

# Delay-Tolerant Network Experiments on the MeshTest Wireless Testbed

Matthew Seligman  
seligman@ltsnet.net

Brenton Walker  
brenton@ltsnet.net

T. Charles Clancy  
clancy@ltsnet.net

Laboratory for Telecommunications Sciences  
US Department of Defense

**Abstract**—Delay Tolerant Networks (DTNs) are a class of networks in which a contemporaneous end-to-end path from source to destination generally does not exist. Such networks often use on a store-carry-forward communication model which relies on the mobility of nodes to transfer data between geographically separated nodes. DTN researchers have relied heavily on simulation for evaluation, due to the difficulty and expense of running live experiments with real nodes and real DTN implementations.

MeshTest is a laboratory-based multi-hop wireless testbed that can subject real wireless nodes with real DTN implementations to reproducible mobile scenarios. It uses shielded enclosures and an RF matrix switch to dynamically control the attenuation experienced between pairs of nodes. The testbed is an ideal platform for DTN testing, offering convenient experimental control and data management.

We have installed the DTN2 Reference Implementation on the testbed nodes, and in this paper we report on a series of experiments based on the well known Data MULE model. Specifically, we investigate the effects of buffer limitations on the data MULEs and sensors node, velocity of the data MULEs, and bundle generation size and rate. We report experimental results including message delivery rate and latency for varying experimental parameters. We have found that an encounter between nodes does not guarantee a successful data transfer. In our experience, the quality and duration of the link, contention, and load on the nodes all influence the performance of the nodes.

## I. INTRODUCTION

Delay Tolerant Networks (DTNs) are a class of networks in which a contemporaneous end-to-end path from source to destination generally does not exist. Such networks often use a store-carry-forward communication model which relies on the mobility of nodes to transfer data between geographically separated nodes. Applications of DTNs include sensor networks, wildlife tracking networks, interplanetary networks, and user-to-user communication networks in harsh conditions where there is no communications infrastructure.

DTN researchers have relied heavily on simulation for evaluation, due to the difficulty and expense of running live experiments with real nodes and real DTN implementations. On one hand, laboratory-based experiments do not reflect the dynamic link conditions nodes experience in the field or the mobility on the scale necessary to test a DTN because it is difficult to achieve indoors. On the other hand, field tests require a large number of subjects and coordination

over large geographic areas. Even with careful choreography, such experiments are only marginally reproducible, which makes testing and debugging difficult. Data management and experimental control are also issues with live tests. Unless there is an out-of-band communication channel available, the transfer of experimental data and control information will have an effect on the experiment and may interfere with the results.

MeshTest is a laboratory-based wireless testbed that offers a hybrid between field testing and simulation [1]. The testbed features wireless nodes which are placed in shielded enclosures, and their RF wired into a matrix switch of programmable attenuators. By dynamically adjusting the attenuations between the nodes, we can subject the nodes to the effects of arbitrary physical arrangements and mobility scenarios. The testbed allows us to run a diverse variety of experiments with real hardware and real implementations, while being able to closely monitor and control the nodes.

MeshTest is an ideal platform for DTN experimentation. We have installed the DTN2 Reference Implementation on the testbed nodes and have run a variety of experiments similar to the well known Data MULE scenario [2]. This type of scenario features mobile data MULEs which provide store-and-carry forwarding from stationary data-generating **sensors** to an **access point (AP)** where all data is offloaded. We have generated a single 8-hour mobility scenario and have run several experiments with varying parameters under these identical conditions. We report a variety of statistics to help understand the effects of these experimental variations, including message completion rate, latency, and buffer size statistics.

In the following section we give a brief overview of some prior work involving DTN implementations, testbeds, and deployments. In section III we offer some more details about the MeshTest testbed. In section IV we describe our experimental methodology, and in section V we present our results and analysis.

## II. PRIOR WORK

In this section, we briefly describe a few types of DTN applications and highlight prior work done in testing DTN algorithms on various platforms, such as simulations, emulations, and testbeds.

### A. DTN Applications

**Remote location Internet service** Remote Internet service projects aim to provide asynchronous Internet access to users that do not have access to infrastructure-based Internet service. In some cases, infrastructure-based Internet service is cost-prohibitive in remote or third-world regions, such as in DakNet [3], TIER [4], KioskNet [5], [6], and the Wizzy Digital Courier [7]. In other cases, traditional Internet service is not available due to the limitations of the technology used, such as the Sami Network [8], where there is no satellite coverage.

**Wildlife sensor networks** These sensor networks are intended to monitor animals in their natural habitat, be it land or sea, and migrate the data to access points and eventually back to researchers. Unique challenges exist in these networks, including the method of deployment and maintenance. Some examples are ZebraNet [9] and SWIM, a sensor network for whales [10], [11].

**Urban Internet service** Urban Internet DTNs aim to provide Internet access through the use of public mobile resources. UMassDieselNet [12] uses public buses in the city of Amherst, MA to provide asynchronous Internet service to users on buses and along bus routes.

