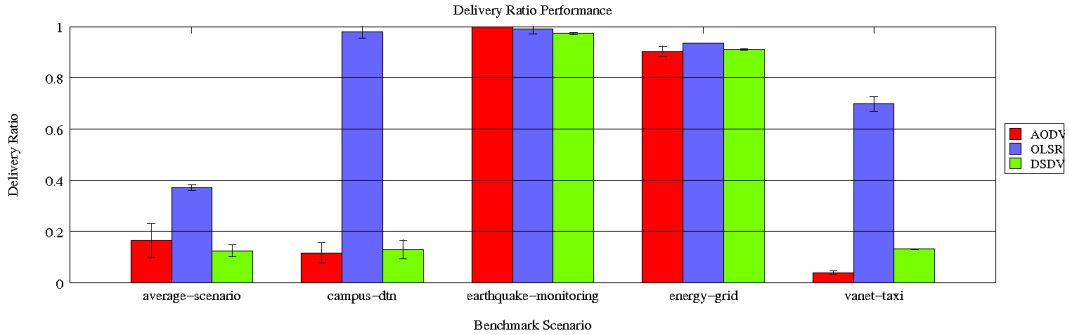

Figure 15: Delivery Ratio results

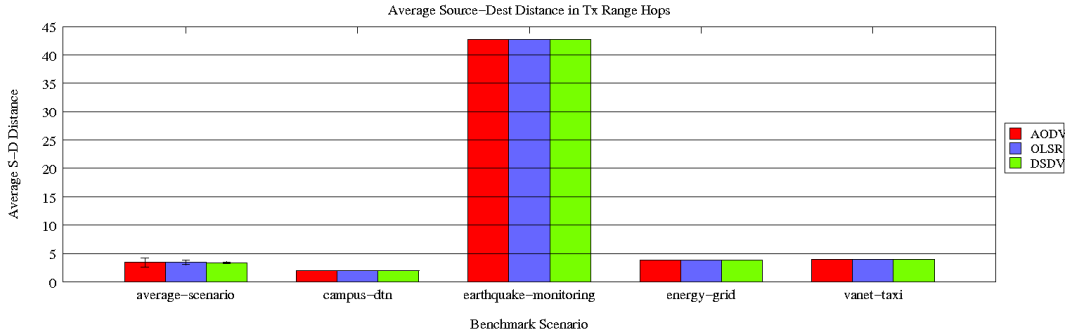

Figure 16: Average Source-Destination Distance results

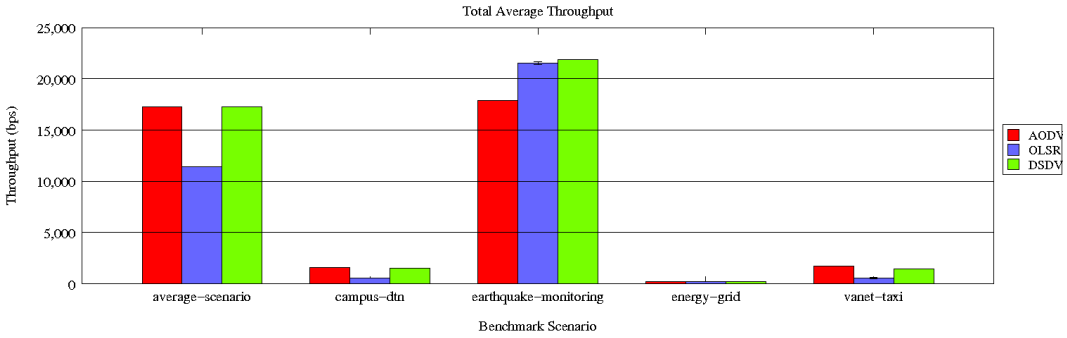

Figure 17: Average Throughput results

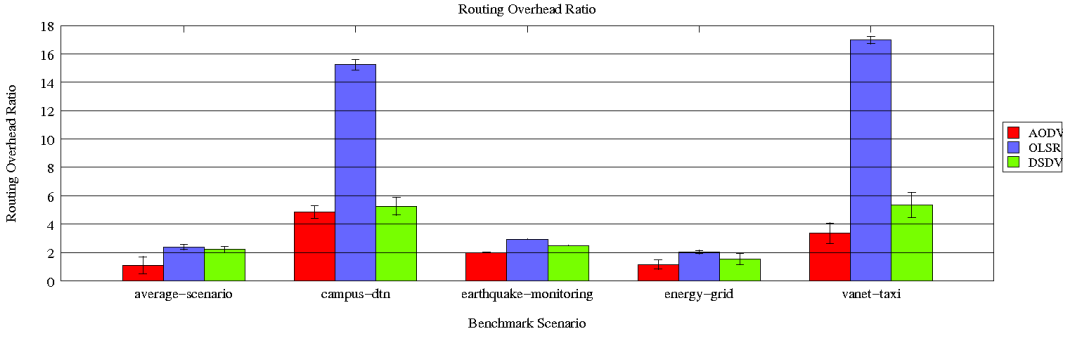

Figure 18: Routing Overhead Ratio results

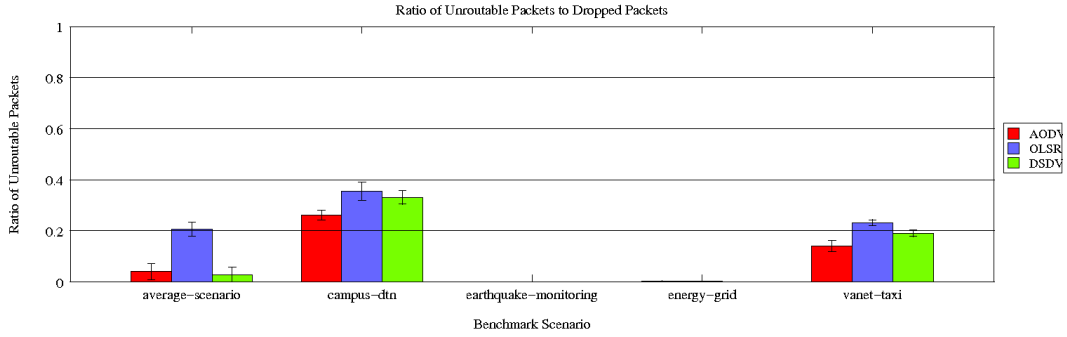

Figure 19: Unroutable Packets Ratio results

parameter selection on a website. There are a number of reasons for this, but a few common ones are that parameters used are forgotten or unknown, differences between simulators or even simulator versions produce different results, and metrics can be defined in different ways. One of the purposes of this benchmark is to provide a way for researchers to reproduce others' results since it is currently very difficult if not impossible.

Taking a different approach, pieces of the benchmark were tested in various ways. For example, as a preliminary test of the benchmark, we ran the `manet-routing-compare` example in ns-3 to produce results for an arbitrary simulation. We then implemented the same scenario in the benchmark and found the results to be very similar. A snippet of the results are shown in Table 10 which shows that the benchmark produces runs the simulations as intended by the simulator.

Table 10: Comparison of ns-3 example to benchmark with AODV

Metric	Benchmark	manet-routing-compare
Avg Throughput	2948.93bps	2938.92bps
Avg Goodput	2045.38bps	1985.97bps
Delivery Ratio	0.6169	0.5939
Avg Delay	22.45ms	28.33ms
Unroutable Packets Ratio	0.0887	0.0738
Total Dropped Packets	5689	5970
Avg Concurrent Flows	9.0194	9.5335

Next, the processing of the FlowMonitor file was checked against another example provided with ns-3 - the `flowmon-parse-results.py` script. We simulated `flowmon-parse-results.py` which is a separate FlowMonitor example scenario and ran the output file through the `flowmon-parse-results.py` script as well as through the metrics processing script in the benchmark, and the results were identical.

Lastly, a scenario was implemented that was introduced in the FlowMonitor paper [5]. This scenario consists of rows of nodes connected by links with alternating capacities of 100kbps and 50kbps. Each node sends traffic to the node two hops away so that all of the traffic for the flow must travel over both the 100kbps link as

well as the 50kbps link. This results in an expected output of 50% packet loss. The results are shown in Figure 11.

