Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs

Outline

- Introduction
- The characteristic calculation for routing
  - Centrality
  - Similarity
- SimBet Routing algorithm
- Evaluation
- Conclusion

Introduction

- In spare Mobile Ad Hoc network (MANET), node density is low, and contacts between nodes do not occur every frequently
- Traditional MANET routing protocols cannot be used in spare MANET
- The use of social network analysis techniques

Centrality

- A quantification of the relative importance of a vertex with in the graph
- A node with high centrality has a strong capability of connecting other network members.
- Three most widely used centrality measures
  - Freeman's degree
  - Closeness
  - Betweenness

Centrality: Degree

- The number of direct ties that involve a given node

$$C_D(p_i) = \sum_{k=1}^{N} \alpha(p_i, p_k)$$

- Where $\alpha(p_i, p_k) = 1$ if a direct link exists between p_i and p_k and $i \neq k$

Centrality: Closeness

- Measure the reciprocal of the mean geodesic distance,
- Distance is the shortest path between a node and all other reachable nodes

$$C_c(p_i) = \frac{N-1}{\sum_{k=1}^{N} d(p_i, p_k)}$$

- Regarded as a measure of how long it will take information to spread from a give node to other nodes

Centrality: Betweenness

- Measure the extent to which a node lies on the paths linking other nodes

$$C_B(p_i) = \sum_{j=1}^{N} \sum_{k=1}^{j-1} \frac{g_{jk}(p_i)}{g_{jk}}$$

- Where g_{jk} is the total number of geodesic paths linking p_j and p_k.
- $g_{jk}(p_i)$ is the number of those geodesic paths that include p_i

Centrality

- Degree centrality can easily be measured for a ego network
- Closeness centrality is uninformative in an ego network
- Betweenness centrality in ego networks has shown to be quite a good measure when compared to that of the sociocentric measure

Similarity

- There is a heightened probability of two people being acquainted if they have one or more other acquaintance in common.
- The probability of a future collaboration:

$$P(x, y) = \left| N(x) \cap N(y) \right|$$

- The probability captures the similarity between node x and y.

SimBet Routing

- Routing based on betweenness centrality and similarity
- No assumption of global knowledge
- Forwarding decisions are based solely on local calculation

SimBet Routing: Betweenness calculation

- Node contacts can be represented by an nxn symmetric matrix A
- n is the number of contracts a given node has encountered

$$A_{ij} = \begin{cases} 1 & \text{if there is a contact between I and j} \\ 0 & \text{otherwise} \end{cases}$$

SimBet Routing: Betweenness calculation

- Betweenness is calculated by computing the number of nodes that are directly connected through the ego node
- The sum of the reciprocals of the entries of

$$A^2[1-A]_{ij}$$

$$w8 = \begin{array}{cc} & \begin{array}{ccccc} w8 & w6 & w7 & w9 & s4 \end{array} \\ \begin{array}{c} w8 \\ w6 \\ w7 \\ w9 \\ s4 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$w8^2[1-w8] = \begin{array}{cc} & \begin{array}{ccccc} w8 & w6 & w7 & w9 & s4 \end{array} \\ \begin{array}{c} w8 \\ w6 \\ w7 \\ w9 \\ s4 \end{array} & \begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & 3 \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix} \end{array}$$

SimBet Routing: Similarity calculation

- For nodes with directed contract, the similarity can be gotten directly from the matrix A
- For indirect encounters, we maintain a separate n x m matrix,
- n is the number of nodes that have been met directly
- m is the number of nodes that have not directly been encountered, but may be indirectly accessible through a direct contact

SimBet Routing: SimBet utility calculation

- The similarity utility $SimUtil_n$ and the betweenness utility $BetUtil_n$ of node *n* for delivering a message to destination node *d* compared to node *m* is given by:

$$SimUtil_n(d) = \frac{Sim_n(d)}{Sim_n(d) + Sim_m(d)}$$

$$BetUtil_n = \frac{Bet_n}{Bet_n + Bet_m}$$

SimBet Routing: SimBet utility calculation

$$SimBetUtil_n = \alpha SimUtil_n(d) + \beta BetUtil_n$$

- Where α and β are tunable parameters and $\alpha + \beta = 1$

SimBet routing: algorithm

- Node n verifies that node m is a new neighbor
- If yes, message destined for m are delivered
- encounter request is sent, and m replies with a list of nodes it has encountered
- This list of contacts is used to update the betweenness value and the similarity value on node *n*
- Exchange a summary vector containing a list of destination nodes they are currently carrying messages for along with their own locally determined betweenness value and the similarity value for each destination

SimBet routing: algorithm

- node *n* calculates the SimBet utility of node *n* and node *m*
- If node *n* has a higher SimBet utility, the destination is added to a vector of destinations for which messages are requested
- node *n* sends the message request list to node *m*
- Node *m* removes all messages requested from its queue and forwards them to node *n*.