**Military communications** The DARPA sponsored Disruption Tolerant Networking program [13] is developing DTN-based military tactical radio for situations where peer-to-peer communications are required and a communications infrastructure is not available.

### B. DTN Research Approaches

We have found that the most common approaches to research and evaluation of new DTN algorithms or the performance of DTN algorithms in diverse networking applications are:

- Simulation and emulation
- Fixed infrastructure testbeds
- Choreographed mobile tests

DTN simulations provide a method for quickly prototyping new algorithms and testing the performance of algorithms under a variety of conditions at relatively low cost. Many facets of the network are abstracted, such as mobility, interference and fading, and real world issues encountered with real implementations, such as processing limitations on the nodes and implementation bugs. In addition, there are a growing number of different DTN simulators, and no particular simulator has become the defacto tool of choice. It is difficult to compare results from different papers due to the lack of simulator conformity (unlike the fairly standard use of ns-2 for TCP/IP research). The high level of abstraction in simulations leads to results that are of questionable realism, and the lack of conformity limits the usefulness of those results we have.

In [14] Emulab was used to test the DTN2 Reference Implementation under a variety of scenarios. In that paper the DTN2 implementation was compared to Sendmail and FTP in its ability to transfer data. Emulab provides a flexible platform to conduct repeatable experiments. The drawback of

the emulation approach is that all aspects of the MAC, PHY, and RF environment are still simulated.

Fixed infrastructure testbeds such as DieselNet and [15] provide realism by using an actual DTN implementation and real hardware. The DTN2 Reference Implementation, freely available through the DTN2RG [16], provides a common platform for these types of experiments. Its adoption by many researchers makes it possible to compare results. One of the drawbacks of this type testbed is that they tend to be limited to one type of test, and once systems are deployed, they are difficult to monitor, modify and update.

Lastly, choreographed mobile tests also provide real mobile and wireless network conditions and allow for the use of the DTN2 reference implementation and real hardware. A drawback of choreographed mobile field tests is that they cannot be re-run arbitrarily, and even when they are re-run, it is impossible to replicate the exact set of conditions each time. Some wireless testbeds have been considered for DTN experimentation such as the Roomba MADNeT testbed [17]. Although this provides a flexible platform to test DTN applications, the physical test space is limited in size and the external wireless environment cannot be fully controlled.

Given the three approaches described above, there is a clear need for a testbed that can provide realistic mobile wireless network conditions, repeatability, and experimental control, while using the DTN2 reference implementation on real wireless hardware. MeshTest meets all of these needs to some extent. We feel that it is an ideal platform for testing wireless DTNs.

## III. MESHTEST TESTBED

MeshTest consists of a rack of 12 computers in shielded enclosures, an RF matrix switch, and a server that provides experiment control, as depicted in Figure 1. The RF from each computer's WiFi card is cabled through the enclosures and into the matrix switch of programmable attenuators. The enclosures prevent inadvertent cross-talk, and the matrix switch allows us to arbitrarily control the attenuation between the devices. A practical evaluation of the testbed's RF environment and mobility management can be found in [18].

Figure 2 shows the logical construction of an  $n \times b$  switch. It has  $n$  inputs that connect through  $nb$  digital attenuators to  $b$  buses. Each bus has a direct, unattenuated, external connection. Note that the RF switch only simulates inter-node channel loss, and not propagation times.

While any device, from cellular telephones to software-defined radios, may be placed into the enclosures, the default configuration involves 802.11-based computers. Through a partnership with Rutgers, nodes and simulation management software from their ORBIT testbed [19] have been acquired.

The MeshTest testbed uses ORBIT's testbed management software to control the nodes. The additional piece necessary to make MeshTest work is software to map physical arrangements of nodes to the appropriate attenuator settings, and upload those settings to the switch in real-time during an experiment. During the summer of 2008, a coop student built a



Fig. 1. Assembly and connection of the shielded enclosures and RF switch.

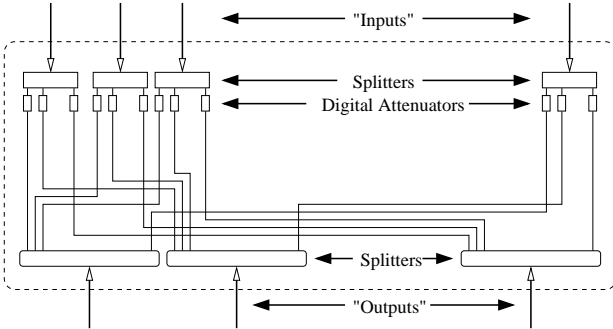


Fig. 2. RF matrix switch diagram, showing  $n$  upper I/O ports,  $b$  lower I/O ports, and  $nb$  Ethernet-controlled digital attenuators with ranges 0-127 dB

GUI-based program to perform these functions. The program allows a user to visually place nodes in a two-dimensional space and draw their desired mobility patterns. The program can also load, display, and process mobility scenarios from XML files generated by other programs.

