

Reality vs Emulation: Running Real Mobility Traces on a Mobile Wireless Testbed *

Youngil Kim
Computer Science Dept
University of Maryland
College Park, MD, USA
ykim75@cs.umd.edu

Keith Taylor
Electrical and Computer
Engineering
University of Maryland
College Park, MD, USA
taylor23@umd.edu

Carson Dunbar
Electrical and Computer
Engineering
University of Maryland
College Park, MD, USA
cdunbar@umd.edu

Brenton Walker
Laboratory for
Telecommunications Sciences
College Park, MD, USA
brenton@ltsnet.net

Padma Mundur
UMIACS
University of Maryland
College Park, MD, USA
pmundur@umiacs.umd.edu

ABSTRACT

Laboratory-based mobile wireless testbeds such as MeshTest and the CMU Wireless Emulator are powerful platforms that allow users to perform controlled, repeatable, mobile wireless experiments in the lab. Unfortunately such systems can only accommodate 10-15 nodes in an experiment. We have designed and built a scalable wireless testbed that uses software virtualization and live migration to facilitate experiments involving intermittently connected networks with many multiples of the number of physical nodes available on such a testbed.

In this paper, we share our experience with using real traces from the DieselNet/DOME project on our testbed. While this trace provide GPS coordinates for where the contact occurs, we needed to deduce the overall mobility from using GPS logs for individual buses. Replicating the bus mobility on our testbed we recreate contact events and compare those to DOME traces. We also survey other traces we can use on VMT and evaluate how challenging they would be to run on the system.

Categories and Subject Descriptors: C.2.0 Computer-Communication Networks: General

General Terms: Experimentation, Measurement

Keywords: Delay Tolerant Networks, Scalable Wireless Testbed, Mobility Traces

*This work was funded in part by the Laboratory for Telecommunications Sciences, US Department of Defense. The opinions expressed in this paper reflect those of the authors, and do not necessarily represent those of the Department of Defense or US Federal Government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HotPlanet'11, June 28, 2011, Bethesda, Maryland, USA.
Copyright 2011 ACM 978-1-4503-0742-0/11/06 ...\$10.00.

1. INTRODUCTION

The goal of any researcher running an experiment is to determine how a complex system will behave under real-world conditions. One way that researchers have tried to add realism to their mobile wireless models and simulations is through the use of real network, mobility, and encounter traces. Since human and animal mobility is difficult to model and synthesize, running experiments driven by real traces allows researchers to see how their protocols would function when handled by real users in a real environment.

The collection and use of such traces has flourished since about 2005, and has been facilitated by Crowdad trace repository [14]. There are a variety of traces that include mobile node encounter information, such as the Hagggle traces [12], and location-only GPS traces, such as the San Francisco CabSpotting trace [11]. Some traces focus on long-term network behavior, while others feature detailed measurements of a single instance of a car passing a WiFi access point.

The use of real traces in simulation has added high-level realism to Delay-Tolerant Networking (DTN) simulations, however simulators such as NS2 [10] or the Opportunistic Network Environment (ONE) [9] still lack realism because everything from the physical world, to the various components of the network stack, to the implementation itself are merely models. Wired network emulators such as Emulab allow researchers to work with real implementations and real operating systems [5], but for wireless experiments the disconnection and loss models will not capture the real behavior of wireless drivers, MAC layer, and wireless hardware [13].

The most realistic experiments will always be achieved by running real field tests. However such tests can be difficult to manage and monitor, expensive, and impossible to reproduce. Because of this, several laboratory-based wireless testbeds have been built which attempt to provide researchers with the convenience and control of a simulation, while running on real devices using real RF [3, 8, 4]. In particular MeshTest and The CMU Wireless Emulator give real wireless nodes access to an emulated RF environment. Unfortunately these wireless emulators are limited to 10-20 nodes. Virtual MeshTest(VMT) was designed to facilitate larger DTN experiments on such systems. The core ideas

