Dynamic Social Grouping Based Routing in a Mobile Ad-Hoc Network

Roy Cabaniss*, Sanjay Madria*, George Rush*, Abbey Trotta† and Srinivasa S. Vulli*
*Department of Computer Science,
Missouri University of Science and Technology, Rolla, Missouri 65401
†School of Computing and Engineering,
University of Missouri, Kansas City, Missouri 64110

***Abstract*—The patterns of movement used by Mobile Ad-Hoc networks are application specific, in the sense that networks use nodes which travel in different paths. When these nodes are used in experiments involving social patterns, such as wildlife tracking, algorithms which detect and use these patterns can be used to improve routing efficiency. The intent of this paper is to introduce a routing algorithm which forms a series of social groups which accurately indicate a node's regular contact patterns while dynamically shifting to represent changes to the social environment. With the social groups formed, a probabilistic routing schema is used to effectively identify which social groups have consistent contact with the base station, and route accordingly. The algorithm can be implemented dynamically, in the sense that the nodes initially have no awareness of their environment, and works to reduce overhead and message traffic while maintaining high delivery ratio.**

***Index Terms*—Social Algorithm, Routing, Grouping, Community Detection, Mobile Ad-Hoc Network, Delay-Tolerant Network**

I. Introduction

MOBILE Ad-hoc networks are a collection of computing devices connected through wireless communications, such as Bluetooth or wireless LAN. They are characterized by the mobility and dynamic nature of the devices, referred to as nodes. This mobility makes conventional routing algorithms ineffective or inapplicable, and to accommodate these environments new routing methods have been developed. These routing methods are generally examples of Delay Tolerant Routing, which holds copies of transmitted messages to transmit them to appropriate nodes, as opposed to traditional routing which broadcasts them immediately.

As a general rule, routing algorithms are more effective when they can rely on more information regarding the mobility patterns of nodes. Conventional probabilistic routing schema assumes consistent avenues of communication - nodes which interact with a set group of nodes in the past will do so again in the future [1]. Similarly, social routing assumes that nodes that are assigned to the same social network (classroom, project team) will regularly interact with members of that social group. To take advantage of the partial applicability of both social and probabilistic routing, a grouping method is studied which will form social groups based on contact patterns. With these groups identified, consistent routes to basestations are identified based on the delivery history of a group or node. We compare this algorithm, called Dynamic Social Grouping (DSG), with two well known algorithms referred as Epidemic routing [2] and Probabilistic routing [1], and results show that DSG performs better than both routing algorithms in terms of message delivery ratio and the time to deliver a message.

The objective of this algorithm is to reliably deliver messages from a group of sensor nodes to basestation nodes, as presented in Probabilistic Routing, using as few communications as possible. The basestations are immobile, and thus can use more reliable communications to transmit the data to the end user, but the sensor nodes are attached to mobile social entities. This algorithm is designed to identify social groups without relying on outside information, based only on its contact patterns with other nodes. A given node may belong to several social groups and will attempt to merge together groups who share common members. Once the social groups are identified, routing occurs through these groups based on which group has more reliable access to the basestations, as determined by the reliability of previously delivered messages. Simulations with real-world data comparing DSG to Probabilistic and Epidemic routing algorithms show that the Dynamic Social Grouping algorithm is superior in terms of delivery ratio and message costs to the Probabilistic scheme in a social environment, and much less expensive than Epidemic routing while maintaining high delivery ratio and low transit time (refer to Section IV for details). Applications of this technique include wildlife tracking or tracking human social behaviors, although it can improve any environment in which groups of regular contacts are formed.

II. Background

Research in the area of Delay Tolerant Networks (DTN) has been receiving considerable attention in the last few years owing to their widespread occurrence in a variety of applications. Routing and data aggregation are of particular interest as they affect the performance of the network as a whole and also affect longevity of the sensor nodes considerably. A survey of routing techniques for DTNs is presented in [3].

A. Epidemic Routing

Epidemic routing [2] was designed for partially connected networks, and its goal is to maximize the message delivery ratio while minimizing the time necessary to deliver a message. The main strategy is to pass messages along to each node encountered, in hopes of making a connection with parts of the network with low connectivity. In wireless sensor networks, this often results in excessive network traffic which reduces the life of the network. Also, minimizing time to message delivery is not as important in delay tolerant networks. That said that the delivery ratio of epidemic routing is considered the ideal delivery ratio possible when ignoring network life or node overflow.

