Understanding and Developing a Dynamic Manycast Solution for DTNs

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 derive structure from human-based interaction and mobility, and hence group-based routing is both a natural and important paradigm for applications. In this work, we introduce and explore the concept 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. Efficiently handling manycast requests at the routing layer allows for both feasible and flexible manycast applications.

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 drastically changing their routing approach, in order 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 human-based interaction and mobility. Furthermore, these networks can be highly heterogeneous and include smart phones, emergency response devices, sensors, laptops, and even vehicles. Unfortunately, the historical, Internet-style design principle of point-to-point communication (e.g., unicast) has carried over into the DTN realm, severely hindering what could be a rich and diverse medium for applications.

Due to both the heterogeneity of devices and the social structure inherent in DTNs, it is our position that multiple routing paradigms must be supported at the network layer in order to provide the flexibility needed for a rich application space. While unicast is still clearly useful, group-based routing paradigms are a natural and flexible means for applications that consider groups to be first-class entities. Group-based routing differs from unicast in that it allows applications to send data to one or more members of a particular group, without having to specify the individual destination nodes. Interestingly, DTNs inherently lead themselves well to group-based communication due to their broadcast nature and heavy use of replication in existing protocols.

In this paper, we introduce and explore the concept of manycast routing in a DTN environment. Manycast can be thought of as an umbrella routing paradigm, which incorporates the entirety of group-based routing, where the goal is to transmit a copy of a message to an application-specified number of members of a particular group. For instance, an emergency response application, run by a civilian trapped in a building, could utilize the network to reach \textit{at least one} emergency responder. This illustrates one well-known extreme of the manycast spectrum – anycast. The other extreme is multicast, where the goal is to reach every member of a particular group. One example of multicast includes an update application pushing out updates to a group of devices. These popular extremes only represent two ends of the broad spectrum, and, while useful, are not alone sufficient. Take, for instance, a sensor reading application that wishes to collect a statistically significant sample of readings from a group of sensors. Anycasting
would clearly be insufficient, and manycasting to the entire group would be extremely inefficient. The application should have the flexibility to specify the target number of nodes to reach, with the network layer dynamically responding to meet that specific request.

Providing efficient manycast in a DTN environment has many challenges. In addition to the standard challenges of all DTN communication, such as intermittent connectivity, heavy partition, high variance in resource constraints, and the lack of instantaneous end-to-end paths, group-based communication has the added difficulty of group management [17]. Furthermore, manycast has the added difficulty of handling two routing parameters, which can vastly change with each application request: the target group size and the target number of group members to reach. 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 [16], while multicast requires a lot of replication for success [1]. These preliminary results indicate to us that there is a wide variance in difficulty of manycast requests, depending on the application-specified target number of group members to reach.

This work takes a three-pronged approach towards the understand and development of manycast in DTNs. First, we perform an extensive analysis to increase our theoretical understanding of the fundamental difficult of manycast requests in a DTN environment. We use MATLAB to visualize this space, and draw conclusions about the difficulty of anycast, manycast, and all points in-between. Many useful and practical conclusions are drawn, such as loose multicast, where meeting almost all nodes in a group is considered a success, is a substantially easier paradigm than strict multicast. 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 requests. The conclusion of this study is one of the major contributions of this work – in particular, we show that for a manycast protocol to be effective in a DTN environment, it must dynamically and drastically change its routing and replication approach on a per-message basis. Third, using conclusions drawn from the 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

The heterogeneity of mobile devices, as well as the inherent human-centric structure found in many DTNs, make group-based communication a natural and useful requirement to obtain a rich, flexible application space. The umbrella routing paradigm of manycast incorporates group-based communication in its most general form. In this section, we specifically define manycast and show how it incorporates other, well-known group-based routing paradigms. Furthermore, we discuss how efficient, network-layer manycast provides the necessary flexibility for DTN applications.

We define a manycast request 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 [17], as opposed to the actual application. The parameter \( k \) is the target number of nodes to meet 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.

Manycast is a highly general paradigm that captures other, well-known group-based paradigms. When \( k = 1 \), the request is analogous to anycast. Similarly, when \( k = m \), the request is analogous to multicast. Depending on how groups are defined, manycast is also general enough to include both unicast (e.g., \( k = m = 1 \)) and broadcast (e.g., \( k = m = n \), where \( n \) is the number of nodes in the network). This is interesting, since it implies that understanding the difficulty of a generic manycast request (see Section 3) directly helps the understanding of the relative difficulty between anycast, unicast, multicast, and broadcast.