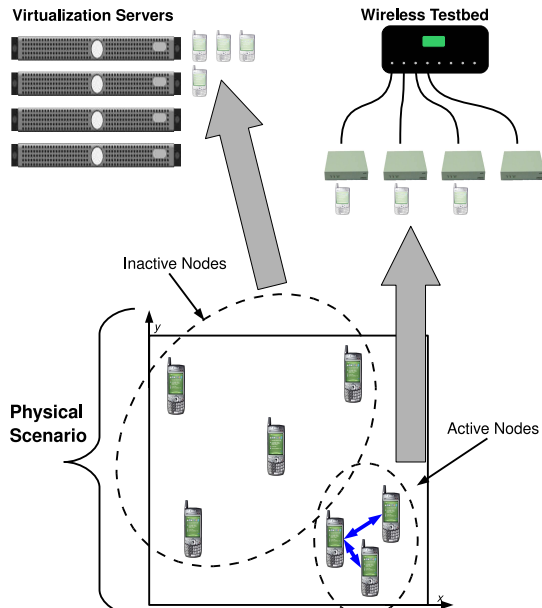


Figure 1: A schematic for the Testbed Design.

behind VMT are virtualization and live migration to share access to an emulated mobile RF environment, facilitating scalable experiments.

Our focus is performance evaluation of intermittently connected mobile wireless networks. This class of networks could span the spectrum from Mobile Ad-hoc Networks which are occasionally partitioned, to very sparse sensor networks with mobile data mules.

In situations where a network is partitioned we are able to conduct a large experiment as a collection of smaller scale mobile interactions. In our testbed the limiting resource is the number of inputs to the emulated RF environment. If a node in a mobile scenario is geographically isolated, it does not require access to the RF environment. Therefore, as a mobile network scenario evolves, only those nodes that are within communication range of other nodes are given access to the wireless medium. It is not practical to physically swap devices connected to the emulated RF environment, so we use software virtualization tools to move running node images on to and off of the wireless testbed nodes during an experiment. In this way, we can experiment with a network that has more nodes than could be supported on a non-virtualized system.

2. HOW VMT WORKS

An earlier version of VMT is described in [7]. Figure 1 shows a high level schematic of the premise behind the VMT testbed. The underlying physical scenario consists of arbitrary arrangements of nodes and a mobility pattern; there are nodes that are within communication range of one another and nodes that are geographically isolated. In our system each node involved in the physical scenario corresponds to a virtual node or *vnode*. The actual computers that the vnode images run on are physical nodes or *pnodes*. The wireless devices in the testbed and several powerful rack-mount

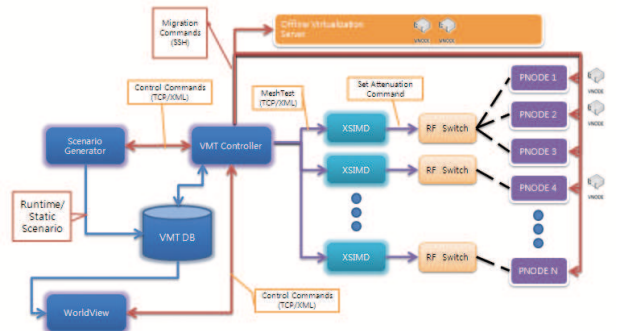


Figure 2: VMT Architecture

servers which have no wireless hardware are all pnodes. Each vnode has a running image on some pnode, and the vnode images are moved from one pnode to another as necessary to achieve the desired physical scenario.

Vnode images that are currently involved in wireless interactions are moved to the wireless pnodes in the testbed. Vnodes that are geographically isolated in the physical scenario are moved to the rack-mount servers. Furthermore, vnodes involved in separate isolated wireless interactions can be moved to pnodes connected to different RF switches. This means that the testbed can be expanded to support more vnodes by increasing the number of RF switches and pnodes, as long as no individual cluster of wireless nodes becomes too large. The evolution of the underlying physical scenario due to node mobility and the migration of vnode images are controlled by a new mobility/migration management system we developed.

2.1 VMT Architecture

The testbed architecture consists of three main components:

- An RF matrix switch and a collection of wireless devices which host vnodes that are within communication range of other nodes.
- Rack-mount servers that host isolated vnodes
- A mobility/migration management system that moves vnode images between these two physical entities and manipulates attenuation settings within the RF switch based on the underlying physical scenario.