B. Probabilistic Routing

Introduced by Yu Wang, the Probabilistic Routing Schema [1] is based on individual nodes having a set probability of successfully delivering a message to a base station. This probability is based on the set of neighbors which the node regularly interacts with, and as such it is based on regular, if not social, patterns of movement. Nodes individually begin with a set delivery probability and transfer the messages they are carrying to neighbors with higher delivery probability, adjusting their own probability upward as they do so. If they are carrying a message when it times out their probability is reduced to reflect the node's inability to transfer. This algorithm shows a marked improvement in terms of message transmission rate while maintaining a high delivery probability, but does not take advantage of any knowledge other than past contacts.

C. Bubble Rap

The Bubble-Rap grouping method [4] allocates nodes into social groups based on direct and indirect contacts. They distribute using a method called $k-cliques$, in which all fully connected groups of k members are considered a distinct social group and are then merged with all other $k-cliques$ which can be reached through $k-1$ nodes. This provides an accurate representation of the social groups formed by a set of nodes. However, it relies on global knowledge of the nodes' contacts, and it must have them before the algorithm can group them. In this sense, k-grouping is neither dynamic nor distributed, which limits it's applicability to a MANET.

D. SocialCast

Costa et al. described an application using social dynamics for a publish/subscribe schemata across a wireless sensor network. They detail a SocialCast model [5], in which nodes are assigned a Utility Value for each interest in which they participate which indicates their routing utility for that group. By separating the routing utility from group participation, the authors improve the lifespan and routing efficiency of the wireless nodes. The given algorithm is designed for use across a publish/subscribe model, as well as a wireless network (as opposed to a sensor network), but can still serve as a basis for further research, especially regarding the social dynamics described, such as the Caveman Model.

E. SimBet

The SimBet routing algorithm [6] has a similar routing structure to DSG in that it takes advantage of Social Grouping and contact patterns to predict paths to node destinations, improving delivery ratio and time. The algorithm calculates the Betweenness rating of a node, which is a measurement of the number of message routes which contain the node. By using this, as well as the Centrality of the node, the algorithm can route messages through to destination nodes with remarkable efficiency. A notable deficiency of this algorithm is it's reliance on near-complete knowledge of neighboring nodes, increasing traffic and memory use. It can, however, deliver messages from node to node, whereas the Dynamic Social Grouping algorithm is strictly node to base station.

III. Proposed Algorithm

The algorithm addresses two distinct subproblems. First, given an arbitrary collection of nodes, the network must identify cohesive social patterns, and identify them as groups, at the same time distributing this information throughout the network. This network organization can change to reflect changes to the network or to the social patterns of the nodes. Second, once the networks have been identified, a route must be identified to deliver messages to a base station. Delivery must minimize the number of message repetitions while ensuring a high percentage of messages delivered to the destination.

A. Grouping

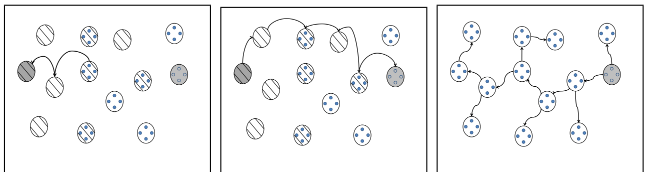
A major advantage this algorithm has over conventional mobile ad-hoc routing methods is the use of social groups to improve communication throughput. The task of identifying such groups, however, requires knowledge of the nodes which is not present at startup.

1) Contact Strength: The first task in identifying a social group is to calculate a metric for the contact frequency two nodes have with one another. The symbol $\lambda_{i,j}$ is used to represent the contact strength between two nodes. This is measured as a function of the time from the previous contact, where the symbol ϕ is used to determine how much the λ changes based on new data. Initially, $\lambda_{i,j}$ will be 0 between all nodes, but when nodes contact one another, it is recalculated as shown in Eqn 1.

$$\lambda_{A,B} = (1-\phi)\lambda_{A,B} + \frac{\phi}{time_{current} - time_{prev}} \tag{1}$$

2) Forming New Groups: When the $\lambda_{i,j}$ exceeds a certain threshold ψ, the nodes can identify as being members of the same group. Initially, all groups will have two members. The node with the higher delivery probability (as defined in section III-B1) be designated the group clusterhead - this node has the responsibility of maintaining the group membership list and approving any changes to the group. The group will also have its probability set to the average of the two founding members (this will be expanded upon in section III-B).

