Anycast, Multicast and Beyond: the Role of Manycast in DTN Communication

Samuel C. Nelson
Department of Computer Science
University of Illinois at Urbana–Champaign
Urbana, IL, USA
snelso20@cs.uiuc.edu

Yih-Chun Hu
Department of Electrical and Computer Engineering
University of Illinois at Urbana–Champaign
Urbana, IL, USA
yihchun@crhc.uiuc.edu

Robin Kravets
Department of Computer Science
University of Illinois at Urbana–Champaign
Urbana, IL, USA
rhk@cs.uiuc.edu

ABSTRACT

With the recent proliferation of wireless communication devices, intermittent connectivity on the edge will quickly become a reality. These *disruption tolerant networks*, which are highly heterogeneous by nature, derive structure from node interaction and mobility. Despite this heterogeneity, the historical Internet-style design principle of point-to-point communication has dominated the DTN realm, severely hindering what could be a rich and diverse medium for new applications. This paper supports the concept of group-based communication in DTNs by exploring the paradigm of *manycast routing*, where the goal is to reach at least k members of a group of size m. This very general paradigm inherently includes other group-based routing concepts such as anycast and multicast.

Our manycast exploration takes a three pronged approach. First, the relative difficulty of manycast requests is quantified via analysis, which greatly deepens our theoretical knowledge of how challenging the general paradigm is in a DTN environment. Second, to understand how different replication-based classes of DTN routing protocols respond to and handle manycast requests, extensive simulations are performed in multiple types of network environments. These results show that any DTN manycast protocol must *dynamically react on a per-message basis* by dynamically changing their routing approach to achieve maximum results. Third, using the conclusions drawn from the analysis and simulation results, we present a DTN manycast meta-protocol that selects the appropriate routing technique based on the current request and network conditions.

1. INTRODUCTION

As mobile, wireless devices increase in popularity, intermittent connectivity becomes the norm. Robust communication to and from these devices requires protocols that are disruption tolerant by nature, with little reliance on static infrastructure. The natural communication patterns of these *disruption tolerant networks* (DTNs) differ from traditional Internet-based communication in that their structure is derived from node interaction and mobility. Furthermore, these networks can be highly heterogeneous and include smart phones, sensors, laptops, and even vehicles. Despite this heterogeneity, the historical Internet-style design principle of point-to-point communication (e.g., unicast) has dominated the DTN realm, severely hindering what could be a rich and diverse medium for new applications.

Unicast communication has so dominated the Internet that other communication paradigms, like anycast and multicast, are difficult to implement effectively. However, DTNs have given us a clean slate with a very different underlying structure. Given the intermittent connectivity expected in DTNs, communication is typically supported by some type of message replication, which naturally enables many non-unicast communication paradigms. Essentially, with multiple replicas of a message, multiple destinations can be reached, enabling group-based communication, like anycast, multicast and broadcast. However, it is very important not to be biased by the demands of Internet-based applications when considering group-based communication in DTNs. Traditional multicast, where all members of a group are guaranteed to receive the same messages, is not only difficult in DTNs, but may not even be the correct approach.

Instead of forcing one communication paradigm on all DTN applications, it is interesting to consider the whole space of communication as captured by the concept of manycast, where the end goal is to reach some k out of the m members of a group. In essence, manycast can be thought of as an umbrella paradigm that spans the space of single- and group-based communication simply by specifying the size of the group and/or the size of the recipient pool. For example, manycast can be configured to achieve anycast by setting the recipient pool to be one unspecified member of the group (i.e., $k = 1$). Similarly, multicast can be captured by setting k to m. However, these popular extremes only represent two ends of the broad spectrum, and, while useful, are not alone sufficient. Consider a sensor network that needs to collect a statistically significant sample of readings from a group of sensors. Anycasting would clearly be insufficient, and multicasting to the entire group would be extremely inefficient. In this case, the application should have the flexibility to specify the target number of nodes to reach, with the network dynamically responding to meet that specific request.

While the goal of flexible communication opens DTNs to a wide range of new applications, providing efficient manycast in a DTN environment has many challenges. Along with the standard challenges of all DTN communication, such as intermittent connectivity, heavy partitioning, high variance in resource constraints, and the lack of instantaneous end-to-end paths, manycast has the added difficulty of handling two new routing parameters, the target group size and the target number of group members to reach, which can vary with each application request. Additionally, any group-based communication must be integrated with a group management protocol [16], where knowledge of which nodes are in which groups is propagated throughout the network. Although group management is a key component, this paper focuses on the core routing protocols and we are currently investigating the integration of group management.

To achieve the full potential of manycast in DTNs, it would be beneficial to expose some of these difficulties to an application in a meaningful way that can guide the application in its decision as to how best to send its data. For example, if the network is relatively well connected, an application may try to reach more members of a group. To provide this information, it is necessary to understand how "difficult" it is to achieve manycast, in all of its dimensions, in a DTN. Furthermore, understanding the difficulty of a manycast request is a necessary first step towards making routing and replication decisions. Unfortunately, almost all of the existing work on DTN routing has exclusively considered unicast. Of the little work on group-based routing, it has been shown that anycast requires little replication for success [15], while multicast requires a lot of replication for success [1]. These preliminary results indicate that there is a wide variance in difficulty of manycast requests, depending on the application-specified target number of group members to reach.

The contribution of our research can be seen through our three-pronged approach towards the understanding and development of manycast in DTNs. First, we perform an extensive analysis to increase the theoretical understanding of the *fundamental difficulty* of achieving manycast in a DTN environment. By visualizing this space through extensive analysis, we are able to draw conclusions about the difficulty of anycast, manycast, and all points in-between. One of the most interesting observations is that *loose multicast*, where reaching almost all nodes in a group is considered a success, is a substantially easier paradigm than strict multicast, and should probably be considered a core component of group-based routing in DTNs. Second, we perform a simulation-based study that incorporates the naturally challenging DTN environment to understand how these factors, in addition to replication rate, affect the success of manycast communication. We show that quota-based protocols perform best when k is relatively small and flooding-based protocols perform best when k is relatively large. Furthermore, we show that the ideal transition point from quota- to flooding-based protocols is highly dependent on the environment, in particular the mobility. From this study, we show that for a manycast protocol to be effective in a DTN environment, it must *dynamically change its routing and replication approach on a per-message basis*. Finally, based on our analysis and simulation results, we develop a DTN manycast meta-protocol that selects the appropriate routing and replication technique based on the current request and network conditions.