The hardware setup consists of two JFW 50PA-338 8-port RF matrix switches, multiple Dell Studio Hybrid computers, an AOE server, and a virtualization server all connected through a gigabit Ethernet switch. The AOE server and virtualization servers are Penguin Relion 1600SC machines. Each Studio Hybrid is equipped with an Broadcom BCM4321 802.11 a/b/g card with Mini PCI Express as host interface and Windows driver using a frequency band of 2.4 GHz and 5 GHz. The Studio Hybrids are similar to Dell Studio-series laptops, but repackaged as a small form-factor desktop which allows them to fit into our shielded enclosures; they support gigabit ethernet and MiniPCI express Wi-Fi cards; they are available with a reasonably capable processor that supports Intel VT (Intel Core 2 Duo T8100).

The wireless testbed part consists of computers in shielded enclosures, two RF matrix switches and a switch control

server. The RF energy from each computer’s Wi-Fi card is cabled out of the enclosures and into the matrix switch of programmable attenuators. The enclosures prevent inadvertent cross-talk, and the RF switch allows us to arbitrarily control the attenuation between the wireless devices.

The program that controls the RF switch at the heart of the Virtual MeshTest system is a daemon called *xsimd*. The *xsimd* daemon was developed for the MeshTest testbed effort at LTS cited in [3] and is used with some modifications in the current VMT implementation. The daemon accepts physical scenarios from the mobility management system, computes the appropriate attenuator settings, and applies them to the RF matrix switch. Physical scenarios consist of an XML-encoded list of node locations. By repeatedly sending physical scenarios to *xsimd* at regular intervals the wireless testbed subjects devices to the simulated effects of mobility.

2.2 Node Migrations using Virtualization Software Xen

The core idea for the proposed testbed design depends on node virtualization and migration. The underlying design requirement is fast and seamless migration of vnodes between pnodes.

Xen is a paravirtualization technology for several architectures that provides a hardware abstraction similar but not identical to the raw hardware through its Virtual Machine Monitor (VMM) called a “hypervisor”. The modified guest OSes called domains run on top of the hypervisor.

Using wireless interfaces from within Xen virtual machines presented some challenges. Each virtual machine can have logical network interfaces. The MAC addresses of the logical interfaces can be either manually assigned or randomly generated. In bridging mode, the MAC address of the logical interface is visible to external networks. In routing mode, the IP address of the logical interface is visible to external networks. Xen also creates virtual network interface (*vif*) which is a pair of connected virtual Ethernet interfaces with one end in guest OS (also called *domainU*) and the other in host OS (also called *domain0*).

We use the bridging mode in the current implementation because common DTN neighbor discovery techniques are simpler if nodes are in the same MAC broadcast domain. A complication with Xen and wireless interface in bridge mode is that acks from destination VM when routed to the source VM can be rejected at the host interface on the source VM because of the wrong MAC address. Utilizing a Linux tool called *ebtables* to translate MAC addresses makes the bridge mode work with the wireless interface. Basically in this set-up, the source and destination addresses of the MP-DUs over the air are the MAC addresses assigned to the 802.11 physical interfaces. On the source side, the MAC address of the interfaces on the virtual machines are translated to the physical addresses before they reach the wireless physical interface. On the destination side, the MAC address of the destination is translated to the address of the virtual machine’s interface before it reaches the bridge.

2.3 Mobility Management Software

The software components in our testbed which control node movement and migration are also shown in figure 2. The system takes as input a set of parameters describing a physical mobility scenario. For example, we have tested

the system with mobility patterns for DataMULE and several variants of Random Direction (random), Mini-Beltway and three node drive-by (deterministic) models. The *Scenario Generator* produces an XML file encoding the desired mobile scenario, which is used by the *VMTController* to determine what node migrations should take place, and when, and encodes what physical scenarios should be sent to the *xsimd* daemons. This last step can get somewhat convoluted because the mapping between nodes in the original scenario and switch inputs changes throughout an experiment. An experiment is started by invoking the *VMTController*. The *VMTController* maintains real time during the experiment, sending physical scenarios to *xsimd* and issuing Xen commands to the pnodes.

The XML file generated from the scenario generator is used as input to execute the migration commands which move vnodes from either the Hybrids or the Penguin server. The controller also sends the current physical scenario to *xsimd* every second. When a vnode migrates from the penguin server to a hybrid, the corresponding attenuation settings between it and the nodes it is in range of should be updated as soon as the migration is complete. The issuing of *xsimd* commands is fairly straightforward, but the procedure for migrating vnodes can become somewhat complicated.