3) Merging: Having formed two-node groups, the next step is to determine which of these groups can be merged with each other to form larger, more applicable social groups. Groups are merged when the similarity between the groups exceeds a threshold value τ. Similarity is defined as the number of



(a) Joint members of the Dot Group and Stripe Group send a Suggestion to the Stripe Clusterhead

(b) Stripe Clusterhead confirms the Group Merge and sends an Invitation to the Dot Clusterhead

(c) The Dot Clusterhead confirms again and adds all Stripe Members to the Dot Group

Fig. 1. Three Stages of Merging Groups

common members divided by the total number of members in both groups. Each node checks through its list of groups periodically to see if the groups should be merged. If it finds two groups that qualify for a merger, it sends a suggestion message to the clusterhead of the smaller group, as shown in Figure 1(a). This message is distributed epidemically through that group until it arrives at the clusterhead, which contains the up-to-date member list of the group. Once the clusterhead receives the suggestion it compares it's member list with the member list for the larger group (contained within the suggestion message), confirms that the groups should be merged, and sends an invitation message to the clusterhead of the larger group (Figure 1(b)). The larger group's clusterhead repeats the confirmation. Once confirmed, the clusterhead updates the member list to include the new members from the smaller group (Figure 1(c)). A Kill message is then sent to remove the smaller group. This process is detailed in Algorithm 1.

4) Dynamic Grouping: Due to changing social patterns, group membership needs to allow nodes to remove themselves from a group's membership, as shown in Algorithm 2. To reduce communication overhead and group fragmentation, however, the clusterhead has to know about the resignation immediately. When a node contacts the clusterhead it will calculate it's $Group\lambda_{A,Y}$, which is the average $\lambda_{A,B}$ between the node and all members of the group, to determine whether it should resign from the group. If this average λ is below the threshold ψ, it sends a resignation to the clusterhead and removes the group data. This method has the advantage of low overhead and communication, but nodes cannot leave the group until they contact the clusterhead, which can result in nodes maintaining group data for groups they don't participate in. Experiments which allowed nodes to periodically review their groups and resign through a message sent to any group member resulted in badly fragmented group data.

5) Group Versions: A constant issue when creating dynamic groups is reflecting changes to the group data to all members of that group. Data fragmentation occurs whenever two nodes, members of the same Social Group, have different data regarding that group's membership. To prevent this issue, all changes are controlled by the group's Clusterhead. This node, arbitrarily chosen, holds the group's master member list. As movement patterns change or are revealed, there will be changes to a group's membership. The Clusterhead will increment the version value of the group as changes are made, including merges and nodes resigning from the group. Whenever two members of the same group meet, they compare the version number of their local copy of the group. The higher version number is the one whose information regarding the group came from the clusterhead more recently. This information is copied over to the less recent node, along with the version number. This process is detailed in Algorithm 3.

Algorithm 1 Merging Groups

```
Notation
$Group_Y$, $Group_Z$ - Any social groups in MANET
$Node_A$ - Member of both $Group_Y$ and $Group_Z$
$Node_B$ - Clusterhead of $Group_Y$, $Clusterhead_Y$
$Node_C$ - Clusterhead of $Group_Z$, $Clusterhead_Z$
$\tau$ - Threshold for Merging a Group

Trigger - Periodically in $Node_A$
for all $Group_Y, Group_Z$ of which $Node_A$ is a member do
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ then
    if $|Group_Y| > |Group_Z|$ then
      Send SUGGEST Message to all members of $Group_Z$
    else
      Send SUGGEST Message to all members of $Group_Y$
    end if
  end if
end for

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$...
for all $Group_Y$ which contain both $Node_A$ and $Node_B$ do
  if $Node_A$ has Control Messages for $Group_Y$ then
    $Node_A$ sends Control Messages to $Node_B$
  end if
end for

Nodes pass Control Messages epidemically to all Group Members

Trigger - $Node_B$ receives SUGGEST message
if $Node_B$ is $Clusterhead_Y$ then
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ then
    Send INVITE Message to all members of $Group_Z$
  end if
end if

The Clusterhead confirms the suggestion, and sends an Invite to the other group

Trigger - $Node_C$ receives INVITE message
if $Node_C$ is $Clusterhead_Z$ then
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ then
    $Group_Z = Group_Z \bigcup Group_Y$
    $Group_Y = \varnothing$
    Send KILL message to $Group_Y$
  end if
end if

If both Group Clusterheads approve, the smaller group is added to the larger, and then removed.
```

B. Routing

Having organized the nodes into social groups, the algorithm can now use this information to route data to the base station. The Probabilistic method assigns a metric to each node to depict the node's chance of successfully delivering the message, and continually routes messages to higher performing