The rest of this paper is presented out as follows. Section 2 explores applications for manycast as well as related work in the area. Section 3 presents a thorough analysis of the difficulty of manycast. To incorporate a more realistic environment, Section 4 explores, via simulation, varying manycast requests and how existing routing techniques and environmental properties affect their success. Drawing conclusions from the analysis and simulation studies, Section 5 presents a meta-protocol that dynamically incorporates multiple routing classes to best handle a user request. Finally, Section 6 discusses future work and concludes.

2. MANYCAST IN DTNS

Manycast is a very general paradigm that spans the space of single- and group-based communication, giving applications a high degree of flexibility in their choice of destinations. A manycast request can be defined using two parameters: m and k. The parameter m is the size of the destination group in the request. This parameter is likely to come from the network itself, or from a distributed group management component running on the network [16], as opposed to the actual application. The parameter k is the target number of nodes to reach in the destination group to satisfy the manycast request. Therefore, an (m, k) manycast request will be successful if a copy of the message is delivered to *at least k of the m nodes in the destination group*. The power of manycast is that it is general enough to include both anycast (i.e., $k = 1$) and multicast (i.e., $k = m$), as well as more traditional routing paradigms such as unicast (i.e., $k = m = 1$) and broadcast (i.e., $k = m = n$, where n is the number of nodes in the network).

The flexibility of manycast routing opens the door to a rich DTN application space. The inherent support of anycast and multicast alone brings some degree of flexibility to applications. For example, in an DTN used in a disaster zone, individuals in need of help can contact *any* emergency responder instead of a specific one. Furthermore, it is useful to transmit information such as building blueprints to *every* emergency response team leader.

However, the true power of manycast lies in the space between anycast and multicast. Building on the emergency response example, first responders who initially survey a scene may conclude that it is necessary to bring in a certain number of ambulances and/or firefighting vehicles. Manycast would allow these responders to request k of the m available vehicles, while anycast would be insufficient (only requesting one) and multicast would be inefficient (requesting all of them). Revisiting the sensor network example from the previous section, a manycast protocol that could deliver a COLLECT message to at least k of the m sensor nodes would allow for a statistically significant sample to be reached, without the inefficiency of reaching all sensors. Another interesting avenue for DTN applications is security. Contacting centralized trust authorities is very difficult due to the inherent lack of instantaneous end-to-end paths. Manycast would allow an application to contact a subset of distributed CAs, a primitive that is necessary for *threshold cryptography*, allowing a more robust form of trust in the network [22]. Even mobile social networking applications can benefit from the flexibility offered by manycast protocols. For instance, many smart-phones and handheld gaming devices have built-in WiFi and Bluetooth, which can be used for multiplayer gaming when friends are in close proximity. Manycast would enable gaming applications to find k of one's local group of m friends to join a game.

To the best of our knowledge, manycast has not been investigated in the context of a DTN environment. In mobile ad hoc networks, manycast has been shown to be quite useful [6]. However, in ad hoc networks, end-to-end paths are assumed to exist much of the time and manycast can be supported through the building of partial multicast trees in the network. Such solutions for ad hoc networks are not directly applicable to the DTN environment, since there is no expectation of such a high degree of connectivity or stability. Essentially, multicast or manycast trees will likely have a very short lifetime. Therefore, DTN manycast protocols must attempt to leverage replication, which does not require knowledge of the exact route, or even the exact set of nodes, that will receive the message.

Since manycast is a general form of group-based routing, existing work in anycast and multicast for DTNs is also relevant. As previous work shows, anycast is a highly useful and practical routing paradigm for DTNs [15]. While anycast has been considered in wired network scenarios [3, 17], it has only briefly been explored in DTN environments, where the exploration has been limited to single-copy routing and/or highly constrained mobility [10, 7]. Multicast, the other extreme of the manycast paradigm, has only very briefly been considered in DTN environments. In particular, a simulation-based study that explored how existing protocols handled multicast requests showed that a considerable amount of redundancy was necessary to reach all group members [1], a somewhat unsurprising result that is further confirmed by our work.

In this work, we thoroughly explore the general concept of manycast in a DTN environment using both analysis and simulation. Our results span from anycast to multicast and include all points in-between. In particular, our work gives insight into how routing and replication techniques must change from "easy" anycast requests to "hard" multicast requests, and how they can determine where the transition points are.

3. ANALYSIS OF MANYCAST DIFFICULTY

The first step in understanding manycast in a DTN environment is understanding how the *difficulty* of a request changes as the parameters k and m change. Since these parameters are highly dynamic, the variance in difficulty of manycast requests is very high. Determining the difficulty of a request is a necessary prerequisite to understanding how to best route and replicate the message. For instance, anycast requests are considered relatively easy, requiring little replication to satisfy [15]. At the other extreme, multicast requests are considered relatively hard, requiring a lot of replication to satisfy [1]. This section thoroughly analyzes the difficulty of all requests, precisely quantifying how that difficulty varies with respect to k and m.

3.1 Mathematical Analysis

The fundamental difficulty of a request can be defined in a probabilistic fashion. Given n nodes in the network, a group size m and a recipient pool size of k, $P(m, k)$ is the difficulty of satisfying a manycast request (m, k). To capture mobility and node contacts, assume a node meets other nodes uniformly at random, and can expect to meet c nodes per time unit. Furthermore, assume messages expire after t time units from creation. To enable our mathematical analysis, we assume routing is done via *direct delivery*, where only the source node ever replicates a message, and only does so to deliver the message to a member of the target group who does not forward it further. This simplistic, but parametrized, system model is assumed for mathematical analysis only. In Section 4, we use a more realistic environment for in depth evaluation.

Problem: Given the previously described system model, compute $P(m,k)$, which represents the *probability of a copy of a generated message successfully reaching at least k of the m destination group members.*

The problem of manycast success is analogous to the following bin-ball problem. Assume there are n balls labeled 1 through n representing the nodes. Since the sender does not count, $n-1$ is more precise, however this is irrelevant to the computation. Further, assume that balls are picked one at a time, with equal probability, and the label of the picked ball is recorded. Balls are replaced after each pick. An experimenter has a total of $c \cdot t$ picks, since this is the total number of non-unique nodes the source node can expect to meet before the message expires. There is one target group with m members. Assume, without loss of generality, that the destination group members are the first m balls and have the labels 1 to m. Therefore, after $c \cdot t$ balls have been picked, we want to determine the probability that the experimenter sees *at least k unique* balls with labels less than or equal to m.

As a quick example, assume $n = 3$, $ct = 2$, $k = 2$, and $m = 2$. All possible sets of picks include (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3). Since $k = 2$, of these sets, only $(1,2)$ and $(2,1)$ meet the requirements for success. Therefore, the difficulty of this request, under the described system parameters, is $\frac{2}{9}$. As another example, consider $n = 4$, $ct = 3$, $k = 2$, and $m = 3$. If we work through this case, we obtain a difficulty of $\frac{42}{64}$, demonstrating how fast the space explodes: the number of possible sets is n^m.