2.4 Migration Algorithms

Two primary migration algorithms are implemented in the current VMTController. The first algorithm called *Depth First Search-range (DFS-D)* uses a single-linkage clustering algorithm, similar to a depth-first search, to group nodes into clusters. When two nodes are within some distance D , called the *migration range*, of each other in the physical scenario, the VMTController *must* assign them to the same RF switch. If a node is more than distance D from any other node it is considered isolated, and is scheduled to be migrated to the offline virtualization server. If a given mobility scenario has communicating clusters that require more resources than are available (more switches, or more ports on a given switch), the VMTController will reject the migration of the corresponding node or nodes.

In addition, we have implemented a second algorithm named *Shortest Link First* intended to allocate as many as possible virtual nodes to physical nodes and the brief description is followed.

- Make a sorted list of inter-node distances
- Iterate through the pairs of nodes in the sorted list:
 1. If the 2 nodes are not already allocated to a switch, allocate them to the same switch. If there is no switch having more than 2 available ports, we cannot allocate them.
 2. If 1 node is already allocated, try to allocate the other node to the same switch, if there is no room in the switch, we cannot allocate it
- To minimize migration, try to allocate nodes to switch ports where it was allocated previously.

The SLF algorithm is designed to allocate as many virtual nodes as possible to physical nodes. Unlike DFS, the SLF algorithm does not have a hard boundary at which migrations must take place. Therefore the behavior does not change even when the wireless range of emulated wireless system is changed. In other words, the SLF algorithm max-

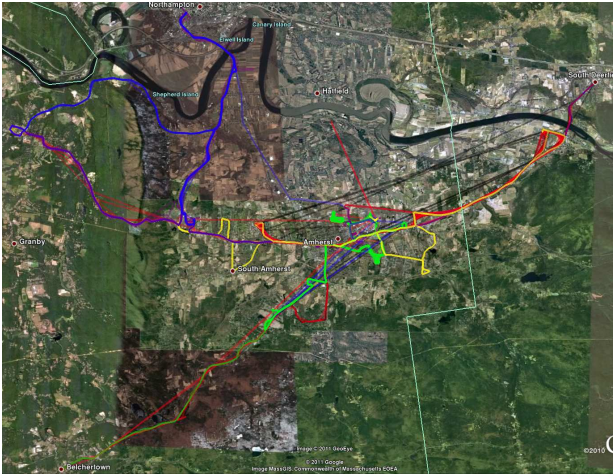


Figure 3: Bus Routes

imizes the utilization of physical nodes based on number of available RF-Switch ports.

3. EXPERIMENTS WITH DOME TRACES

In this section, we investigate using DOME traces on our testbed. DOME, which stands for Diverse Outdoor Mobile Environment, is a testbed meant for large-scale mobile experiments consisting of buses equipped with a Linux box, a WiFi radio, a GPS unit, and a long-range, low bandwidth 900 MHz Xtend radio [1]. The data collected from the DOME testbed over a period of four years is available through the CRAWDAD website [6]. In this paper we use the trace collected in the months of October to November 2007. In particular, we use two traces:

- *gps_logs* : These logs contain time stamped GPS locations for every bus and for each day during the collection period.
- *mobile-mobile*: This trace contains mobile-mobile contact events for each day during the collection period. Each line in the trace includes the time at which a contact occurred, the amount of data that was transferred, the duration of contact and the position where the contact occurred.

3.1 Mobility Scenario

The mobile-mobile trace gives the location of the contact but we need another means of computing the mobility routes for the buses. We implemented a *GPS parser* program for producing that mobility scenario with xml format based on the GPS information of buses in the *gps_logs*. This scenario is fed into the VMTController and desired mobility scenario is be executed. Since the VMTController requires the coordinate information for each node on a per second basis, the gaps in GPS logs are filled with linearly interpolated GPS coordinates. Figure 3 shows bus routes for the trace data including some interpolated sections.

For this paper, we use data for one hour from 2007-10-23 trace. This data has the most GPS information in the *gps_logs* directory for that day. Even though we tried to use the most informative time range, some buses do not have GPS information for more than 20 minutes. The missing

GPS information for a particular bus will result in missed contact when we later analyze the trace on the VMT testbed. The number of buses used in the experiment is 34.