Efficient manycast routing has the ability to greatly increase the richness and diversity of DTN applications. In addition to allowing anycast- and multicast-based applications, it provides the flexibility for applications that desire to work somewhere between the two. Consider the example from Section 1, where a sensor reading application desires to obtain information from a statistically significant sample of sensor nodes. An anycast request would be insufficient, as the result would most likely not be statistically significant, and a multicast request would be overkill by unnecessarily wasting resources. A manycast routing protocol that could deliver a COLLECT message to at least \( k \) of the \( m \) sensor nodes would greatly benefit the application.
To further see the usefulness of manycast, consider an application run by first responders surveying a disaster scene. The first responders may conclude that it is necessary to bring a certain number of ambulances and/or fire response vehicles. The first responders would most likely not care which specific vehicles arrived, and so unicast would be inappropriate. Furthermore, in a DTN environment, it is unlikely that the sender could choose a precise set of nodes it could unicast to, since the reachability of the individual nodes is something not known until the message is in-transit. Anycasting the request would be insufficient, as more than one vehicle may be desired. Multicasting the request would be highly inefficient, particularly if the group of response vehicles were large. Instead, a manycast routing option where a first responder could deliver a message to \( k \) of \( m \) emergency response vehicles is needed.

The examples of manycast are numerous and bridge many types of environments. For instance, many smartphones and handheld gaming devices have WiFi and Bluetooth [2] built-in, which can be used for multiplayer gaming when friends are in close proximity. Manycast would be useful for gaming applications to find \( k \) of one’s local group of \( m \) friends to join a game. Many other examples exist in the form of contacting distributed servers for freshness or security purposes. One specific use here is threshold cryptography, where a certain subset of distributed CAs must be contacted for the security protocol to work [23].

To the best of our knowledge, this is the first work that practically explores the idea of manycast routing in a DTN environment. Manycast has been considered in wireless ad-hoc environment by Carter et al., where end-to-end paths were always assumed to exist [7]. In this work, the authors explore different techniques for building manycast trees and evaluate the techniques in terms of relays required for success. This work, however, is not directly applicable to the DTN environment, as it takes as a premise a high degree of connectivity. In DTN environments, it cannot be assumed that the routes computed for previously established manycast trees will exist in the future. Therefore, DTN manycast protocols must attempt to progress replicas of the message before knowing 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, anycast and multicast work for DTNs is also relevant. As previous work shows, anycast is a highly useful and practical routing paradigm for DTNs [16]. While anycast has been considered in wired network scenarios [4, 18], it has only briefly been explored in DTN environments, where the exploration has been limited to single-copy routing and/or highly constrained mobility [11, 8]. Multicast, the other extreme of the manycast paradigm, has only very briefly been considered in DTN environments. In particular, a simulation-based study which 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.

3. ANALYSIS OF MANYCAST DIFFICULTY

Manycast, being such a general paradigm, incorporates a very wide range of group-based routing classes. Due to this, as well as the flexibility the application should have in making a manycast request, it is necessary to understand the fundamental difficulty of a manycast request. A manycast request can be thought of as a tuple \((m, k)\), where \(k\) is the target number of nodes in a group to contact, and \(m\) is the group size. The application may or may not specify \(m\), as that could be the job of the group management component [17], but should be allowed to specify any value of \(k\) from 1 to \(m\). Therefore, there will be a high variance in request difficulty, from relatively easy (when \(k = 1\) [16]) to relatively hard (when \(k = m\) [1]). Understanding the difficulty of a request is necessary to determine which routing class or technique to use. For instance, for easy requests, quota-based or even single copy protocols are best, as they will not overwhelm resources and can afford to not have many copies in the network [16]. For hard requests, however, more replication is beneficial. We confirm these hypotheses in Section 4. Therefore, by classifying the difficulty of an \((m, k)\) multicast request, it can be understood how to best route that request. Furthermore, by understanding how to compute difficulty for such a general paradigm, the relatively difficulty of other group-based paradigms, such as anycast and multicast, is obtained for “free”.