Solving the whole problem of determining the probability that after ct non-unique picks, one sees at least k labels less than or equal to m is quite complex. To make the problem more tractable, we divide it into two steps: (1) given ct, the chance of getting exactly u unique picks (referred to as $f(ct,u)$), multiplied by (2) given u unique picks, the chance of seeing at least k values less than or equal to m (referred to as $g(u,k,m)$). These two steps are iterated over all reasonable values of u, which range from from k, since anything below k unique picks cannot result in success, to the minimum of n and ct, since the number of unique picks cannot exceed either the total number of balls or the total number of picks. Therefore, P is defined in terms of f and g as follows:

$$P(m,k) = \sum_{u=k}^{min(ct,n)} (f(ct,u) \cdot g(u,k,m)).$$

Recall that f captures the chance of getting exactly u unique values given ct picks. There are two ways this can occur: (1) the first $ct-1$ picks contain the u unique values needed, and so the last pick must be a duplicate, or (2) the first $ct-1$ picks contain $u-1$ unique values, and so the last pick must be unique. Note that the chance of the last pick being a duplicate if there are already u unique values is $\frac{u}{n}$. Similarly, the chance of the last pick being unique if there are already $u-1$ unique values is $1 - \frac{u-1}{n}$. Additionally, the chance of the last pick being unique if there are already $u-1$ unique values is $1 - \frac{u-1}{n}$. We can therefore write f using the recurrence relation

$$f(ct,u) = f(ct-1,u) \cdot \frac{u}{n} + f(ct-1,u-1) \cdot (1 - \frac{u-1}{n}).$$

The initial conditions of the recurrence are as follows. If there are any picks, there must be at least one unique pick, hence $f(ct,0) = 0$. If there is one pick, there must be exactly one unique value, hence $f(1,1) = 1$, and $f(1,u) = 0$ if $u \neq 1$.

Next, recall that g captures the chance of seeing at least k values less than or equal to m, given u unique picks. Seeing *at least* k values means seeing exactly k values or seeing exactly $k+1$ values or seeing exactly $k+2$ values, etc, up to seeing exactly m unique values less than or equal to m. We therefore introduce another variable, l, that ranges from k to m, and focus on computing the probability of seeing *exactly l* values less than or equal to m. This turns out to be a relatively simple counting problem. We first count the number of ways to see the l values less than or equal to m, and then count the number of ways to have the rest of the values greater than m. This is then divided by the total number of possible label combinations. Putting this all together, we define g as follows:

$$g(u,k,m) = \sum_{l=k}^{m} \frac{\binom{m}{l}\binom{n-m}{u-l}}{\binom{n}{u}}.$$

This completes the definition of $P(m,k)$, representing the difficulty of a manycast request to reach at least k nodes out of a group of size m. To help visualize the function, we implemented it in MATLAB, as described in the next subsection.

3.2 MATLAB Computation

To explore how manycast difficulty changes with varying request parameters, we implemented P in MATLAB and visualized the results over a wide range of system and user parameters. Memoization was used to both speed up and cut down on the memory consumption of the recursively defined f function. We are currently looking into finding a closed form of f.

The goal of this analysis is to understand how difficult it is for the target number of group members, k, to receive a message for a group size m. It is also important to understand how this difficulty changes as one or both of these parameters change. To capture how P varies with varying values of m and k, the results are presented as 3D graphs. The two control variables are m,

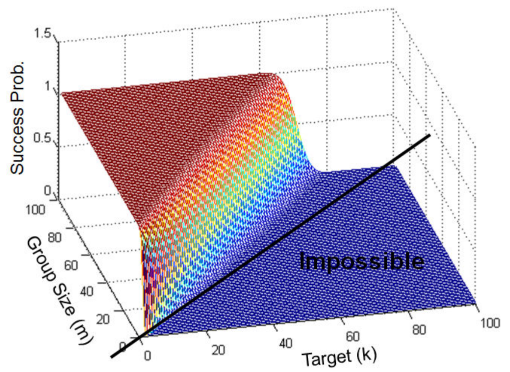


Figure 1: $P(m,k)$ for 100 Node Network

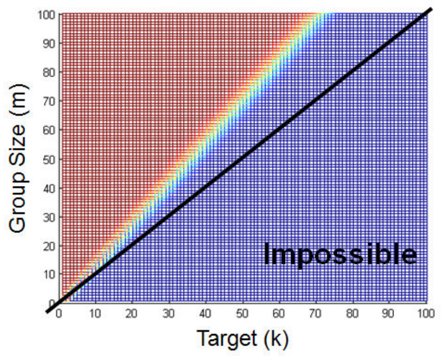


Figure 2: Top-Down View (100 Nodes; 2 Hour Expiration)

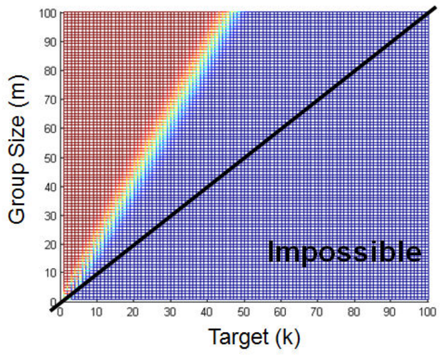


Figure 3: Top-Down View (100 Nodes; 1 Hour Expiration)

which ranges from 1 (e.g., unicast) to the total number of nodes in the network (e.g., broadcast), and k, which ranges from 1 (e.g., anycast) to m (e.g., multicast). The z axis represents the probability of success, or $P(m,k)$. All graphs incorporate color to improve the quality of the presentation, however, we will explain how to read the gray-scale versions.

As a first step, we look at a moderately sized network, $n = 100$, with reasonable encounter rates, $c = 1$ *per minute*, and reasonable message expiration times, $t = 2$ *hours*. Starting at $k = 1$ and increasing to around $k = \frac{2}{3}m$, there are a relatively large set of values where the success probability is close to 1. Essentially, these requests should be relatively easy to satisfy (see a 3D representation of the success probability in Figure 1). There is then a somewhat narrow transition point where the success rate falls to values close to 0. This transition point is interesting, since this is where routing techniques may have to change to provide the more challenging levels of manycast. When k is close to m, there is another relatively large portion with values close to 0, indicating that these requests are relatively difficult to satisfy. Finally, there is a large zone labeled "Impossible", where $k < m$. These requests are impossible to satisfy, since one cannot deliver a message to $k > m$ group members if there are only m members in the group.