3.2 Experimental Setup and Results

Using the prepared mobility scenario explained above, we test 3 algorithms: DFS with wireless range 500m, DFS with 100m range, and SLF algorithm. The total number of buses emulated in the experiment for this paper is 34. Running experiments with that many nodes is possible with our scalable testbed using virtualization and live migration.

When an experiment starts, VMT allocates virtual nodes to the network nodes in the mobility scenario. In our experiment, virtual nodes represent each bus.

All virtual nodes send out UDP broadcasts every second, and check for connectivity with other nodes using tcpdump to watch for the other nodes' broadcasts. If other virtual nodes detected, a timestamped record of is logged to a database Using a central database helps us keep time consistency among data from vnodes whose clocks have a tendency to drift by up to 30s per hour.

We ran the DOME experiments on the VMT testbed three times for each migration algorithm with the goal to replicate contact events from the mobile-mobile trace.

3.3 Contact Accuracy: DOME versus VMT

When VMT emulates given mobility scenario, it may miss contacts between two buses. For example, even though there is contact information in the mobile-mobile directory, there is might be no contact information in the experiment database. The *missed contact rate* can be calculated using following equation.

$$missed\ rate(\%) = \frac{(E^{DOME} \cup E^{VMT}) \setminus E^{VMT}}{|E^{DOME}|} \times 100$$

where E^{DOME} and E^{VMT} are the sets of seconds pairs of nodes were in contact for DOME and VMT, respectively. For example a 10s encounter between two nodes in VMT would correspond to 10 1-second elements in the set E^{VMT} . The elements of the sets are triples (*timestamp, node₁, node₂*).

Algorithm	Run1	Run2	Run3	Average
DFS 100	15.06	19.88	16.86	17.27
DFS 500	9.64	8.43	12.05	10.04
SLF	2.4	4.82	4.82	4.01

Table 1: Missed Contact Percentage in VMT from DOME Scenario

In Table 1, the percentage of contact misses, for the VMT results, are shown. The DFS-100 algorithm missed about 17% of the contact information for the mobile-mobile data, on average. However, the DFS-500 algorithm missed 10% of the contacts in the mobile-mobile data. The reason DFS-500 shows better contact accuracy than the DFS-100 algorithm is because the DFS-500 algorithm can allocate more virtual nodes to the physical nodes. There is a risk that it can lose an appreciable number of contacts if the size of group is bigger than the available number of ports on the RF-Switch. However, SLF algorithm shows better contact accuracy than the other two algorithms, missing only 4% of possible contacts in the mobile-mobile directory.

Possible reasons for missing contacts in the VMT experiment are:

- virtual nodes are not allocated to the physical nodes due to complications in migration algorithm
- the wireless interface of each virtual node requires time to reconfigure connections with other nodes after migration
- imperfections in hardware due to physical environment
- imperfect interpolation due to sparse GPS data
- mobility scenario does not match with the bus mobility

3.4 Contact Accuracy: Ideal versus VMT

In this section we compare contact traces generated on the VMT testbed with the ideal trace as generated using DOME GPS logs for the buses. The *ideal trace* is the trace generated from GPS data assuming the nodes are always in contact when they are within a given range, D . A range of 100m was used for the ideal scenario when comparing with the DFS100 algorithm while a range of 500m was used for comparison with the other two algorithms.

Algorithm	Run1	Run2	Run3	Average
DFS 100	13.22	14.53	13.73	13.22
DFS 500	28.56	24.80	25.14	26.17
SLF	35.92	36.29	36.38	36.20

Table 2: Missed Contact Percentage in VMT from Ideal Scenario

Algorithm	Run1	Run2	Run3	Average
DFS 100	6.38	6.26	5.55	6.06
DFS 500	11.87	12.8	12.73	12.46
SLF	27.46	29.42	28.34	28.40

Table 3: Extra Contact Percentage in VMT that is not in Ideal Scenario

The ideal trace uses a hard boundary of 100m or 500m to determine connection events. While in the VMT, we use real hardware devices where the connection range is variable to some limited extent. For instance, in a drive-by experiment for 1000m with a static node and a mobile node, we observed that a connection may actually start at 400m on one side of the static node and stay even longer for 700m on the other side. Even though our RF switch is calibrated for the range used in the experiment, the range may vary to a limited extent between pairs of nodes and will certainly be not exact as in the ideal trace. A late connection in the experiment will show up in the missing contact statistic and a slow disconnection will show up in the extra contact statistic as shown in Tables 2 and 3. SLF, with its more efficient assignment of nodes to the RF switch, has more connection events exhibiting that pattern and shows a higher percentage in both missed and extra contact than either of the range-based algorithms.