Programming the digital attenuator settings for the matrix switch is not as obvious as it may seem. In particular, taking an arbitrary physical arrangement of devices one can easily compute a matrix,  $L$ , of inter-node attenuations, but adapting these values for the switch's matrix of attenuators can be challenging. In Figure 2, we can see that  $n$  nodes connect through  $nb$  attenuators to  $b$  buses. Let  $S$  be an  $n \times b$  matrix

representing the settings of the  $nb$  attenuators. In [20], it is shown that finding appropriate attenuator settings is equivalent to finding  $S$  such that

$$\Lambda .* S^T S = L \quad (1)$$

where  $.*$  is MATLAB notation for entry-wise multiplication of matrices, rather than standard matrix multiplication, and  $\Lambda$  is the insertion loss of the switch.

In [20] it is shown that one can use simulated annealing [21] to compute an approximate decomposition,  $S$ , for a variety of scenarios, including static topologies, mobile topologies, and situations that involve multi-path fading.

#### IV. EXPERIMENTS

We set up and ran experiments analogous to the Data MULE model described in [2]. Specifically, we adopt the three-tier architecture, including the assumptions made about the functionality of each tier. This includes stationary access points (AP) and sensors, and mobile data MULEs. The key assumptions are:

- Each sensor is resource-constrained
- Data MULEs do not exchange data
- Mobility of Data MULEs cannot be controlled

##### A. Software

All nodes are running version 2.5.0 of the DTN2 reference implementation, available from the DTNRG website[16]. The DTN2 reference implementation includes an application called *dtmsend* for sending bundles, and a receive bundle application called *dtmrecv*. When *dtmsend* is invoked, the generated bundle is stored by dtnd on the sensor until a data MULE comes within range. When a MULE comes into contact with the AP, as many bundles as possible are transferred to the AP and stored locally by dtnd. The *dtmrecv* application processes them asynchronously and may fall behind the actual data transfer. This functionality is required for the applications to be delay-tolerant.

##### B. Routing

We used the DTN2 *static routing* module with the following two forwarding rules:

- Sensors only forward data to MULEs
- MULEs only forward data to the AP

Each data MULE periodically sends out discovery announcements, and all nodes listen for these announcements. When a node detects a MULE, it opens an *opportunistic* link. All opportunistic links remain in the routing table until dtnd is restarted, though they are marked as "Unavailable" once the MULE is out of range. This discovery scheme was used for two reasons. First, as [2] states, sensor are assumed to be power constrained and should minimize their transmissions. Second, we have observed that if a sensor discovers the AP, the sensor would never pass any bundles to the MULEs. There is a certain probability that a sensor would pick up a discovery announcement sent by the AP, even if they were outside

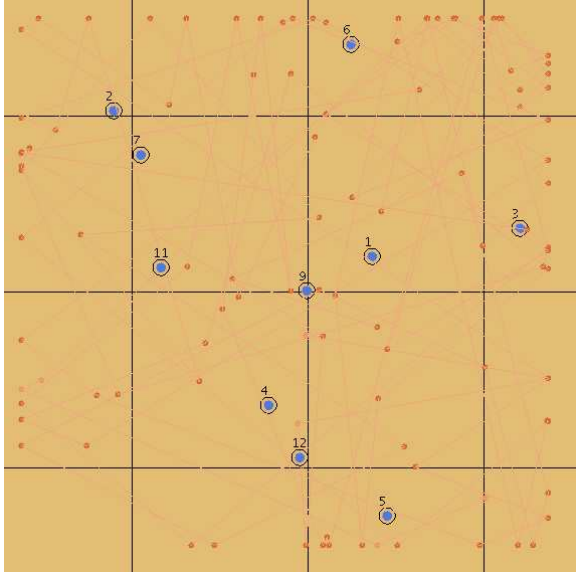


Fig. 3. Node arrangement and mobility pattern for the first two hours of the experiment. Nodes 1-7 are stationary sensors, node 9 is the AP, and nodes 11-12 are mobile data MULEs. The red dots along the MULE paths are points where the boundary was encountered, or a new random direction was chosen.

practical communication range. Therefore only the MULEs are allowed to send out discovery announcements.

### C. The Base Case Configuration

We ran one experiment that serves as the control, or **base case** experiment, and all other experiments vary in one aspect from the base case. In this section, we describe in detail the base case configuration.

1) *Data Generation*: Each sensor in the experiment generates bundles according to a Poisson process with intensity  $\lambda = 0.0333/sec$ , i.e. on average one bundle every 30 seconds. The size of the bundles is normally distributed with mean  $\mu = 10KB$  and variance  $\sigma^2 = 5KB$ . The data generator on each node is also run with the same random seed for each experiment, to ensure that the experiments are as repeatable as possible. The random seeds are based on the nodes' IP addresses, so no two nodes have the same seed.