One of the most interesting features of these 3D graphs is the transition from easy to difficult. To better compare where these transitions fall, we also present 2D top-down graphs using color to indicate the third dimension. These can be thought of as heat maps, where red indicates values closer to 1 and blue indicates values closer to 0. For example, Figure 2 is top-down view of the graph in Figure 1. For the gray-scale versions of these graphs, the upper left portion are the red values close to 1, the dividing lines are the transitional areas, and the lower right portion are the blue values close to 0.

Two interesting regions are the "slices" where $k = 1$ (along the y-axis in Figure 2), representing anycast requests, and $k = m$ (along the diagonal in Figure 2), representing multicast requests. As expected, anycast is close to 1 (red) and so can be satisfied very easily as long as m is not too small, while multicast is almost always 0 (blue) and so very difficult to satisfy unless m is very small.

Two interesting observations can be made about the transition from high delivery probability (left-red) to low delivery probability (right-blue). First, the transition happens relatively quickly, as indicated by the thin white band. This means that if the difficulty of an application's request is close to the transition point, it can increase its success drastically if it is willing to decrease k slightly. Second, the transition line is seemingly linear in nature. This means that if m decreases (e.g., nodes leave the group), then to keep a similar level of success, k must decrease proportionally. Hence, if the slope of the transition line is known, applications can adjust their requests accordingly when group size changes without actually knowing the exact group size.

Message lifetime obviously has an impact on the success of a manycast request. We evaluate this effect by using using a message expiration time of 1 hour, essentially reducing $c \cdot t$ by one half. Since nodes now have

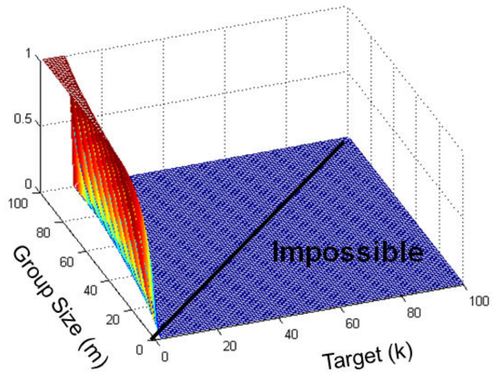


Figure 4: Very Quick Expiration (10 Minutes)

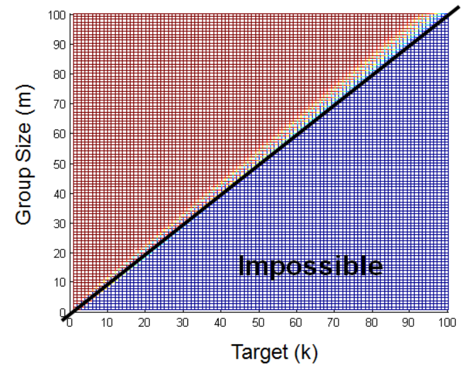


Figure 6: Very Long Expiration (5 Hours)

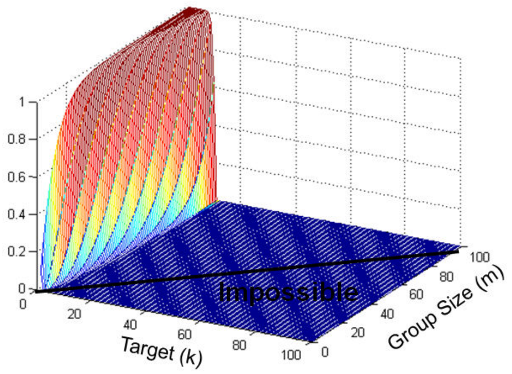


Figure 5: Very Quick Expiration (10 Minutes)

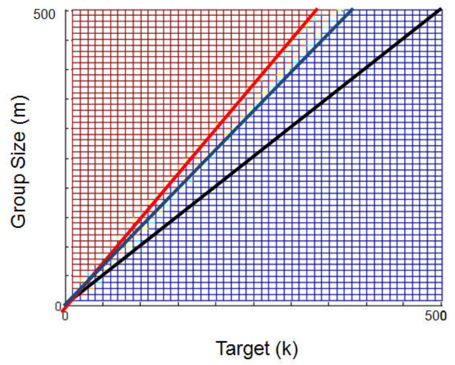


Figure 7: 500 Node Network

a shorter amount of time to deliver messages, this increases the difficulty of all requests. All other system parameters remain the same. The result is a shift in the transition line, as shown in Figure 3. In essence, changing c or t results in a change in the slope of the transition line. Therefore, some requests that had a high probability of success changed to having a very low probability of success, indicating that message expiration time is a critical factor in determining success. Interestingly, the width and linearity of the slope remained relatively unchanged. Since some applications would like to be able to predict the success probability of a given request within a given lifetime, we can also use these evaluations to find the minimum value of t, for a given k and m, that would result in a high success probability.

We have looked at manycast in a network with reasonable system parameters. However, it is also interesting to consider how manycast performs with extreme system parameters. Therefore, P has been reevaluated for very small and very large values of ct. Consider first a very small value of t, namely *10 minutes* (see Figure 4). As expected, the transitional region has shifted very close to the anycast "slice", indicating that, unless k is quite small, requests in general have a low chance of success. Perhaps more interestingly, though, is that the linearity of the transitional region breaks. Instead, it seems more exponential in nature. This implies that even with a large value of m, k must be small to have a reasonable chance of success. To better illustrate what is occurring at lower values of m, Figure 5 is a rotated version of Figure 4. From this view, it can be seen that even anycast requests (e.g., $k = 1$) have a very low chance of success when m is small.

On the other extreme, consider a very large value of t, namely *5 hours*. The top-down view of this graph, shown in Figure 6 clearly indicates almost all requests can be satisfied with a high degree of certainty. However, it is interesting to note that multicast requests (e.g., $k = m$) still have a low probability of success, particularly when m is large. This further confirms that the multicast paradigm is simply too hard to satisfy in DTN environments. In fact, as the graph indicates, it is much easier to satisfy "almost all members of a group" than "all members of the group". We refer to the *almost all* paradigm as *loose multicast*, and will further show via simulation that loose multicast is substantially easier that strict multicast.

Finally, to understand the impact of network size, we evaluate a large network of 500 nodes. For this network, the contact rate is set to 2 nodes per minute and messages expire after 5 hours. The resulting graph, shown in Figure 7, further confirms a linear, thin transition line. As a visual guide, we have included solid lines indicating the left and right edges of the transition.

In conclusion, analyzing P for varying values of m and k, as well as with different system parameters, can lead to many interesting and useful observations. Some of the more prominent ones include: (1) the clear division of very high and very low probability regions, indicating the need for routing protocols to dynamically shift their approach based on the application request and (2) a dramatic increase in success if the application is willing to relax requests that fall close to the transition line. To gain a better understanding of how real protocols in more realistic environments handle manycast requests, the following section continues the discussion of the difficulty of manycast in a simulation environment.