4. INVESTIGATION OF OTHER TRACES

Using the DOME/DieselNet traces in our initial experiments was an obvious choice because they include GPS information and a variety of contact information. There is a huge variety of other trace data available, however. In this section we discuss how different types of trace data could be used on the VMT system, and investigate the suitability of a couple other traces for driving VMT experiments. The

scenario	mean migration intensity
synthetic trace(DFS-500)	1.41 ± 1.61
synthetic trace(SLF)	2.13 ± 3.38
DNet range 100	3.20 ± 3.26
DNet range 500	4.38 ± 4.14
DNet SLF	2.16 ± 2.81
DNet range 500 with APs	20.88 ± 14.24
DNet SLF with APs	13.64 ± 9.30
CabSpotting 24 nodes	4.33 ± 4.99

Table 4: Migration intensity for various scenarios

key quantity we look at when checking suitability is the *migration intensity*. That is, the rate at which the system will need to migrate virtual machines in order to run the trace. The unit we use in our results is number of migrations per minute. The migration intensity will, of course, depend on the specific nodes selected for inclusion in the experiment and the migration algorithm used.

All of the migration data here was computed assuming a VMT testbed with 16 physical nodes connected to two 8-port RF switches. Results for the different scenarios described below are given in table 4.

4.1 Simple Synthetic Traces

We have run experiments on VMT using a simple synthetic trace based on the Bonn Motion [2] mobility generator. These early experiments were designed partly because of the ease with which we could use them on VMT. These scenarios featured 48 stationary nodes placed uniformly randomly on a grid of streets in a $4\text{km} \times 4\text{km}$ square area. For the first 2.5 hours of the experiment a source node does a random walk through the grid of streets. For the next 2.5 hours a destination node does a similar random walk.

With only one mobile node at a time to cause migrations, the migration intensity for these experiments was very low. In fact for the 50 node experiment only one 8-port RF switch was used, and it was never completely occupied. We computed the migration intensity for three independent 5 hour scenarios. The mean migration intensities are all between 1 and 2 migrations per minute with low variance.

4.2 DieselNet/DOME

The DieselNet/DOME experiment is well described in the previous sections. We computed the migration intensities for the 1 hour scenario described, with both 100m and 500m migration radii, and for the same scenario with several stationary access points (APs) along the routes. The access point locations were chosen from the APs reported in the dieselnet mobile-ap traces. We note that the migration intensity is drastically higher for the scenario with fixed APs. Since many of the chosen APs were in the center of the city, this leads to more, larger connected clusters of nodes, and hence more migrations. In the future we may choose which nodes to include partly based on the effect they have on migration intensity.

4.3 EPFL Cabspotting traces

The Cabspotting traces are GPS-only traces collected from over 500 taxi cabs in San Francisco in 2008 [11]. While there is no real contact data to compare to, the large number of nodes and long duration gives us an opportunity to experiment with the number of nodes included in an experiment.

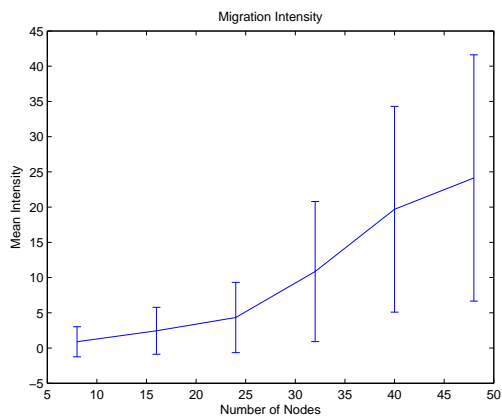


Figure 4: Migration intensity vs number of nodes used for the cabspotting traces.

For the results reported here we used random subsets of nodes of varying size. For each size, 8, 16, 24, 32, 40, and 48, we selected five independent random subsets. All of the scenarios were taken from the 8 hour period from 12:00 to 20:00 on May 17, 2008.