3.1 Mathematical Analysis

We define the fundamental difficulty of a request in a probabilistic fashion: specifically, the difficulty of request \((m, k)\) is \(P(m, k)\), which is now described. A simplistic, but parametrized system model is assumed for analysis purposes; however, in Section 4 a more realistic environment is used for evaluation. Assume there are \(n\) total nodes in the network, and routing is via direct delivery. Therefore, only the source node will ever replicate a message, and will only do so to deliver the message to a group members (who does not forward it further). A node will meet another node uniformly at random, and a node can expect to meet \(c\) nodes per time unit. Further assume that messages expire after \(t\) time units from creation.

**Problem:** Given the previously described system model, we now compute \(P(m, k)\), which represents the probability of a copy of a generated message to successfully reach at least \(k\) of the \(m\) destination group mem-
To make the problem easier to grasp, we describe it in an analogous and familiar bin-ball setup. Assume there are \( n \) balls labeled 1 through \( n \) (representing the nodes; \( n - 1 \) to be 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. Assume, without loss of generality, that the destination group members have the labels 1 to \( m \). We therefore want to determine the probability that the experimenter will have seen at least \( k \) unique balls with labels less than or equal to \( m \).

As a quick example, assume \( n = 3 \), \( c = 2 \), \( t = 2 \), and \( m = 2 \). All possible games include \((1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\). Of these, only \((1,2)\) and \((2,1)\) meet the requirements for success. Therefore, we consider the difficulty of this request, under the aforementioned system parameters, \( \frac{2}{9} \). As another example, consider \( n = 4 \), \( c = 3 \), \( t = 2 \), and \( m = 3 \). We encourage the reader to work through this case, and obtain a difficulty of \( \frac{10}{81} \). Note how fast the space explodes - the number of possible games is \( n^m \).

Recall the problem: What is the probability that after \( ct \) non-unique picks one sees at least \( k \) labels less than or equal to \( m \)? In order to make the problem more trackable, we divide it into two steps: (1) given \( ct \), the chance of getting exactly \( u \) unique picks, multiplied by (2) given \( u \) unique picks, the chance of seeing at least \( k \) values less than or equal to \( m \). These steps are iterated over all reasonable values of \( u \). Let the first step be defined as \( f(ct, u) \) and the second step be defined as \( g(u, k, m) \). Note that all reasonable values of \( u \) go from \( k \) (since anything below \( k \) unique picks cannot result in success) to the minimum of \( n \) and \( ct \). Therefore, \( P \) is defined as follows, in terms of \( f \) and \( g \):

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

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 \( 1 - \frac{u-1}{n} \). We can therefore define \( f \) as a recursive function:

\[
f(ct, u) = f(ct - 1, u) \cdot \frac{u}{n} + f(ct - 1, u - 1) \cdot \left(1 - \frac{u - 1}{n}\right)
\]

The base cases for the recursion 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:

\[
g(u, k, m) = \sum_{l=k}^{m} \binom{m}{l} \binom{n-m}{u-l}
\]

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

In order to thoroughly understand how the 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. It is worth noting that memoization was used to both speed up and cut down on the memory consumption of the recursively defined \( f \) function. We leave finding a closed form for \( f \) as future work.

The goal of this analysis is to understand how difficult a request is, given the group size \( m \), and the target number of group members \( k \) to receive the message. It is also important to understand how this difficulty changes as one or both of these parameters change. Two capture how \( P \) varies with varying values of \( m \) and \( k \), the results are presented as 3D graphs. The two control variables are \( m \), 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 \( P(m, k) \).
First, a 100 node network is considered where messages expire after 2 hours and the nodes meet each other at a rate of once per minute. Therefore, \( n = 100, t = 2 \) hours, and \( c = 1 \) per minute. A 3D representation of this scenario is seen in Figure 1. There are multiple interesting sections of this graph that deserve comment. First, there is a relatively large set of values where the success probability is close to 1, meaning the request should be relatively easy to satisfy. Second, there is 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. Third, there is another relatively large portion with values close to 0, indicating requests that 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.

Since 3D graphs are somewhat difficult to view, most of the graphs presenting from here on out are 2D top-down representations using color to indicate the third dimension. This can be thought of as a heat map, where red indicates values closer to 1 and blue indicates values closer to 0. Figure 2 is top-down view of the graph in Figure 1.