2) *Node Positioning*: The positions of the seven sensors were chosen uniformly randomly in a  $3km \times 3km$  area. The AP is placed directly in the center of the area. The initial positions of the MULEs were also chosen uniformly randomly. A single arrangement of nodes was chosen and used for all experiments. The positions of all nodes during an experiment are shown in figure 3.

3) *Node Storage*: Each sensor's bundle storage is restricted to 500KB. In the base case no limits are placed on the storage space of the mules.

4) *Mobility*: The data MULEs follow a Random Direction mobility pattern [22]. On a bounded square this model gives a uniform long-term spatial distribution. That is, the model does not suffer from *edge effects*, which would tend to disadvantage nodes close the edges of the test area. In our implementation

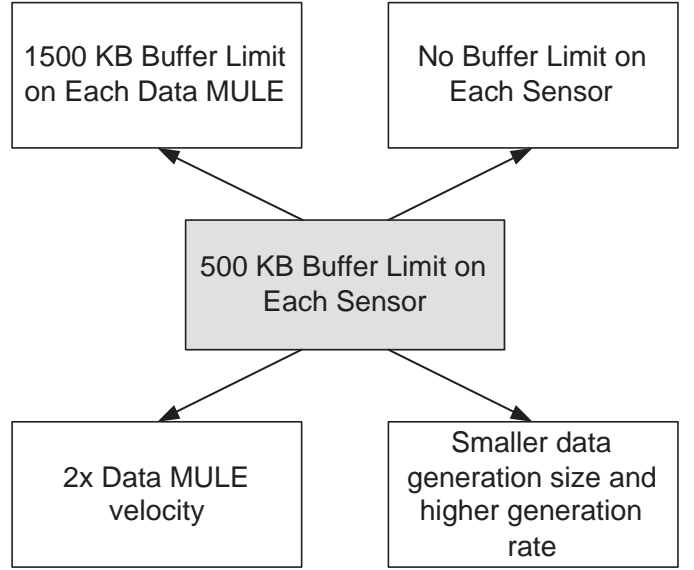


Fig. 4. The relationships between our different experiments. The middle gray box denotes the control, or *base case* experiment, and all other experiments vary a single aspect of the base case.

each mule chooses a direction  $\theta$  uniformly from  $[0, 2\pi)$ , a velocity  $v$  from  $[20km/h, 40km/h]$ , and a trip duration from  $[2min, 10min]$ . It then travels in the chosen direction at the chosen velocity for the chosen amount of time and repeats the process. The simulation takes place in a  $3km \times 3km$  bounded area, so when a node encounters the boundary it reflects, that is, the component of the velocity perpendicular to the boundary is reversed.

A single eight hour mobility scenario was generated and the identical scenario was used for each experiment. Figure 3 shows the paths followed by the mules during the first two hours of the experiment.

### D. Experiment Methodology

In this section we describe the rest of our experiments. Figure 4 visualizes the relationships between the different experiments. The following is a list of parameters that we experimented with:

- **Buffer Capacity**: The base case has a 500KB limit on each sensor and no limits on the MULEs or AP. One of our experiments removes all buffer capacity limits, and another experiment adds 1500KB buffer limits to the MULEs.
- **Data MULE Velocity**: In the base case data MULEs choose their velocity uniformly from  $[20km/hr, 40km/hr]$ . We ran one experiment which doubled the speed of the mules, so they fall in the interval  $[40km/hr, 80km/hr]$ .
- **Data Generation**: The data bundles are generated according to a Poisson process, on average once every 30 seconds, with normally distributed sizes. We ran one experiment in which 341 Byte bundles were generated on average every 1.0 seconds. This gives the same average

data generation rate as the base case, but with many small bundles.

### E. Metrics

The following metrics are used to evaluate and compare results among the experiments:

- Bundle Completion Rate (BCR) - Based on number of bundles

$$BCR = \frac{\# \text{ Unique Received Bundles}}{\# \text{ Generated Bundles}}$$

- Data Completion Rate (DCR) - Based on number of Bytes

$$DCR = \frac{\# \text{ Unique Received Bytes}}{\# \text{ Generated Bytes}}$$

- Mean Latency - Only data received by *dmrecv* is included in this metric
- Buffer Usage - Amount of storage used on a node at a given time
- Time-Weighted Network Storage [23] - Average amount of buffer usage, weighted by time. In a DTN, data may be stored for a long period of time, so the nominal buffer usage is not always of particular interest. In many cases, we are particularly interested in how long a node's buffer usage is high or at capacity, which is more accurately captured by the time-weighted network storage.
- Time Spent at Capacity (TSC) - Amount of time a node's buffer size is at its capacity

## V. RESULTS

Each experiment lasted 8 hours, which equates to 8 hours of wall time since the testbed runs in real-time. The results presented here are based only on the first 15000 sec, or approximately 4 hours. We made this cutoff because some runs experienced some variety of node failure about halfway through. This provides consistent data sets that can be analyzed and compared fairly.