Algorithm 2 Dynamic Groups

```
Notation
$Node_A$ - Member of $Group_Y$
$Node_B$ - Clusterhead of $Group_Y$, $Clusterhead_Y$
$NodeList_Y$ - All nodes in $Group_Y$
$Group\lambda_{A,Y}$ - Contact Strength between $Node_A$ and $NodeList_Y$
$\psi$ - Threshold for Forming a Group

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
if $Node_B$ is $Clusterhead_Y$ then
   $Group\lambda_Y = Average(\lambda_{A,Y})$
   if $Group\lambda_Y < \psi$ then
      Send RESIGN to $Node_B$
      Remove $Group_Y$ from $Node_B$
   end if
end if

Trigger - $Node_B$ receives RESIGN from $Node_A$
$NodeList_Y = NodeList_Y - Node_A$
```

Algorithm 3 Group Updates

```
Notation
$Version_{A,Y}$ - Version Number of $Group_Y$ kept by $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
for all $Group_Y$ which contains both $Node_A$ and $Node_B$ do
   Send $Version_{A,Y}$ to $Node_B$
   Receive $Version_{B,Y}$ from $Node_B$
   if $Version_{A,Y} > Version_{B,Y}$ then
      Send $Group_Y$ to $Node_B$
   else if $Version_{A,Y} < Version_{B,Y}$ then
      Receive $Group_Y$ from $Node_B$
      $Version_{A,Y} = Version_{B,Y}$
   end if
end for
```

nodes until the message reaches a destination. To improve on this the DSG identifies a similar metric to measure a group's ability to deliver a message to the base station. Both methods are described below.

1) Individual Probability: Nodes are initially assigned a default probability σ, while base stations are assigned a probability of 100%. When nodes encounter one another, they compare probabilities to determine routing (more on this in section III-B3). The member with the lower probability will forward all its messages to the other, and then it will update its σ to reflect the ability of the node to deliver either directly or indirectly to a base station. The result is that nodes with immediate access to a base station achieve a higher probability, and the probability cascades outwards through node contacts. The cascade rate is determined by a control variable α, which is similar to ϕ in that is controls how quickly the probability changes based on new data. For details, review Algorithm 4.

In addition to message transmission, the Individual Probability σ can also update to reflect inability to deliver a message. Messages log the time they were originally sent, and can use this to determine if they have been in the system too long. When these messages expire, all nodes which contain the message have their individual probability reduced to reflect their inability to reach a sink.

Algorithm 4 Calculating Individual Probability

```
Notation
$Node_A$, $Node_B$ - Nodes in MANET
$\sigma_A$ - Individual Probability for $Node_A$
$\sigma_B$ - Individual Probability for $Node_B$
$message_i$ - Message in MANET
$timeSent_i$ - Time $message_i$ was sent
$timeOut$ - Parameter determining max duration of messages
$\alpha$ - Control Parameter determining Probability decay ratio

Trigger - $Node_A$ transmits $message_i$ to $Node_B$
$\sigma_A = (1-\alpha)\sigma_A + \alpha\sigma_B$

Trigger - Periodical maintenance in $Node_A$
for all $message_i$ in $Node_A$ do
   if $time_{current} - timeSent_i > TimeOut$ then
      Remove $message_i$
      $\sigma_A = (1-\alpha)\sigma_A$
   end if
end for
```

2) Group Probability : Each node calculates group probability, depicted as β, independently, based on the contact patterns of that specific node. This means that each node will have different values for the probability of the same group, but these values are based on the subset of the group which each node contacts. Nodes which exist in a common group and are not encountered are estimated using the current group probability. The exact method, described also in Algorithm 5, begins when $Node_A$ contacts $Node_B$ and both are in $Group_Y$. Since the σ for all other nodes in $Group_Y$ are unknown, they are assumed to be the current β_Y. It then calculates the average probability of all members of the group, based on previous β_Y, σ_A, and σ_B, as detailed in Algorithm 5.

Algorithm 5 Calculating Group Probability

```
Definition
$\beta_Y$ - Group Probability for $Group_Y$

Trigger - $Node_A$ contacts $Node_B$
$Node_A$ sends $\sigma_A$ to $Node_B$
$Node_A$ receives $\sigma_B$ from $Node_B$
for all $Group_Y$ which contain $Node_B$ and $Node_A$ do
   $\beta_Y = \frac{\sigma_A + \sigma_B + \beta_Y \times (|Group_Y| - 2)}{|Group_Y|}$
end for
```

3) Using Probabilities to Route: When two nodes contact each other, they independently determine their γ value. This is the maximum probability of all the groups they participate with and their individual probability. The node with the higher value is assumed to either have more consistent contact with the base station or to be a member of a group which has consistent contact. In either event, the node with the lower value transfers all messages to the node with the higher probability, then updates its individual probability based on successful delivery of a message. This algorithm is detailed in Algorithm 6.