4. SIMULATION STUDY OF MANYCAST

The difficulty analysis presented in the previous section gives insight into the relative difficulty of an (m, k) manycast request, which in turn provides guidance on routing-level decisions such as replication. This section incorporates realism into the equation by studying how effective different classes of DTN routing protocols are and how different types of mobility factor in. For this evaluation, a popular DTN simulator called the Opportunistic Network Environment (ONE) simulator is used [11].

4.1 Evaluation Criteria

To gain a broad understanding of manycast performance, two major components are explored: routing mechanisms and mobility.

Although designed for unicast communication, many of the current unicast protocols use mechanisms that can support manycast. One of our goals is to understand which routing mechanisms can best be used for manycast and so can be integrated into our target manycast protocol. Based on the level of replication used, these mechanisms can be divided into four classes: direct delivery, quota-based, flooding, and epidemic. *Direct delivery* is the most basic form of DTN routing, where a node simply carries around messages it sources until the destinations are directly met. No forwarding ever occurs, and hence this can be considered the most resource-friendly protocol. *Quota-based* protocols allow limited forwarding and replication to improve delivery, but reduce resource usage (e.g., Spray and Wait [19], Spray and Focus [20], and Encounter-based Routing (EBR) [13]). Essentially, every sourced message contains a quota, which is a hard limit on the number of replicas of the message allowed in the system. This is enforced by decreasing the quota of a message upon replication. Next, *flooding-based* protocols take advantage of abundantly available in-network storage and are allowed to freely replicate to any or all contacts, without limit (e.g., Prophet [12], MaxProp [4], and RAPID [2]). These protocols work well in highly disconnected environments; however, they can quickly overwhelm resources in resource-constrained environments. Finally, while technically a flooding-based protocol, Epidemic routing [21] attempts to replicate all messages to all nodes in the network. This is a popular protocol due to the fact that it is optimal, in terms of delivery ratio and latency, if there are no resource constraints in the network. This protocol can be improved upon by smart buffer management techniques [18].

To properly evaluate how these protocols handle manycast requests, we choose to implement (or use existing implementations in the simulator) one protocol per class as a representative of that routing class: Directly Delivery, Spray and Focus, Prophet, and Epidemic. We also implemented a "group-based" version of these protocols, where destinations are groups, not individual nodes. Any utility functions utilized by the protocols have been adapted to capture group utility instead of node utility. This is done by having members of the same group "look" like the same node from the perspective of utility functions in the routing protocols. In other words, groups look and act like virtual nodes. The utility functions used in the routing protocols update for a particular group whenever a group member is met. The protocol labels in the results are appended with "-G" to further emphasize this.

The second evaluation criteria is mobility. In our analysis from the previous section, we assumed a very simple connection model, where a node had an equally likely chance of meeting any other node at any time. Simulation allows us to understand manycast in a wider range of mobility patterns. There are two main types of mobility that are critical to the understanding of DTN routing: unstructured mobility and structured mobility. By unstructured mobility, we mean that there is very little actual structure that can be extracted from the movement patterns of nodes (i.e., random waypoint and random walk [5]). Many DTN unicast protocols are analyzed by their performance in these types of unstructured mobility. For instance, the binary quota distribution technique used by Spray and Wait has been shown to be optimal in random mobility [19]. While less realistic, this type of mobility is generally easier to analyze. On the other hand, structured mobility can be thought of as mobility patterns that generally arise from nodes that follow different types of movement patterns, possibly related to their environment. For instance, in a disaster response scenario, emergency responders may be moving towards an event, civilians may be fleeing from it, and ambulances may be oscillating to and from it [14]. Another example is a community network, which could be composed of pedestrians, cars, and trams [8].

Structure from these networks (i.e., popularity) can be extracted and exploited for routing purposes [13, 9]. To explore manycast in both types of environments, our simulations use both random waypoint as well as the built-in community model of the ONE simulator.

4.2 Simulation Setup

The goal of our simulations is to understand how manycast requests perform under various classes of routing protocols and various types of DTN environments. Simulations are divided into two main classes related to the mobility pattern: unstructured and structured. The unstructured environment is random waypoint, with each node moving at a speed between 1 and 10 meters per second and waiting at the waypoint for a random period of time between 0 and 2 minutes. The structured environment is the build-in community mobility model, which places pedestrians, cars, and trams on a real map of Helsinki, Finland. Pedestrians walk at a speed of 0.5 to 1.5 meters per second, cars travel at a speed of 2.7 to 13.9 meters per second, and trams travel at a speed of 7 to 10 meters per second. These nodes follow intuitive routes to and from local hot-spots. The total map size for the random waypoint mobility model is 3.5km x 3.5km, while the structured mobility model is 4.5km x 3.4km. Within each of these two classes, we explore how the routers react in small groups (where $m = 16$) and larger groups (where $m = 32$). Each graph contains results from each of the aforementioned routing protocols, with the x-axis being the target number of nodes to reach (k), ranging from 1 (anycast) to m (multicast).

The total number of nodes in the simulation is 126. In the structured mobility model, there are 80 pedestrians, 40 cars, and 6 trams. Each node has a communication range of 100m, transmits at 256kbps, and has a buffer size of 5MB, except trams which have a communication range of 1000m, transmit at 10Mbps, and have a buffer size of 50MB. Messages are generated randomly by every node every 50 to 70 seconds, with a size randomly chosen between 500kB and 1MB. This setup allows for a somewhat resource-constrained environment. Each simulation is run for 4000 seconds and each data point is the average of 10 runs and includes a 95% confidence interval.

Simulations are evaluated using both group-based message delivery ratio (MDR) as well as group-based latency. MDR is defined as the number of successfully completed manycast requests (e.g., the message reached at least k of the m nodes) divided by the total number of manycast requests. The *Average MDR* is the average of each node's MDR. Latency, or delay, is defined as the time from message source until the time that the k^{th} node of the group received the multicast message. *Average delay* is the average of all message delays in the network. Note that a message can only have a delay if it was successfully delivered, and hence this metric should be viewed only in relation to the average MDR. If two protocols have widely differing average MDRs, then the average delay is less meaningful. For this reason, we consider average MDR to be the *primary metric of evaluation* and the average delay to be the *secondary metric of evaluation.*

4.3 Structured Mobility

The first class presented uses *structured mobility*, specifically the community mobility model built into the ONE simulator. Within this class, we first consider a group size of 16. The first major observation, as seen in Figure 8(a), is that no single protocol is dominate over all values of k in terms of message delivery ratio. This immediately confirms that an efficient manycast protocol must dynamically shift techniques depending on the individual request. When $k < 8$, Spray and Focus clearly obtains the best performance; however, when $k > 11$, Prophet is superior. Note that the downward slope of Spray and Focus is greater than both Prophet and Epidemic. This exposes an interesting feature of quota-based protocols, in that they can be considered more risky than flooding-based ones. Essentially, quota-based protocols can perform very well when the target number of nodes to meet is relatively small. Limiting the number of replications keeps resources from being overwhelmed, which can lead to message drops and missed contact opportunities, and at the same time is still be sufficient for reaching the target number of nodes. On the other hand, they perform very poorly when the target number of nodes is relatively large, since limiting the number of replications does not get the message out fast enough to a large fraction of the network.