Two addition regions of interest are the “slices” where \( k = 1 \), representing anycast requests, and \( k = m \), representing multicast requests. As expected, Figure 2 shows that anycast can be considered a easy paradigm when \( m \) is not very small, and multicast can be consider a hard paradigm when \( m \) is not very small.

Two interesting observations can be made regarding the transition from high delivery probability to low delivery probability. First, the transition happens relatively quickly. This indicates 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 left the group), then in order 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.

In order to understand how changing the \( ct \) product in the definition of \( P \) changes the results, the function was reevaluated using a message expiration time of 1 hour. This increases the difficulty of all requests, as routers now have a shorter amount of time to deliver messages. All other system parameters remained the same. The result is a shift in the transition line, as shown in Figure 3. In essence, changing \( c \) or \( t \) result in a change in the transition line slope. Therefore, certain requests that had a high probability of success changed to having a very low probability of success. This indicates that message expiration time is a critical factor in determining success. Interestingly, the width and linearity of the slope remained relatively unchanged.

Another type of question that can be answered by varying \( c \) or \( t \) is, given a request of \( k \) (and knowing \( m \)), determining the ideal value of \( t \) by shifting the transition line (e.g., varying \( t \)) until it meets the \((m, k)\) point.
in question. This can give the application an idea of the amount of time that must pass to ensure with, say, 95% confidence that the request succeeded.

The previous two graphs give an idea of how the success probability looks with reasonable system parameters. However, it is also interesting to consider how the graph looks 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. This graph, shown in Figure 4, is presented in 3D form to better illustrate its characteristics. As expected, the transition line has shifted very close to the anycast “slice”. This indicates 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 transition line break. Instead, it seems more exponential in nature. This implies that even with a large value of $m$, $k$ must be small in order to have a reasonable chance of success.

To better illustrate what is occurring at lower values of $m$, Figure 5 is presented as 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, a very large value of $t$ is considered, 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 meet almost all members of a group then 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, a large network of 500 nodes is considered. 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, solid lines indicating the ends of the transition are presented.

In conclusion, analyzing the $P$ function 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. In order to gain a better understanding of how real protocols in more realistic environments handle manycast requests, the following section continues the discussion of multicast difficult in a simulation environment.

4. SIMULATION STUDY OF MANYCAST

The difficulty analysis presented in the previous section gives insight into how difficult a manycast request is given \( k, m, \) and some basic system parameters. Being able to determine, on-the-fly, how easy or hard a manycast request is is the first step towards understanding how to route it. 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. A popular DTN simulator call the Opportunistic Network Environment (ONE) simulator is used [12].

4.1 Evaluation Concerns

In order to gain a broad understanding of manycast performance, two major factors are explored. First, it is necessary to determine how different classes of DTN routing protocols handle manycast requests. Current unicast DTN routing protocols can be classified based on how much replication is used. We divide these protocols into four major classes: direct delivery routing, quota-based routing, flooding-based routing, and epidemic routing.

- **Direct delivery** (which we consider a class and protocol together) 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. A slightly less resource-friendly class is **quota-based**, where forwarding and replication is allowed, but limited. 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 having quota decreased and copied to replicas during replication. Examples of quota-based protocols include Spray and Wait [20], Spray and Focus [21], and Encounter-based Routing (EBR) [14]. Continuing to an even less resource-friendly class of protocols is **flooding-based**. These protocols take advantage of abundantly available in-network storage and are allowed to freely replicate to any or all contacts, without limit. These protocols work well in highly disconnected environments, where mobility is not structured; however, they can quickly overwhelm resources in resource-constrained environments. Examples of flooding-based protocols include Prophet [13], MaxProp [5], and RAPID [3]. Finally, while technically a flooding-based protocol, Epidemic routing [22] can be considered the most resource taxing of all. This protocol 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 radio and latency, if there are no resource constraints in the network. This protocol can be improved upon by smart buffer management techniques [19].