Migration intensity for these scenarios is shown in figure 4. As expected the migration intensity increases with the number of nodes in the experiment. Manual inspection of the migration logs reveals that these traces run with the DFS migration algorithm are prone to “migration flapping”. That is, the phenomenon of a large group of nodes migrating onto the testbed, and then migrating off a few seconds later. This will be less of an issue with SLF and other migration algorithms we are developing, but it does reflect an obstacle to running larger experiments with our current default algorithm.

4.4 Contact-only Traces

Contact-only traces such as huggle Bluetooth encounter traces [12] include contact information, but no GPS or other location information. Therefore fully running these traces on VMT will be impossible. Our approach to approximating the traces, however, is to group all the nodes in contact at each time slice into a cluster, put that entire group at a single arbitrary point in a physical scenario. By placing the groups at locations far enough apart we emulate a discrete on/off connectivity model that contact-only traces provide.

5. CONCLUSION AND FUTURE WORK

This paper provides a summary of the current state of the VMT system, and results from experiments based on real encounter traces. The results demonstrate the power of VMT to easily and reproducibly run experiments using a diverse variety of real trace-based scenarios. However they also highlight some of the challenges and limitations of the system. Though the testbed is now being actively used for real Delay-Tolerant Network experiments, we continue to develop the system. Issues highlighted in this paper that we are currently addressing include:

- **Migration algorithms:** In order to efficiently run experiments with larger numbers of nodes we need to ad-

dress inefficiencies with the current DFS migration algorithm. SLF is a first step in that direction.

- **Channel and terrain modeling:** Using our default channel model, our DieselNet encounter results were far more optimistic than the actual DieselNet traces turn out to be. Making this aspect of the experiments more realistic involves modeling of terrain and urban areas, and more advanced channel modelling.
- **Experiment management:** Setting up experiments on the current system requires considerable knowledge of the internals of the system. We have an effort underway to use Emulab to control VMT experiments. This will facilitate our eventual goal of opening the system to external researchers.

6. REFERENCES

- [1] BANERJEE, N., CORNER, M. D., TOWSLEY, D., AND LEVINE, B. N. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. In *ACM MobiCom* (2008).
- [2] Bonnmotion synthetic mobility traces. <http://bonnmotion.net.cs.uni-bonn.de/>.
- [3] CLANCY, T. C., AND WALKER, B. D. Messtest: Laboratory-based wireless testbed for large topologies. In *IEEE TridentCom 2007* (2007), pp. 1–6.
- [4] Carnegie Mellon University Wireless Emulator. <http://www.cs.cmu.edu/~emulator/>.
- [5] DEMMER, M., DU, B., AND BREWER, E. Tierstore: a distributed filesystem for challenged networks in developing regions. In *FAST’08* (2008), USENIX Association, pp. 1–14.
- [6] Dieselnet traces. <http://crawdad.cs.dartmouth.edu/meta.php?name=umass/diesel>.
- [7] HAHN, D., LEE, G., WALKER, B., BEECHER, M., AND MUNDUR, P. Using virtualization and live migration in a scalable mobile wireless testbed. *SIGMETRICS Perform. Eval. Rev.* 38 (January 2011), 21–25.
- [8] JUDD, G., AND STEENKISTE, P. Repeatable and realistic wireless experimentation through physical emulation. *SIGCOMM Comput. Commun. Rev.* 34, 1 (2004), 63–68.
- [9] KERÄNEN, A., OTT, J., AND KÄRKKÄINEN, T. The one simulator for dtn protocol evaluation. In *Simutools ’09* (2009), ICST, pp. 1–10.
- [10] The Network Simulator NS-2. <http://www.isi.edu/nsnam/ns/>.
- [11] PIORKOWSKI, M., SARAFIJANOVIC-DJUKIC, N., AND GROSSGLAUSER, M. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from <http://crawdad.cs.dartmouth.edu/epfl/mobility>.
- [12] SCOTT, J., GASS, R., CROWCROFT, J., HUI, P., DIOT, C., AND CHAINTREAU, A. CRAWDAD trace set. May 2009 Downloaded from <http://crawdad.cs.dartmouth.edu/cambridge/huggle/imote>.
- [13] SEASTROM, J. Characterizing transport layer behavior in the messtest wireless testbed. Master’s thesis, University of Maryland-College Park, 2008.
- [14] SONG, L., AND KOTZ, D. F. Evaluating opportunistic routing protocols with large realistic contact traces. In *CHANTS 2007* (2007).