The results found in Figure 8(a) can be broken down further by considering four different regions, which we refer to as regions A, B, C, and D. Viewing results such as these in terms of discrete regions hints at how a dynamic manycast protocol can be developed, which is explored in Section 5. We define region A as the region where Direct Delivery and quota-based protocols are the top performers. It can be seen that region A includes $k = 1$ (and hence anycast requests) and $k = 2$. Region B is defined as the region where quota-based protocols alone are superior. This region includes values of k from 3 to 9, in this figure. Region C is defined as the region where quota-based and flooding-based protocols are best. Hence this can be considered the region where k ranges from 10 to 13. And finally, region D is defined as the region where flooding-based protocols are dominant over all others. This includes values of k from 14 to 16 (and hence includes multicast requests).

It is important to comment on the behavior of pure epidemic routing. While epidemic routing is considered

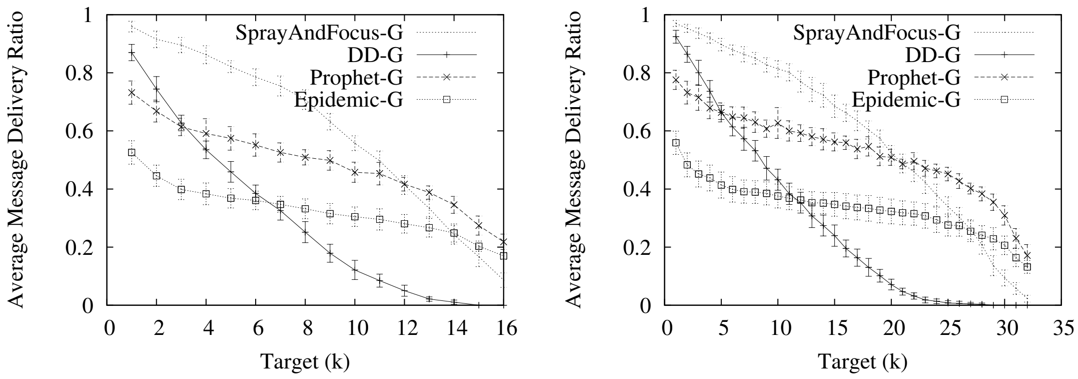


Figure 8: MDR - Structured Mobility: (a) 16 Node Groups, (b) 32 Node Groups

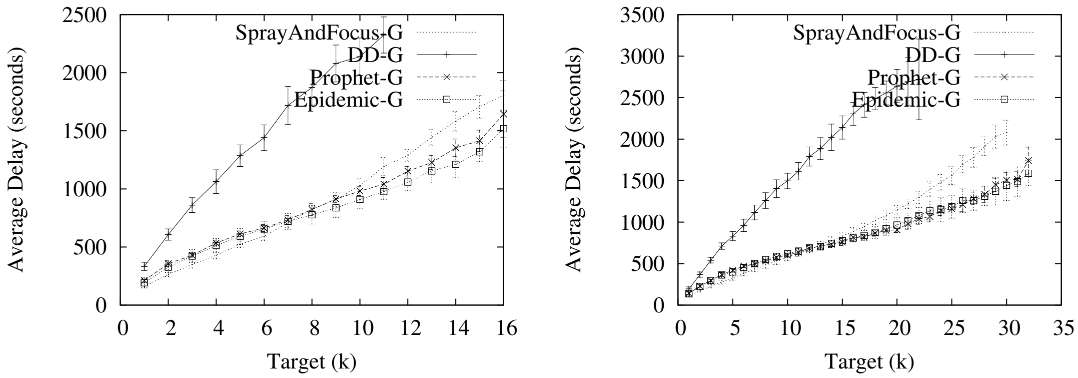


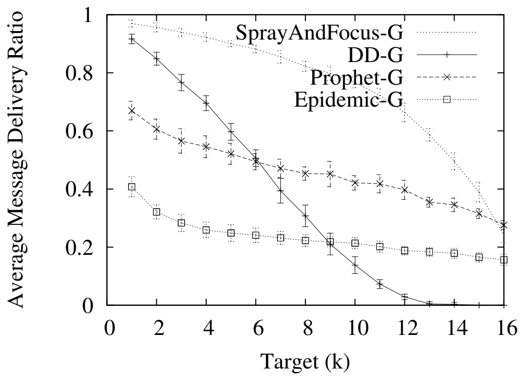
Figure 9: Delay - Structured Mobility: (a) 16 Node Groups, (b) 32 Node Groups

optimal when there are no resource constraints, it has been shown many times before that its performance is severely hindered when bandwidth, buffer size, and contact duration are limited [13, 18, 12, 15]. Our results further confirm this behavior for manycast.

When the group size is increased to 32, as shown in Figure 8(b), the characteristics of the graph stay the same. Primarily, the point at which flooding-based protocols overtake quota-based protocols stays in proportion to the group size. This is actually quite a significant observation since it provides further evidence that to keep the same success ratio, k must be increased proportionally to the increase in m. Recall that this behavior was seen as a linear transition line in the MATLAB evaluation. To be clear, in Figure 8(a) (when $m = 16$), the Spray and Focus MDR crosses the Prophet MDR at around $k = 11$; in ratio form, this is $\frac{11}{16} = 0.6875$. In Figure 8(b) (when $m = 32$), the two cross at around $k = 22$; in ratio form, $\frac{22}{32} = 0.6875$. Hence, the crossing point for quota-based and flooding-based protocols seems to occur in constant proportion to the group size.

Another interesting observation, when $m = 32$, is the relatively sharp drop-off as k approaches m. This further confirms the difficulty of multicast in DTNs, and gives support for the theory that applications willing to relax multicast requests will experience significantly higher success ratios. The relative ranges covered by regions A, B, C, and D, in relation to the group size, can be considered the same as with $m = 16$, due to the similar crossing points. Hence region A contains $1 \leq k \leq 4$, region B contains $5 \leq k \leq 18$, region C contains $18 \leq k \leq 26$, and region D contains $27 \leq k \leq 32$.