Previously, the individual probability was adjusted by the individual probability, but as the system gains more information it can update based on the groups it contacts, rather than the individual nodes. For this reason, the individual probability (Algorithm 4) is adjusted to use the γ_B of the destination node, rather than the individual probability σ_B.

Algorithm 6 Routing Algorithm

```
Notation
$\beta_A$ - List of all Group Probabilities in $Node_A$
$\gamma_A$ - Max Group / Individual Probability of $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
$\gamma_A = max(\beta_A, \sigma_A)$
Transmit $\gamma$ to $Node_B$
Receive $\gamma_B$ from $Node_B$
if $\gamma_B > \gamma_A$ then
    Transmit messages to $Node_B$
    $\sigma_A = (1 - \alpha)\sigma_a + \alpha\gamma_B$
else
    Receive messages from $Node_B$
end if
```

4) Individual Message Probability: A method was introduced by Yu Wang [1] to reduce the overhead of the system by calculating an individual messages probability of delivery. If a probability exceeds a certain threshold then messages, upon being forwarded, will be removed from the sending node. Initially, the probability of the individual message is the same as the σ of the sending node. When transferred to another node, the message's probability is updated to include the chance of the new node successfully delivering the message (which is simpler to calculate as the inverse of the probabilities of all nodes involved failing). This algorithm is a useful approximation which allows nodes to reduce redundant messages in the environment, and can be implemented for minimal cost, but it relies on the accuracy of σ as a metric for the node's delivery probability. Depending on the control parameters used, the σ value may not be an accurate representation for the node's probability to deliver a message. For this reason, the threshold value was set to be very high, to prevent removal of messages until the system was virtually guaranteed to have delivered the message. The procedure is detailed in Algorithm 7.

Algorithm 7 Individual Message Probability

```
Notation
$prob_i$ - Probability of $Message_i$ being successfully delivered
$\sigma_A$ - Individual Probability of $Node_A$
$\sigma_B$ - Individual Probability of $Node_B$

Trigger - $Node_A$ transfers $Message_i$ to $Node_B$
In $Node_A$
$prob_i = 1 - (1 - prob_i) * (1 - \sigma_B)$
Transmit $Message_i$ to $Node_B$
if $prob_i > .999999$ then
    Remove $message_i$ from $Node_A$
end if
```

Control Variables		Range Tested	Ideal Value
α	Probability Decay Rate	0.05 - 0.5	0.075
ψ	Group Formation Threshold	0.0001 - 0.005	0.004
τ	Group Merge Threshold	0.1 - 0.33	0.3
ϕ	Contact Decay Ratio	0.1 - 0.5	0.3
Ideal values tested by experimentation			
Variables			
λ	Contact Strength between nodes		
σ	Node's Probability of Delivery		
β	Group's Probability of Delivery		
γ	Maximum probability of Delivery		

TABLE I
VARIABLE REFERENCE CHART

IV. ANALYSIS

A. Experimental Setup and Evaluation

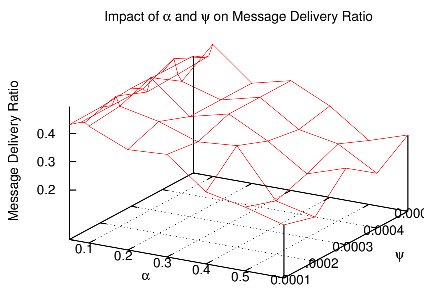
To evaluate this algorithm and the impact of control variables a simulation was designed and implemented using MATLAB. Once the routing efficiency was determined, a power consumption comparison was performed in NesC, using a TOSSIM simulator modified with PowerTossim Z. For comparison, both simulations also ran Epidemic Routing and base Probabilistic Routing schema. Epidemic Routing is considered to be the ideal in terms of message time and delivery ratio, and is therefore used for comparison, while Probabilistic Routing is an improvement over Epidemic in terms of resources used. Other comparable algorithms were reviewed, but either use previous knowledge of the environment (such as Bubble Rap) or are targeted to solve different communication issues (Simbet performing node-to-node communication, while SocialCast performing publish/subscribe broadcasts). For these reasons, the Probabilistic and Epidemic schemas were considered as comparable algorithms.