### A. Base Case

As described in Section IV, we put a 500KB buffer limit on each sensor. On average, the data generation rate is 0.33KB/sec. Therefore, a buffer capacity limit of 500KB was selected to ensure that multiple sensors would operate at or near capacity during the experiment.

Figure 5 shows the buffer usage vs. time for Node 3 and Node 4. Node 3 spends the majority of the time from 8000 – 15000s at capacity. When a node is at capacity any newly generated data by is dropped due to insufficient storage. On the other hand, node 4 did reach capacity several times, but in all cases, a MULE was nearby to offload some or all of its data. Therefore, the TSC of node 3 is significantly higher than the TSC of node 4. In fact, table I shows that the node 3 TSC is approximately 4 times that of node 4. Both nodes experience instances of *partial offloading*, where a MULE was able to take some, but not all of the sensor's data.

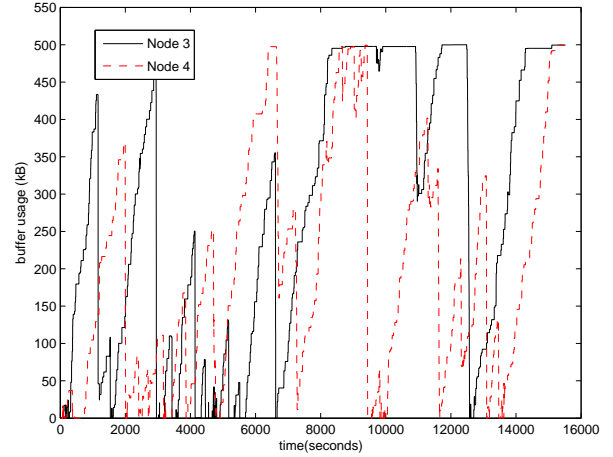


Fig. 5. Base case buffer usage for nodes 3 and 4. Both nodes reach capacity at some point, and both experience instances of partial offloading.

In table I we observe a correlation between the mean latency and the TSC. Similarly, nodes with a high BCR and DCR value experience a low mean latency. This is what we would intuitively expect, as sensors that have low TSC must be visited frequently by the MULEs, leading to more, and more frequent bundle deliveries, and the buffering of younger bundles. If the sensors used a different queueing strategy we might see the correlation between TSC and latency disappear.

Node	BCR	DCR	Latency(s)	TSC(s)
1	0.9084	0.9030	878.68	0
4	0.8310	0.8177	1083.64	1067
7	0.7596	0.7794	1279.90	1373
2	0.7204	0.7162	1176.83	1510
6	0.6820	0.6768	2158.50	3183
3	0.6424	0.6268	1785.23	4522

TABLE I

BCR, DCR, MEAN LATENCY, AND TSC FOR SENSOR NODES WITH 500 KB BUFFER LIMIT SORTED BY BCR

### B. No Buffer Limit on Sensors

This experiment is identical to the base case, except there was no limit on the sensors' buffer capacity. The buffer usage vs time for three sensors is plotted in figure 6. We observe the steady increase in buffer usage due to data generation, and the occasional sharp decreases corresponding to a data MULE coming within range. This figure shows that given our particular mobility scenario, node 7 has less frequent contacts with the data MULEs in comparison to the other nodes. This is especially true during the time period,  $t = 0 - 5000$ , during which time its buffer usage grows to over 1.5MB.

Histograms for the buffer usage of the same three nodes are shown in figure 7. Given a finite buffer limit,  $C$ , node 7 would drop bundles during all time periods when buffer usage is greater than  $C$ .

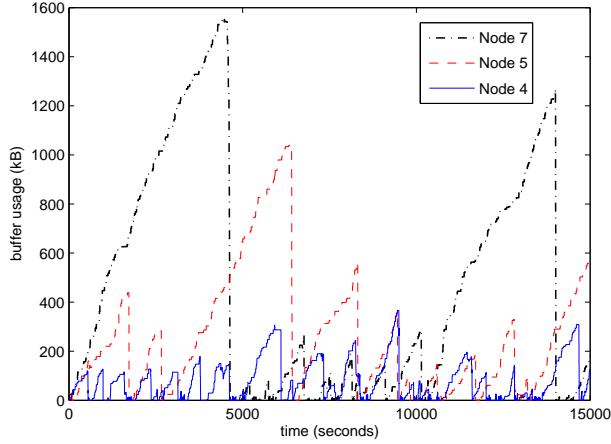


Fig. 6. No buffer limit on sensors. We observe that some nodes have more frequent contact with the data MULEs.