In terms of average delay, it is clear that Direct Delivery is substantially worse than the other protocols for all cases except anycast, where $k = 1$, as shown in Figures 9(a) and 9(b). This is because messages are carried only by the source nodes, and hence the source node itself would have to meet all k of the target nodes. It is interesting to note that while the resource-friendly property of Direct Delivery can help its average MDR in resource-constrained environments, it will not help its average delay. Therefore, if delay is a critical factor for the application, a Direct Delivery routing protocol would be a poor choice. Another interesting observation is that, as noted with MDR, the average delay characteristics are similar between small and large groups. The other three protocols are relatively similar until k gets large. When $k \approx \frac{2}{3}m$, Spray and Focus starts to diverge. This reinforces the idea that flooding-based protocols perform best when k approaches m.



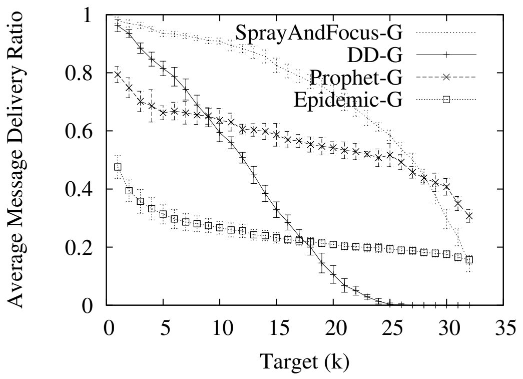
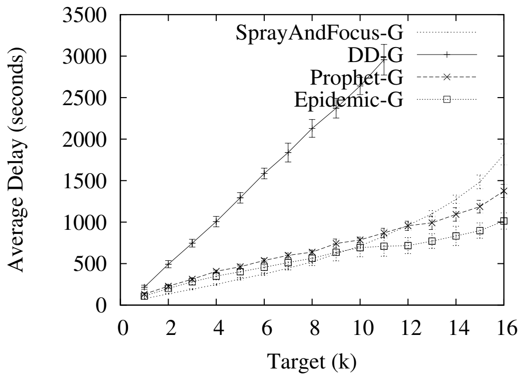


Figure 10: MDR - Unstructured Mobility: (a) 16 Node Groups, (b) 32 Node Groups



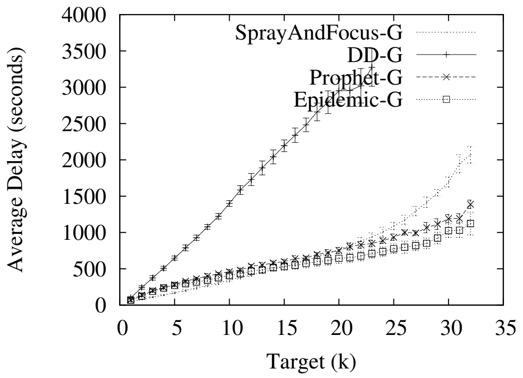


Figure 11: Delay - Unstructured Mobility: (a) 16 Node Groups, (b) 32 Node Groups

4.4 Unstructured Mobility

The second set of results evaluates manycast under *unstructured mobility* using the random waypoint mobility model. As before, we first present results where group sizes are relatively small, namely 16 nodes. In contrast to the previous results, Spray and Focus consistently performs at the highest level, as shown in Figure 10(a). This is due to unstructured, random mobility allowing message replicas to spread better throughout the network [19]. In structured mobility environments, protocols that limit replication have to deal with the possibility that most of the replicas will stay in a relatively local area. However, in unstructured, random mobility environments, nodes tend to have a higher degree of mixing. For this same reason, Direct Delivery also performs at a high rate for a longer period of time. Overall, this leads to the interesting observation that limiting replication is most beneficial to networks whose nodes mix well with one another. It is also worth noting the relatively sharp drop-off for Spray and Focus and Prophet from $k = 15$ to $k = 16$. This illustrates the difficulty of multicast in DTN environments.

In terms of dividing the figure into regions, there is no point where flooding-based protocols are convincingly better than quota-based protocols. Therefore, we can divide the graph into 3 regions, eliminating region D. Region A includes $k = 1$ and $k = 2$, where Direct Delivery and Spray and Focus both perform at a high level. Region B includes $3 \leq k \leq 14$, where Spray and Focus has a clear dominance over all other protocols. And finally, region C includes $k = 15$ and $k = 16$, where Spray and Focus as well as Prophet perform well.

With a larger group size, namely $m = 32$, the most interesting feature is the sharp drop-off in MDR as k approaches m, as seen in Figure 10(b) as well as the other MDR figures previously presented. This common thread indicates that *loose* multicast, where applications are satisfied if almost all of the group is reached, will have a much greater chance of success than strict multicast. It is therefore advantageous for DTN applications to accept and make use of loose multicast if they want to significantly improve their message delivery ratios. Note that, in our simulations, Epidemic is never superior to Prophet, since Prophet is better at managing resources efficiently.

To capture the trends, this figure can be divided to four regions, since there is a clear point when flooding-based protocols perform best. Region A can be viewed as the region where $1 \leq k \leq 3$. Region B contains the region where $3 \leq k \leq 24$. Region C can be defined as $25 \leq k \leq 29$. Finally, region D includes $30 \leq k \leq 32$.

The average delay trends of the protocols in the unstructured environment are similar to that of structured environments. As seen in Figures 11(a) and 11(b), Direct Delivery incurs the largest average delay by far for all cases except anycast. All of the protocols have a somewhat linear trend until k approaches m. Prophet, Epidemic, and Spray and Focus all quickly increase as k approaches m, with Spray and Focus being the most pronounced. This further emphasizes the difficulty of multicast requests, and strongly suggests that applications consider loose multicast.

5. A MANYCAST META-PROTOCOL

Given the trends exposed in our evaluation, it is clear that no one protocol mechanism performs best all of the time. Therefore, a successful manycast solution for DTNs should be dynamic and change replication techniques *per request* as necessary to achieve the best performance. In this section, we present a discussion of general guidelines that can be used for handling requests, and build a manycast *meta-protocol* framework based on the observations from the previous sections. Essentially, the goal of this meta-protocol is to select a protocol from the appropriate replication class such that it maximizes the average message delivery ratio. This can be thought of as “staying on top of the curve”.