SimBet routing: algorithm

- 1: **upon** reception of Hello message *h* from node *m* **do**
- 2: **if** newNeighbour(*m*) == true
- 3: **if** *msgQueue*.hasMsgsForDest(*m*) == true
- 4: deliverMsgs(*m*)
- 5: requestEncounters(*m*)
- 6:
- 7: **upon** reception of encounter vector *ev* from node *m* **do**
- 8: addNodeEncounters(*m*, *ev*)
- 9: updateBetweenness()
- 10: updateSimilarity()
- 11: exchangeSummaryVector(*m*)
- 12:
- 13: **upon** reception of summary vector *sv* from node *m* **do**
- 14: Vector *requestMsgs*
- 15: **for all** *destinations* $\in$ *sv* **do**

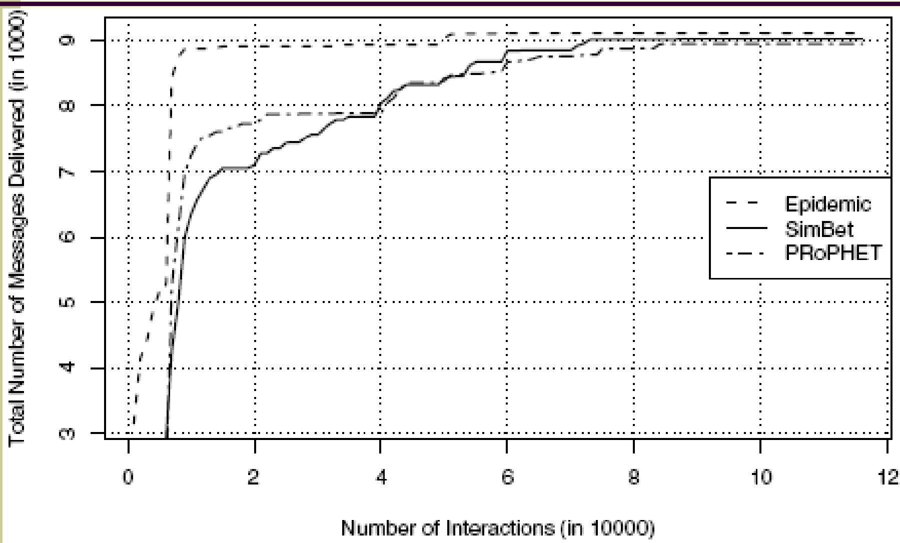
SimBet routing: algorithm

- **if** *m*.simBet(*d*) < simBet(*d*)
- *requestMsgs*.add(*d*)
- sendMsgRequest(*m*, *requestMsgs*)
- **upon** reception of message request vector *mrv* from node *m*
- **do**
- Vector *transferMsgs*
- f**or all** *messages* $\in$ *mrv* **do**
- *transferMsgs*.add(*msgQueue*.getMsgs(*d*))
- sendTransferMsgs(*m*, *transferMsgs*)
- **upon** reception of transfer message *tm* from node *m* **do**
- *msgQueue*.add(*tm*)