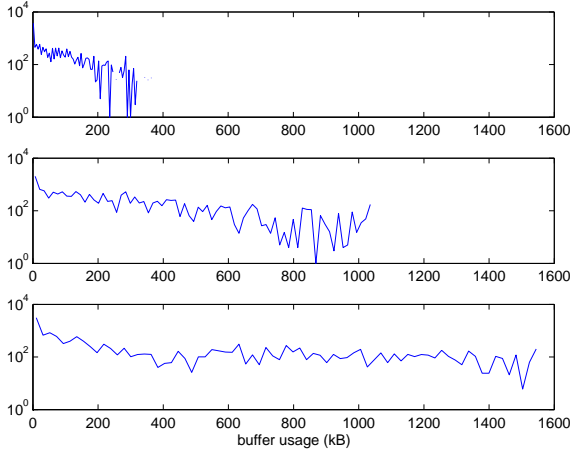


Fig. 7. Histogram of buffer usage for the No Buffer Limit experiment. From top to bottom, the plots are for nodes 4, 5, and 7.

At the end of the experiment the two data MULEs were left with significant quantities of undelivered data. Node 11 had 587 bundles totaling 5856KB and node 12 had 1366 bundles totaling 13469KB. For finite experiments BCR, DCR, and latency will be biased towards lower values, and the 19MB of data left stranded on the MULEs highlights this fact. In comparison to the base case results, we do not observe any correlation between these statistics in this experiment.

In this experiment, we computed a message duplication rate of 0.057. In the DTN bundle protocol specification [24], the ability to identify duplicate bundles is provided by custody transfer. In this experiment, custody transfer was not enabled. A comparison of this same experiment with custody transfer enabled could be an interesting future experiment.

Node	BCR	DCR	Latency(s)
4	0.3942	0.3886	1782.83
2	0.4053	0.4055	1818.13
1	0.3865	0.3815	1908.24
6	0.3620	0.3648	3329.73
3	0.4512	0.4579	3461.55
5	0.4189	0.4089	3934.16
7	0.4330	0.4294	4283.06

TABLE II  
BCR, DCR, AND MEAN LATENCY FOR SENSOR NODES WITH NO BUFFER LIMIT, SORTED BY LATENCY

### C. Buffer Limit on Sensors and Data MULEs

In this experiment we put a 1500KB limit on the the Data MULE buffers. This is three times the limit on the sensors. We chose this limit expecting that the MULEs would frequently be at capacity, and hoping that this fact would yield contrasting results. Results are shown in table III. When a MULE is at capacity it will not accept any bundles from the sensors. As expected, the BCR and DCR for each sensor node is significantly worse than in the *base case*.

Another effect of the data MULE buffer limitation is starvation. We observe that none of the data from node 2 is delivered to the AP during our experimental window. In this case, it seems that the data MULEs are always at capacity by the time they reach node 2. We believe that under such conditions, some mechanism, such as rate-limiting or a more pre-emptive scheme, such as WRED, needs to implemented to enforce fairness and allow all sensor nodes to offload data. Without this mechanism in place, this scenario always favors sensor nodes seen by the data MULE soon after it visits the access point. We leave the research and implementation of a traffic management scheme as future work.

In comparison to the *base case*, the mean latency values, in general, are higher. This occurs due to the fact that a data MULE is biased towards delivering data with a higher latency than if the MULE had no buffer limitation, or if it dropped older bundles in favor of accepting younger bundles. This also would be interesting future work.

Node	BCR	DCR	Latency(s)	TSC(s)
3	0.5105	0.5078	2365.36	2158
1	0.5040	0.4959	1714.79	1877
6	0.4460	0.4417	3162.65	714
4	0.4058	0.4047	1140.94	0
7	0.2782	0.2727	3718.47	1920
5	0.1709	0.1739	1568.26	1996
2	0.0	0.0	undefined	12864

TABLE III  
BCR, DCR, MEAN LATENCY, AND TSC FOR SENSOR NODES WITH 500 KB BUFFER LIMIT AND DATA MULES WITH 1500 KB BUFFER LIMIT SORTED BY BCR

#### D. Double MULE Velocity

In this section, the velocity of each data MULE is doubled. The mobility pattern is otherwise exactly the same, and the 8 hour experiment now takes 4 hours. It is speculated that by increasing the data MULEs' velocity, there will be more frequent offloading opportunities for the sensor nodes, and more frequent opportunities for the data MULEs to offload data to the access point. Results are shown in table IV. We observe that TSC values are lower for most sensor nodes compared to the base case, as hypothesized. Given more opportunities to offload data, each sensor should spend less of its time with its buffer usage at capacity.

Node	BCR	DCR	Latency(s)	TSC(s)
4	0.8295	0.8187	1013.73	1575
1	0.7809	0.7831	1195.40	0
6	0.7426	0.7371	1965.43	1388
3	0.7186	0.7134	1725.15	0
5	0.5146	0.4979	2228.07	4657
7	0.4475	0.4368	2953.86	6602

TABLE IV

BCR, DCR, MEAN LATENCY, AND TSC FOR SENSOR NODES WITH DATA MULES MOVING AT 2X VELOCITY SORTED BY BCR

The only node not consistent with this trend is node 7 for which the TSC increased for this experiment. This anomaly is partly illuminated by figure 8, which compares the buffer usage plot for this experiment to the base case. This figure shows that node 7 had fewer successful data transfers with the MULEs. In this particular case, the doubled MULE velocity seems to put node 7 at a disadvantage due to the fact that the increased velocity led to shorter, less reliable contacts. As we see, node 7 was simply unable to offload data to the MULE as often as it did during the base case experiment, even though we know that the MULEs were in range more frequently.