There are three main factors to consider when deciding whether to use no replication, little replication, or a lot of replication. Additionally, these factors change with every request, and hence must be re-evaluated per-request. The first factor is the target number of nodes, k, of the request. If k is small, less replication is necessary to achieve success. If k is large, more replication is necessary. The second factor is the network and group characteristics. If the mobility of the network is structured or nodes do not mix evenly, then quota-based protocols may have a harder time properly distributing replicas. In this case, more replication may be necessary. On the other hand, if the mobility of the network is unstructured, where nodes mix relatively evenly, quota-based protocols are sufficient in many cases. Furthermore, the group size of the request's destination group will influence the decision. While not directly explored in this paper, resources such as battery life also fall into the “network characteristics” property. If battery life is a major constraint, then less replication is desirable. The third factor is the application's tolerance to delay. This factor is dependent on the request and, hence, will change per request. If low delay is important, then Direct Delivery should never be favored.

Using these observations, a general framework for routing manycast requests can be constructed. Recall from Section 4 that the network and group characteristics, the second factor in our previous discussion, can be used to break the range of k into four regions. If k falls in region A, Direct Delivery or quota-based protocols can be used. If k falls in region B, quota-based protocols alone are superior. If k falls in region C, quota-based or flooding-based protocols can be used. And if k falls in region D, flooding-based protocols are preferred. Therefore, the meta-protocol will take the following steps: (1) Divide the k range into four regions based on the network and group characteristics: A, B, C, and D (note that some regions may be empty, such as region D as shown in Figure 10(a)); (2) if the request is time-sensitive, eliminate region A, and extend region B to cover it; (3) consider the target number of nodes, k, and determine which region the request falls in; and (4) select a routing protocol from the appropriate class based on the region

In a more algorithmic form, a general skeleton for the dynamic manycast protocol can be seen in Algorithm 1.

Algorithm 1 Dynamic Manycast Meta-Protocol

```
$m \leftarrow size(request.destGroup)$
$regions \leftarrow getRegions(networkState, m)$
if $request.timeSensity$ then
    $regions.B = regions.B \cup regions.A$
    $regions.A = EMPTY$
end if
$reg = whichRegion(regions, request.k)$
if $reg == A$ then
    $protocol = selectFromClass(DD \cup QUOTA)$
else if $reg == B$ then
    $protocol = selectFromClass(QUOTA)$
else if $reg == C$ then
    $protocol = selectFromClass(QUOTA \cup FLOODING)$
else if $reg == D$ then
    $protocol = selectFromClass(FLOODING)$
end if
return $protocol$
```

This algorithm can help a node decide which low-level protocol to use for routing a specific request. The algorithm should be run only at the source node. Any intermediate nodes simply route the message based on the protocol originally selected by the source node. Therefore, the overall process would be as follows. First, an application generates a manycast request. The meta-routing protocol at the source selects a low-level routing protocol to use for the request. Finally, the network routes the request, using the low-level protocol originally decided on by the source meta-routing protocol.

Using a meta-protocol such as this allows applications to be very specific in their requests, and in turn allows for a much richer DTN application space. As future work, we plan to understand the interaction between the low-level protocols all working together in the same network.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have explored the concept of *many-cast* routing, where an application desires to reach at least k of m members of a group, where m is the group size. This very general paradigm inherently incorporates more specific group-based paradigms such as anycast and multicast. Through thorough analysis and simulation, we have quantified the difficulty of many-cast requests in relation to one another, and illustrated the need for a dynamic manycast protocol that changes techniques on a per request basis. Utilizing these discoveries, we demonstrated a practical approach to many-cast routing by using a meta-protocol to appropriately select a low-level routing protocol based on network factors and the specific request.

In the future, we plan to understand how different DTN protocols interact with each other while running simultaneously. Our results from this paper show that a dynamic manycast protocol is necessary to change the replication rate on a per packet basis. Taking this a step further, we plan to thoroughly explore how the replication decisions from one request affect the delivery rate and other metrics of subsequent requests; in other words, we will explore the interplay between requests that are routed using different routing techniques. Furthermore, we plan to extend our results to include resources such as battery life, which will force a new trade-off regarding replication. Finally, we plan to implement our protocol and explore its characteristics on live testbeds such as DieselNet [23].

7. REFERENCES

[1] M. Abdulla and R. Simon. A simulation analysis of multicasting in delay tolerant networks. In *Proc. of Winter Simulation Conference*, 2006.

[2] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. ACM SIGCOMM*, 2007.

[3] S. Bhattachargee, M. Ammar, E. Zegura, N. Shah, and Z. Fei. Application layer anycasting. In *Proceedings of IEEE INFOCOM*, 1997.

[4] J. Burgess, Br. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *WCMC: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2:483–502, 2002.

[6] Casey Carter, Seung Yi, Prashant Ratanchandani, and Robin Kravets. Manycast: exploring the space between anycast and multicast in ad hoc networks. In *Proceedings of ACM MobiCom 03*, 2003.

[7] Ederson Rosa da Silva and Paulo Guardieiro. Anycast routing in delay tolerant networks using genetic algorithms for route decision. In *Proceedings of IDCS*, 2008.

[8] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *ACM MobilityModels Workshop*, 2008.

[9] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *ACM MobiHoc*, 2008.

[10] Y. Gong, Y. Xiong, Q. Zhang, Z. Zhang, W. Wang, and Z. Xu. Anycast routing in delay tolerant networks. In *GLOBECOM*, 2006.

[11] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proceedings of SIMUTools '09*, 2009.

[12] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *MobiHoc 03*, 2003.

[13] S. Nelson, M. Bahkt, and R. Kravets. Encounter-based routing in dtns. *Proceedings of IEEE InfoCom*, 2009.

[14] S. Nelson, A. Harris, and R. Kravets. Event-driven, role-based mobility in disaster recovery networks. In *Proc. of ACM CHANTS*, 2007.

[15] S. Nelson and R. Kravets. Achieving anycast in dtns by enhancing existing unicast protocols. In *Proc. of the ACM CHANTS workshop*, 2010.

[16] S. Nelson and R. Kravets. For members only: Local and robust group management in dtns. In *Proc. of the ACM CHANTS workshop*, 2010.

[17] V.D. Park and J.P. Macker. Anycast routing for mobile networking. In *Proceedings of IEEE Military Communications Conference*, 1999.

[18] Ram Ramanathan and Regina Rosales-hain. Prioritized epidemic for routing in opportunistic networks. In *In Proc. ACM MobiOpp*, 2007.

[19] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of ACM WDTN '05*, 2005.

[20] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Proceedings of IEEE PerCom*, 2007.

[21] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Department of Computer Science, Duke University, April 2000.

[22] Seung Yi, Seung Yi, and Robin Kravets. Moca : Mobile certificate authority for wireless ad hoc. In *PKI Workshop*, 2003.

[23] X. Zhang, J. Kurose, B. Levine, D. Towsley, and H. Zhang. Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact on Routing. In *Proc. ACM Mobicom*, 2007.