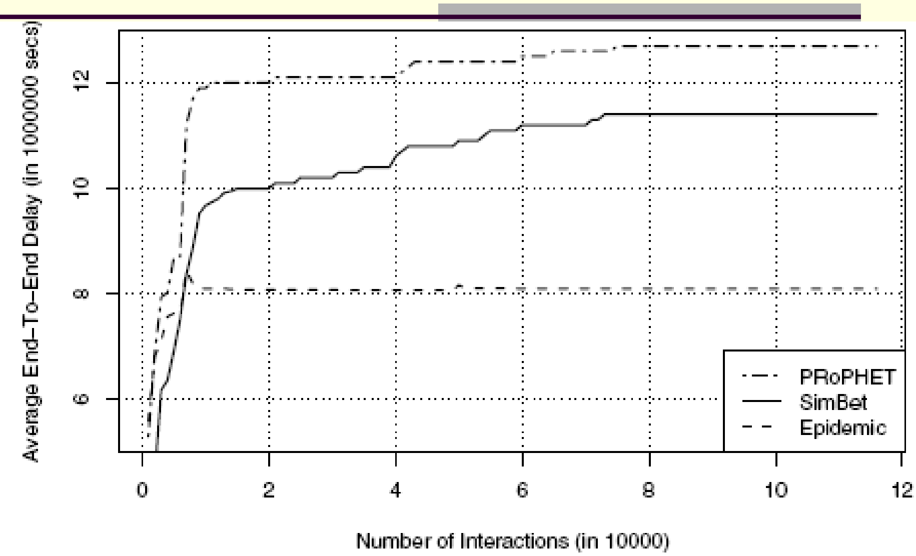
Evaluation result

- The performance comparison between epidemic, Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) and SimBet routing.

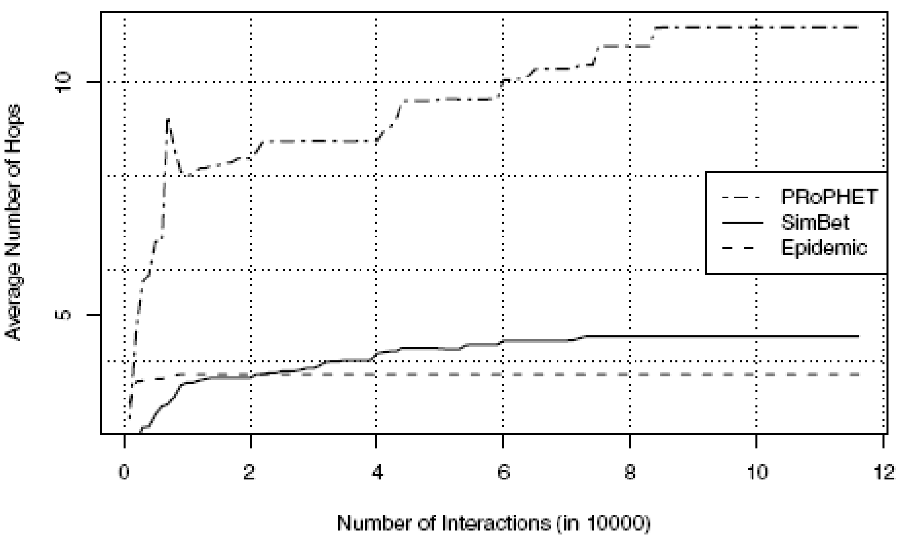
Evaluation



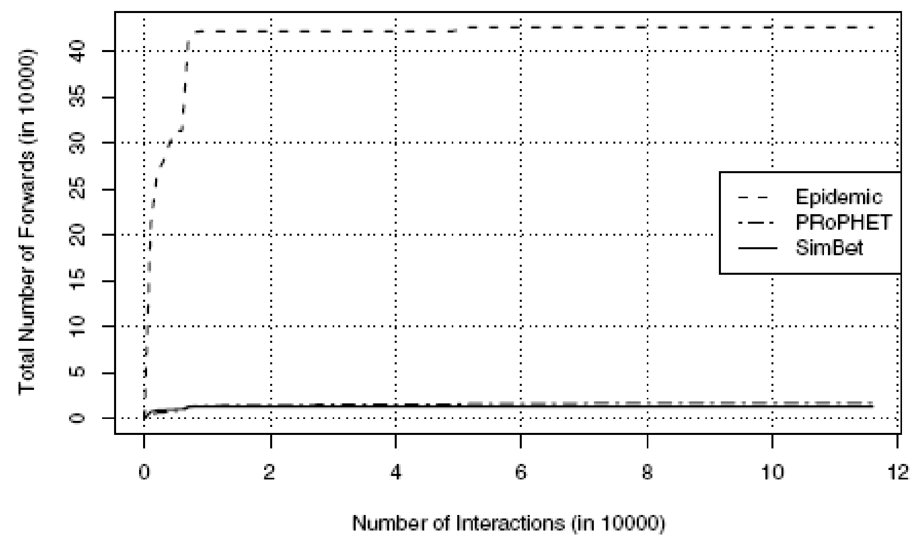
(a) Total Number of Messages Delivered



(b) End-to-End Delay (secs)



(c) Average Number of Hops per Message



(d) Total Number of Forwards

Conclusion

- The conception and the calculation of centrality
- SimBet routing algorithm
- The evaluation and performance comparison