#### E. Decrease Bundle Size and Increase Bundle Generation Rate

In this experiment we modified the data generation process to maintain the same average data generation rate, but split the data into many small bundles. All bundles in this experiment are 341 bytes, and the generation process is still Poisson, but with an intensity of  $\lambda = 1.0/sec$ . Since we maintain the same average data generation rate, it is still reasonable to make comparisons to the base case. For all nodes, the results in table V exhibit lower BCR and higher mean latency compared to the base case. Part of the reason for this seems to be the overhead associated with processing so many bundles. We have observed that when a MULE is near the AP, it will often be running with load average 1.50 or above, and still be unable to move all of its bundles to the AP during the encounter. After about three hours both MULEs seem to be completely overwhelmed and their buffers soar to over 10MB, which corresponds to over 30,000 bundles. Of course our test parameters may seem extreme, but it does show that when dealing with many small bits of data, some sort of

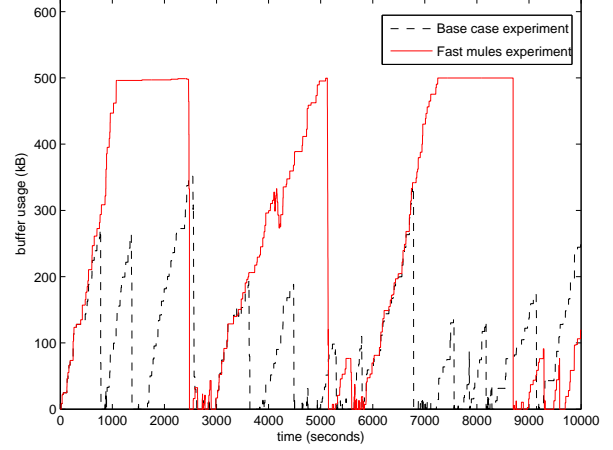


Fig. 8. Buffer usage at node 7 for the 2x velocity MULE experiment compared to base case. Despite the more frequent encounters, node 7 had fewer successful uploads to the MULE.

data consolidation may be necessary. We note that this sort of behavior is unlikely to be accurately modeled in a simulation.

Node	BCR	DCR	Latency(s)	TSC(s)
4	0.5307	0.5307	2472.05	2520
2	0.5199	0.5199	2448.75	1336
7	0.5136	0.5136	2653.64	2846
3	0.4985	0.4985	2588.42	2126
1	0.4530	0.4530	2401.97	1552
6	0.4063	0.4063	3110.20	2314
5	0.1742	0.1742	8573.97	6473

TABLE V

BCR, DCR, MEAN LATENCY, AND TSC FOR SENSOR NODES WITH BUNDLE SIZE OF 341 BYTES AND BUNDLE GENERATION RATE  $\lambda = 1/sec$  SORTED BY BCR

## VI. CONCLUSION

In this paper, we conducted original experiments using the MeshTest testbed and the DTN2 reference implementation. We focused on the Data MULE scenario in which stationary sensors generate data that is aggregated onto mobile data MULEs, and eventually offloaded to an AP. The *base case* experiment, which was derived from the original Data MULE work [2], provided a foundation of results to compare to in terms of completion rate, buffer usage, and time spent at capacity. From the *base case*, we varied several parameters including relaxing the buffer capacity limitation, adding a buffer limitation to the data MULE, and increasing the velocity of the data MULEs.

We found that eliminating the limit on sensor buffers leads to MULEs buffering a larger amount of data. Surprisingly, this actually decreased the completion rates. This occurs because all data generated at a sensor eventually ends up on a MULE, the MULEs end up carrying a significant amount

of data, and they cannot always offload all of the data to the AP during a contact (which was an assumption contained in the original data MULE work.) In addition, we found this experiment to have a non-negligible bundle duplication rate. This demonstrates the utility of a duplication detection mechanism at a higher layer than the convergence layer, such as *custody transfer*.

When limiting the buffer capacity at the data MULE, we found that specific sensor nodes were starved, as the MULEs were providing unfair service to the sensors. This result shows the need for some type of traffic management scheme to provide fair service to all sensor nodes. In addition, the buffer-limited data MULE was biased towards higher latency values in its results, since at capacity, the MULE would always hold older bundles rather than pick up younger ones.