Table 11: Comparison of FlowMonitor theoretical test implemented in the benchmark

Metric	Benchmark	Flowmon Measured	Expected Value
Avg Throughput	105476.87bps	99646.06bps	99631.00bps
Avg Goodput	50869.62bps	49832.11bps	49815.50bps
Delivery Ratio	0.4830	0.4978	0.5000
Avg Delay	8.2326s	8.8005s	8.8020s

9 Conclusions

As intuitive as it seems, it is extremely important to document the settings of an environment for any scientific test, and this is no different for MANET routing protocol simulations. There have been several studies showing the differences between protocol implementations between simulators, so simply relying on default values in a simulator can produce wildly different results when comparing one simulator to another. While it has been known for many years that the problem of not being able to accurately reproduce or compare routing protocol simulations has existed. It has also been known that researchers typically use completely synthetic scenarios with no real-world basis when evaluating their protocols. We believe that we are the first to produce a viable solution aimed at solving both of these problems. We have examined the de facto metrics that are collected to evaluate routing protocols as well as proposed several new auxiliary metrics which will aid in the design of new scenarios. We have implemented a prototype version of this benchmark in ns-3, and we have released the source code to this project in hopes that the community will adopt this method for producing reliable, reproducible, and distributable evaluations of their protocols.

References

[1] T.R. Andel and A. Yasinsac. On the credibility of manet simulations. *Computer*, 39(7):48 –54, july 2006.

[2] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 51:1–51:10, ICST, Brussels, Belgium, Belgium, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[3] Derek Bruening. Clustered/stacked filled bar graph generator, 2012.

[4] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[5] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3). In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 1:1–1:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[6] United States Census. Providence, r.i. housing unit density, census 2000, 1993.

[7] Yang Chen, Vincent Borrel, Mostafa Ammar, and Ellen Zegura. A framework for characterizing the wireless and mobile network continuum. *SIGCOMM Comput. Commun. Rev.*, 41:5–13, 2011.