B. Simulation Data Source

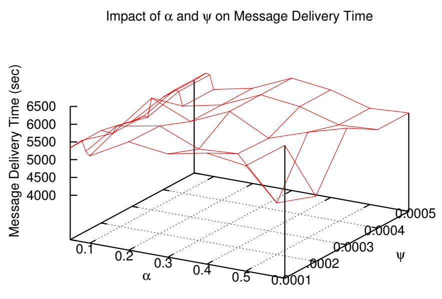
To accurately implement this algorithm, the node contacts must follow social patterns. Potential sources for this are either real-life tracking data or the results from a social prediction algorithm, such as the Caveman Model. For this reason the simulation used data obtained from an experiment conducted by the University of Cambridge at the 2005 Grand Hyatt Miami IEEE Infocom conference. Participants were asked to carry iMotes with them during the conference for 3 to 4 days to capture data on social interactions [7]. Although the experiment did not include base stations capable of collecting information, they did include static immobile nodes. For simulation purposes, these units were treated as data sinks.

C. Impact of α and ψ

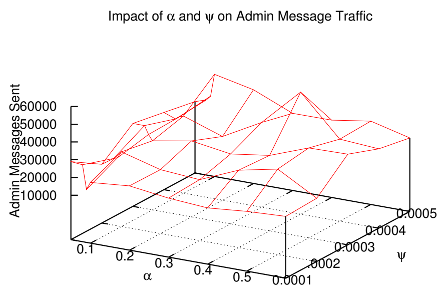
Based on the dataset used, the α setting is directly influenced by the dynamic nature of the contact patterns. In either long range simulations or simulations in which nodes consistently follow similar contact patterns, a lower value for α can result in more effective simulations. In contrast, a higher α results in the algorithm placing more weight in recent data. If the probability is more dynamic, it can more rapidly respond to changes in the layout.



(a) Message Delivery Ratio

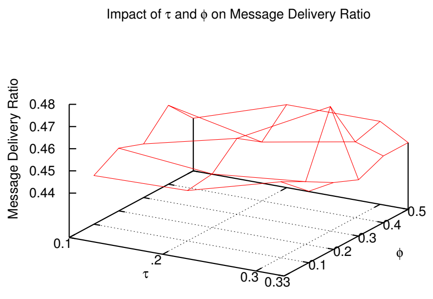


(b) Message Delivery Time

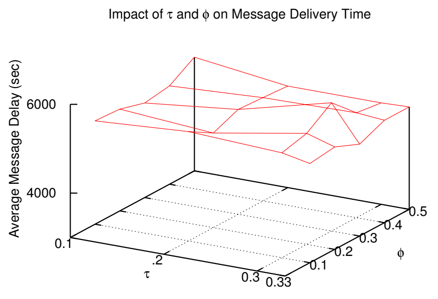


(c) Admin Message Traffic

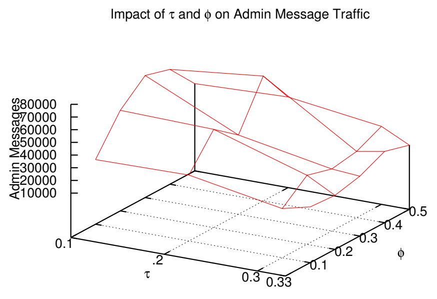
Fig. 2. Impact of α and ψ



(a) Message Delivery Ratio



(b) Message Delivery Time



(c) Admin Message Traffic

Fig. 3. Impact of τ and ϕ

The ψ setting influences the ease with which the nodes form social groups. In an environment in which nodes only encounter other nodes they are in a social group of, a lower value can result in the network being rapidly organized, whereas a higher ψ results in groups forming slowly, and only with regular contacts.

The results (as shown in Figure 2) indicate a peak Message Delivery Ratio as α decreases and ψ increases. The lower α seems to be due to the large amount of 'garbage' data - a node which successfully delivers to the basestation is likely just passing through, and will not likely be a good long-range contact. A lower α allows the algorithm to ignore these incidental contacts and concentrate on nodes which regularly successfully deliver. Similarly, the ψ peak value is based on ignoring the large number of casual contacts. Because this dataset occurs at a conference, there are several regular contacts between nodes which are not members of the same group. To ignore this noise, a higher ψ setting is optimal.