We found that increasing the velocity of the data MULE did not necessarily lead to better performance. It clearly reduced the TSC, which is to be expected, but increasing the velocity of the data MULEs lead to shorter, less reliable contacts with sensors. In our experiment, one of the nodes had significantly fewer successful contacts with the MULE. This demonstrates the need to consider an efficient data exchange algorithm when contact times are small.

Finally, we demonstrated that MeshTest is an ideal evaluation platform for real DTN implementations. It allows us a great deal of realism, as it uses real implementations and real hardware, and also features repeatability and experimental control. We expect that experiments and testing carried out on MeshTest will lead to better, more reliable DTN implementations, and more reality-focused research.

## VII. FUTURE WORK

Since the purpose of these experiments was to create an initial DTN testing capability on MeshTest, there is a great deal of future work necessary to provide mature, reliable results. First, we need to better understand the functionality and behavior of the DTN2 reference implementation. During our experimentation, we often found it difficult to monitor the real status of the dtnd daemon, especially when tracking which bundles were residing in a particular node's storage. We also observed certain node behaviors that we suspect are bugs. The testbed will be an ideal platform for debugging implementations such as DTN2.

In addition to the experimental setup, there are several items tagged as future work throughout this paper. In particular, we noted that the non-trivial number of duplicate bundles demonstrate the need to study the effects of *custody transfer* mechanisms. Another interesting future task, particularly for the data MULE scenario, is to identify an approach allowing fair service to all sensor nodes in a DTN when data MULE buffer capacity limits exist. For sensor node applications, the data collected would be biased if they were collected from a biased subset of the sensor nodes. Therefore, research on traffic management schemes (i.e. how much data a sensor node is allowed to offload during a given contact with the MULE) is necessary.

Lastly, the data MULE scenario is representative of a subset of DTN applications, so for future experiments, we intend to study other DTN scenarios as well, such as message ferries [25] and limited or no knowledge DTNs using multi-copy routing algorithms. The work presented in this paper provides an initial demonstration of the utility of this platform for DTN research.

## REFERENCES

- [1] T. C. Clancy and B. D. Walker, "Messtest: Laboratory-based wireless testbed for large topologies," in *IEEE TridentCom 2007*, pp. 1–6, 2007.
- [2] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," in *IEEE SNPA*, 2003.
- [3] A. Pentland, R. Fletcher, and A. Hasson, "DakNet: Rethinking Connectivity in Developing Nations," in *IEEE Computer*, 2004.
- [4] "TIER Project." <http://tier.cs.berkeley.edu/>.
- [5] "KioskNet." <http://blizzard.cs.uwaterloo.ca/tetherless/index.php/KioskNet/>.
- [6] A. Seth, D. Kroeker, M. Zaharia, S. Guo, , and S. Keshav
- [7] "Wizzy Project." <http://www.wizzy.org.za/>.
- [8] "Sami Network." <http://www.snc.sapmi.net/>.
- [9] P. Juang, H. Oki, Y. Wang, M. Margaret, P. Li-Shiuan, and R. Daniel, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in *ASPLOS-X*, 2002.
- [10] T. Small and Z. J. Haas, "The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 233–244, ACM, 2003.
- [11] T. Small and Z. J. Haas, "Resource and performance tradeoffs in delay-tolerant wireless networks," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, (New York, NY, USA), pp. 260–267, ACM, 2005.
- [12] "UMassDieselNet." <http://prisms.cs.umass.edu/dome/>.
- [13] "DARPA Disruption Tolerant Networking Program." <http://www.darpa.mil/sto/strategic/dtn.html>.
- [14] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra, "Implementing Delay Tolerant Networking," Tech. Rep. IRB-TR-04-020, Intel Research Berkeley, 2004.
- [15] E. Oliver and H. Falaki, "Performance evaluation and analysis of delay tolerant networking," in *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*, (New York, NY, USA), pp. 1–6, ACM, 2007.
- [16] "DTN Research Group." <http://www.dtnrg.org/>.
- [17] "Roomba MADNeT : A Mobile Ad-Hoc Delay Tolerant Network Testbed." <http://dna-pubs.cs.columbia.edu/citation/paperfile/150/reich.MC2R.pdf>.
- [18] B. Walker and T. C. Clancy, "A quantitative evaluation of the messtest wireless testbed," in *TridentCom 2008*, March 2008.
- [19] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," *IEEE WCNC 2005*.
- [20] T. Clancy and B. Walker, "Messtest: Laboratory testbed for large wireless topologies," *IEEE TridentCOM 2007*.
- [21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.
- [22] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser, "An analysis of the optimum node density for ad hoc mobile networks," in *IEEE ICC 2001*, vol. 3, pp. 857–861, 2001.
- [23] M. Seligman, K. Fall, and P. Mundur, "Storage routing for dtn congestion control," *Wireless Communications and Mobile Computing*, vol. 7, pp. 1183–1196, 2007.
- [24] K. Scott and S. Burleigh, "Bundle Protocol Specification," *IETF RFC 5050*, 2007.
- [25] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *ACM MobiHoc*, 2004.