In order to properly understand how these protocols handle manycast requests, we choose to implement one protocol per class as a representative of that routing class. Therefore, we implemented (or used the implementations in the simulator) Directly Delivery, Spray and Focus, Prophet, and Epidemic. We 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 second evaluation consideration is mobility. In the analysis, a very simple connection model was assumed, 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. There are two main types of mobility that are critical to the understanding of DTN routing: unstructured and structured. These terms are not well defined, and we use them loosely here. Unstructured mobility means there is very little actual structure that can be extracted from the movement patterns of nodes. Examples include random waypoint and random walk [6]. 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 [20]. While less realistic, this type of mobility is generally easier to analyze. On the other hand, structured mobility generally arises 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 and event, civilians may be fleeing from it, and ambulances may be oscillating to and from it [15]. Another example is a community network, which could be composed on pedestrians, cars, and trains [9]. Structure from these networks (popularity, for instance) can be extracted from these networks and exploited for routing purposes [14, 10].

In order to explore manycast in both types of envi-
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. The ONE simulator [12], a popular DTN simulator with many built-in DTN unicast routers, was used. 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 256kpbs, 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, which can be considered representative of human-centric DTNs. 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 Results

The simulation results are divided into two classes, depending on the mobility model used. Structured mobility refers to the use of the community mobility model built into the ONE simulator. Unstructured mobility refers to the use of the random waypoint mobility model, also built into the ONE simulator.

#### 4.3.1 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 due to have too large a buffer, 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 as 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. According to the figure, this region
includes values of $k$ from 3 to 9. 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 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 [14, 19, 13, 16]. Our results further confirm this behavior.

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 as 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 $k = 10$; in ratio form, this is $\frac{10}{16} = 0.625$. 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 $m$.

4.3.2 Unstructured Mobility

The second class presented uses the random waypoint mobility model, which is a form of unstructured mobility. 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 [20]. 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 in general (and hence nodes that are carrying replicas) 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.

Next, a larger group size, namely $m = 32$, is considered with unstructured mobility. The most interesting feature of this graph 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 is 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.

The figure can be divided to four regions, as 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 is 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

In the previous section, it was shown that a dynamic manycast protocol that changes replication techniques per request is necessary to achieve the best performance. In this section, we present a discussion of general guidelines that can be used for handling a request, 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”.

It has been shown in Section 4 that there are three main factors in considering whether to use no replication, little replication, or a lot of replication. These factors change with every request, and hence must be re-evaluated based on the 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, then 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 a low delay is important, then quota-based and flooding-based protocols should always be favored over Direct Delivery.

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 (e.g., 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.
4. Select a routing manycast 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.

\begin{algorithm}
\caption{Dynamic Manycast Meta-Protocol}
\begin{algorithmic}
\STATE $m \leftarrow \text{size(request.destGroup)}$
\STATE $\text{regions} \leftarrow \text{getRegions(networkState, } m\text{)}$
\IF{$\text{request.timeSensitivity}$}
\STATE $\text{regions.B} = \text{regions.B} \cup \text{regions.A}$
\STATE $\text{regions.A} = \text{EMPTY}$
\ENDIF
\STATE $\text{reg} = \text{whichRegion(regions, request.k)}$
\IF{$\text{reg} == A$}
\STATE $\text{protocol} = \text{selectFromClass}($DD$ \cup \text{QUOTA})$
\ELSIF{$\text{reg} == B$}
\STATE $\text{protocol} = \text{selectFromClass}($QUOTA$)$
\ELSIF{$\text{reg} == C$}
\STATE $\text{protocol} = \text{selectFromClass}($QUOTA$ \cup \text{FLOODING})$
\ELSIF{$\text{reg} == D$}
\STATE $\text{protocol} = \text{selectFromClass}($FLOODING$)$
\ENDIF
\STATE \text{return protocol}
\end{algorithmic}
\end{algorithm}

This algorithm can help a router decide which low-level protocol to send the request to. The algorithm should be run only at the source node; any intermediate nodes would simply route the message based on the
protocol originally selected by the algorithm. 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 for the request. Finally, the network routes the request, using the low-level protocol originally decided on by the source meta-routing protocol.

6. FUTURE DIRECTIONS AND CONCLUSIONS

Group-based routing is a natural communication paradigm in many types of DTNs, particular those that are human-centric in nature. The ability to give DTN applications flexibility in describing group-based requests is therefore of critical importance to the future of DTN communications. In this work, we have explored the concept of manycast routing, where an application desires to reach at least \( k \) of \( m \) members of a group, where \( m \) is the system 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 manycast 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 manycast 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 [24].

7. REFERENCES

Military Communications Conference, 1999.