D. Impact of τ and ϕ

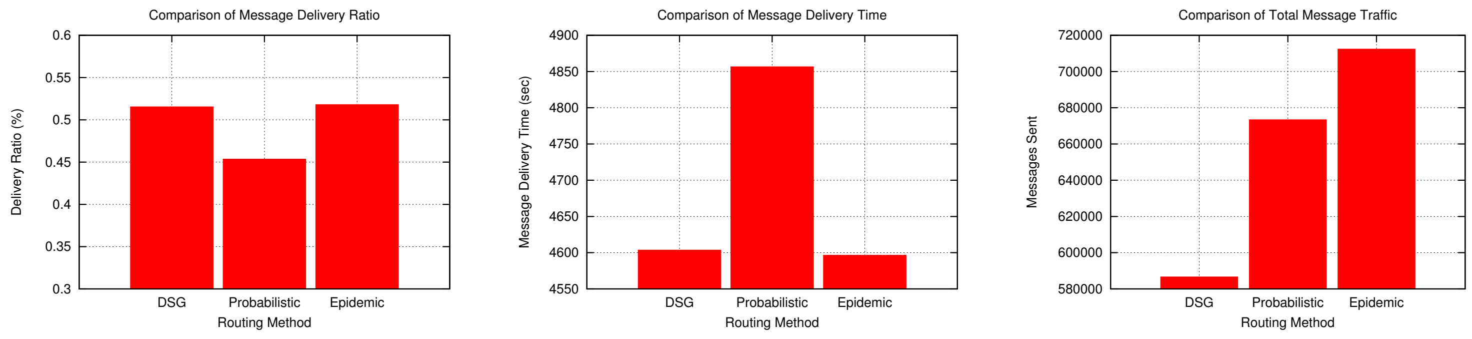
The control variable τ represents the level of similarity necessary for groups to be merged. A higher value for τ results in fewer groups being merged and keeps the group size small. This does not influence the number of groups being created; a higher value of τ will result in several smaller groups. Due to the ratio needed to ensure any group merges occur, the maximum value for τ is $\frac{1}{3}$. If it is higher, the initial 2-member groups cannot have enough common members to merge.

The contact deterioration ϕ determines how quickly the contact strength changes over time. A higher value for ϕ results in the contact strength changing more rapidly. Higher values would be used in more dynamic environment, so the nodes' contact strength are more influenced by recent events than historical data. This value is closely linked to the ψ value given earlier; more rapidly changing contact strength would result in needing a lower value of ψ to successfully form groups.

The results, shown in Figure 3, indicate that the algorithm functions best when the τ value is nearly, but not quite, at the maximum value. A τ of 0.3 allows groups to merge regularly, reflecting larger groups while still ignoring the casual group affiliations. Similarly, the ϕ results show peak functionality at the value 0.3, although this may be due to the ψ value already inserted. These metrics show the largest increase of delivery ratio, but also increase the average message delivery time (Figure 3(b)). That is due to this time only reflecting delivered messages – by increasing delivered messages, the system includes several messages which were ignored.

E. Comparison of Routing Algorithms

A review of the results (Figure 4) shows that the DSG Routing Scheme performs better than the Probabilistic Scheme in all metrics, and is comparable to the Epidemic in terms



(a) Message Delivery Ratio (b) Message Delivery Time (c) Total Message Traffic

Fig. 4. Comparison of DSG Routing, Probabilistic Routing, and Epidemic Routing

of both the delivery ratio and delivery time while having considerably less message traffic. Some of this is due to the applicability of the dataset – the social environment provided at the conference provided an area ideal for forming short-term dynamic groups. In contrast, the Probabilistic Schema had difficulty adjusting to the social groups that formed, and took too long to cascade the probabilities that would result from successful delivery. The results indicate that DSG is comparable to the ideal routing, with considerably less cost.

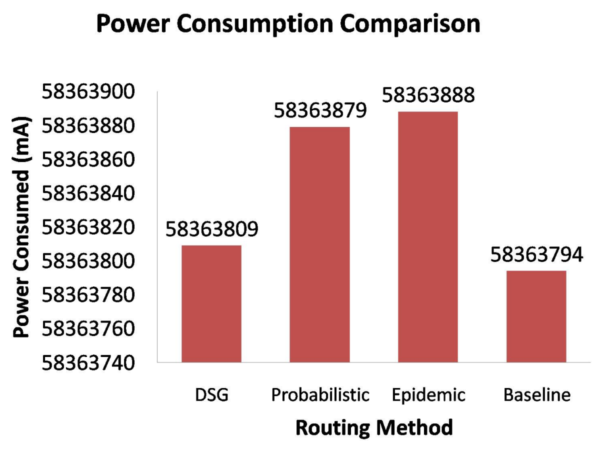
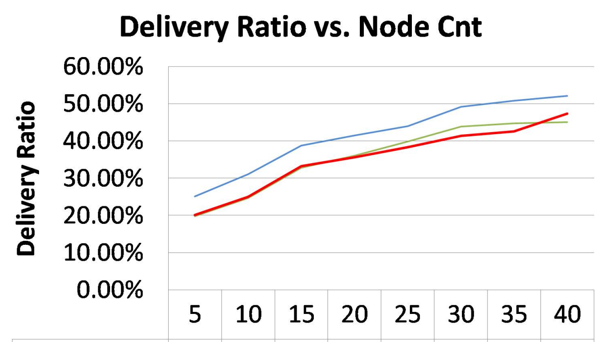