[8] Dartmouth College. A community resource for archiving wireless data at dartmouth, 2011.

[9] Standard Performance Evaluation Corporation. Spec - standard performance evaluation corporation.

[10] Paul Davis, Deborah Estrin, William Kaiser, Dustin McIntire, Igor Stubailo, and John Wallace. Sei 03 geonet: A platform for rapid distributed geophysical sensing, 2011.

[11] Pengfei Di, Yaser Houri, Kendy Kutzner, and Thomas Furhmann. towards comparable network simulations, 2008.

[12] Nathan Eagle. Mit media lab: Reality mining, 2009.

[13] Dr. Stephan Bohacek et al. Udelmodels for simulation of mobile wireless networks, 2012.

[14] European Telecommunications Standards Institute (ETSI). Digital cellular telecommunications system (phase 2+); technical realization of the short message service (sms) point-to-point (pp) (gsm 03.40). *GSM Technical Specification*, 1996.

[15] Michael Feeley, Norman Hutchinson, and Suprio Ray. Realistic mobility for mobile ad hoc network simulation. In Ioanis Nikolaidis, Michel Barbeau, and Evangelos Kranakis, editors, *Ad-Hoc, Mobile, and Wireless Networks*, volume 3158 of *Lecture Notes in Computer Science*, pages 630–630. Springer Berlin - Heidelberg, 2004.

[16] Leslie D. Fife. A mobile ad-hoc network data communication benchmark. In *In Proc. 18 th International Conference on Parallel and Distributed Computing Systems, Las Vegas*, 2005.

[17] Laboratory for Advanced System Software. Umass trace repository, 2009.

[18] Pacific Gas and Electric Company. Smartmeter system–how it works, 2010.

[19] Institute of Transportation Systems German Aerospace Center. Sumo simulation of urban mobility, 2011.

[20] Daniel Hiranandani, Katia Obraczka, and J.J. Garcia-Luna-Aceves. Manet protocol simulations considered harmful: The case for benchmarking. *Submitted to: IEEE Wireless Communications Magazine*, 2012.

[21] D. Kidston and T. Kunz. Towards network simulations credibility: Lessons from applying five key principles. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1 –6, nov. 2008.

[22] Jonghyun Kim, Vinay Sridhara, and Stephan Bohacek. Realistic mobility simulation of urban mesh networks. *Ad Hoc Netw.*, 7(2):411–430, March 2009.

[23] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9:50–61, October 2005.

[24] Stuart Kurkowski, Tracy Camp, and William Navidi. Two standards for rigorous manet routing protocol evaluation. In *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 256 –266, oct. 2006.

[25] Engage Consulting Limited. High-level smart meter data traffic analysis, 2010.

[26] Bilal Maqbool, Dr. M.A. Peer, and S.M.K. Quadri. Towards the benchmarking of routing protocols for adhoc wireless networks. *International Journal on Computer Science and Technology*, 2:28–35, 2011.

[27] A. Munjal, T. Camp, and W.C. Navidi. Constructing rigorous manet simulation scenarios with realistic mobility. In *Wireless Conference (EW), 2010 European*, pages 817 –824, april 2010.

[28] P. Murphy, E. Welsh, and P. Frantz. Using bluetooth for short-term ad-hoc connections between moving vehicles: A feasibility study. *IEEE Vehicular Technology Conference (VTC)*, pages 414–418, 2002.

[29] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37:95–98, March 2007.

[30] Vinayak Naik, Martin Lukac, and Deborah Estrin. Tutorial about seismic sensor network, 2007.

[31] California Institute of Technology Tectonics Observatory. Proyecto mase, 2011.

[32] M. Piórkowski, M. Raya, A. Lezama Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux. Trans: realistic joint traffic and network simulator for vanets. *SIGMOBILE Mob. Comput. Commun. Rev.*, 12(1):31–33, January 2008.

[33] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, February 2009.

[34] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seongjoon Kim, and Song Chong. CRAWDAD data set ncsu/mobilitymodels (v. 2009-07-23). Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels, July 2009.

[35] Robert Karl Schmidt, Thomas Kllmer, Tim Leinmller, Bert Bddeker, and Gnter Schfer. Degradation of transmission range in vanets caused by interference. *Praxis der Informationsverarbeitung und Kommunikation*, 32(4):224–234, 2009.

[36] UCLA Center For Embedded Sensing. Seismic applications (sei), 2011.

[37] Arnis Siksna. The effects of block size and form in north american and australian city centres. *Urban Morphology*, 1:19–33, 1997.

[38] I. Stojmenovic. Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions. *Communications Magazine, IEEE*, 46(12):102 –107, december 2008.

[39] Mineo Takai, Rajive Bagrodia, Ken Tang, and Mario Gerla. Efficient wireless network simulations with detailed propagation models. *Wireless Networks*, 7:297–305, 2001. 10.1023/A:1016682323833.

[40] Jamal Toutouh and Enrique Alba. Optimizing olsr in vanets with differential evolution: a comprehensive study. In *Proceedings of the first ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, DIVANet '11, pages 1–8, New York, NY, USA, 2011. ACM.

[41] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312 – 1321 vol.2, march-3 april 2003.