Fig. 5. Comparison of DSG, Probabilistic, and Epidemic Power Consumption

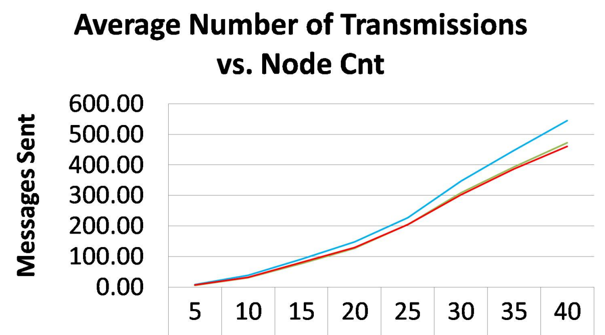
A cost comparison of the algorithms was performed in the TOSSIM simulation, using the PowerTOSSIMZ [8] model, as shown in Figure 5. The baseline shown in the graph shows the simulation's power comsumption when no routing occurs, to be used as a comparison point. We use this to find that the Dynamic Social Grouping's cost is 16% of Epidemic's cost, and 18% of the base Probabilistic algorithm's cost. This includes the overhead from group management such as group merge invitations, suggestions, and membership updates, but not standard maintenance such as neighbor discovery or message generation.

When considering a routing algorithm for large-scale deployment, there is always concern by how well the algorithm



	5	10	15	20	25	30	35	40
Probabilistic	19.7	24.6	32.8	36.0	39.8	43.8	44.7	45.1
Epidemic	25.0	31.1	38.7	41.5	44.0	49.1	50.8	52.1
DSG	20.1	25.0	33.2	35.6	38.3	41.4	42.5	47.3

(a) Comparison of Delivery Probability versus Node Count



	5	10	15	20	25	30	35	40
Probabilistic	6.08	30.3	77.2	127.	205.	309.	392.	472.
Epidemic	7.60	38.0	91.2	147.	227.	347.	446.	545.
DSG	6.36	31.5	80.1	129.	205.	302.	386.	460.

(b) Comparison of Total Message Traffic versus Node Count

Fig. 6. Scalability Experiments

scales to a higher number of nodes. For this reason, scalability experiments were performed to find the performance of the DSG algorithm with differing number of nodes. Because the experiment was performed using a real-world dataset, it is impossible to add additional nodes to the experiment, but it is simple to randomly remove nodes from consideration. The results show that the ratios from the three algorithms tend to increase as the number of nodes increases, improving connectivity, but the DSG algorithm doesn't show a marked improvement until the full 40 nodes are in use (Figure 6(a)). This is due the same control variables being used through the entire experiment set, from 5 to 40 nodes. As indicated earlier, the algorithm is sensitive to the control variables used during execution (detailed in Table I) - by using these variables despite the different environments, performance fluctuates wildly. Nonetheless, the costs of the system remains consistently lower than the alternatives (Figure 6(b)). This is due to the nature of the administration traffic. As the number of nodes decreases, reducing node connectivity, the administration overhead is reduced. Fewer groups are being formed, merged, or updated. As the connectivity increases, it improves routing performance by enough to offset the overhead.

V. Conclusions

The Dynamic Social Grouping (DSG) algorithm has been shown to provide a significant increase in efficiency over probabilistic routing and epidemic routing. While maintaining a lower overhead than either epidemic or probabilistic routing, DSG has managed to achieve as high a message delivery ratio and as low a delivery time as can be expected. This will lead to better data aggregation and longer battery life, both of which are primary goals in modern wireless sensor networks. There remains further research to consider the implications of alternative merge methods. The current method using a simple ratio depicting commonality may not be optimal, and experiments in alternative methods may provide further improvement. It may also be improved by introducing a load-balancing method to equalize the drain on commonly used nodes, further improving the network lifespan. Load balancing techniques can also change the clusterhead, to spread energy drain among the group. A final improvement to consider is the expansion of this algorithm to include point-to-point communication, expanding its impact to include a wider range of applications.

References

[1] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, 2007.

[2] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., 2000.

[3] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[4] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," pp. 241–250, 2008.

[5] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 748–760, June 2008.

[6] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," New York, NY, USA, pp. 32–40, 2007.

[7] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2009-05-29)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[8] E. Perla, A. O. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "Powertossim z: realistic energy modelling for wireless sensor network environments," in *PM2HW2N '08: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. New York, NY, USA: ACM, 2008, pp. 35–42.