Efficient Resource Discovery in Mobile Ad Hoc Networks

Ravi Thanawala and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: {rthanawa@, jie@cse.}fau.edu

Avinash Srinivasan
Mathematics, Computer Science, and Statistics
Bloomsburg University
Bloomsburg, PA 17815
Email: avinash@bloomu.edu

***Abstract*—The highly dynamic nature of infrastructureless ad-hoc networks poses new challenges during resource discovery. In this paper, we propose a novel algorithm for resource discovery in mobile ad hoc networks called Efficient Resource-Discovery (ERD). Our primary goal in proposing this novel algorithm is to spread the most relevant resources and queries to the nodes in the network. The proposed algorithm ERD is very efficient in dynamically ranking resources and queries based on their priority, selecting the transmission time, and determining how many resources and queries are to be transmitted. ERD utilizes the network bandwidth in an optimal manner avoiding the spread of redundant data in the network, which otherwise can significantly overload the network with duplicate copies. We compare ERD with periodic flooding and rank based broadcast (RBB) algorithms for mobile ad hoc networks. Results show that ERD outperforms both these algorithms significantly.**

***Index Terms*—** Resource discovery, mobile ad hoc networks (MANET), mobility, simulation

I. Introduction

A mobile ad hoc network (MANET) is a network in which nodes exchange messages with each other through unregulated short-range wireless technologies such as IEEE 802.11 and Bluetooth. Resource discovery, in particular, is critical in the design of a MANET where nodes do not have any prior knowledge of the available resources in the network.

Searching for information is a very challenging task in MANET because of the unpredictable mobility of the nodes. As the nodes travel around the network, they continually establish several ephemeral connections with other peers along the way and exchange the necessary messages with their neighbors. For instance, a driver who is looking for a parking space spreads a request throughout the network until he gets information about an available space. To tackle such a situation in an optimized way, we introduce the Efficient Resource-Discovery (ERD) algorithm for searching resources in the network. In ERD, a moving node disseminates the request for a resource (namely queries) and the resource availability information in the network. The algorithm efficiently spreads the information request in the network without overloading it and searches for a node that has the requested resource.

In resource discovery, there are two approaches for searching the resource: the *push* approach and the *pull* approach [1][5][7]. In the *push* approach, the resources are pushed through the network so that they reach the nodes that have requested the resources. In the *pull* approach, a node floods the network with a resource request. Upon finding the node which has the requested resource, a routing path is created to connect the resource to the request originator. Our algorithm is a hybrid between the *push* and *pull* approaches. In ERD, when a moving node A comes in contact with another node B, they exchange the list of queries and resources that have a high priority in the network. On receiving the list from node A, node B updates its database. Later, when B comes in contact with other nodes, it repeats the process of spreading high-priority data. Consequently, important queries and resources are spread among nodes across the network.

Resource discovery in MANETs is a challenging problem for three primary reasons. First – due to the dynamic nature of the network, there is no proper knowledge about the availability of resources and the locations of nodes. The nodes have to search for this information, which is difficult with nodes moving at different speeds in different directions. Second – message duplication is another major concern while disseminating the messages. A node can discard a duplicate message, but this leads to problems concerning bandwidth usage. The spread of redundant data reduces the efficiency of spreading the messages in the network and performance is negatively affected as a result. In ERD, the spread of duplicate messages is avoided to ensure efficient utilization of network bandwidth. Third – the memory of every node is limited regarding where the messages are stored. As a node moves throughout the network, it continues to receive new messages from other nodes and delete older messages from its memory. There is a high probability that some of the high-priority messages will be lost in this process. The strategy of ERD is to disseminate a list of important queries and resources in the network that keeps the nodes updated on the latest requirements of the network at any given point in time.

The remainder of the paper is organized as follows: Section II lists the related works in this field. Section III explains the model of our algorithm. Section IV presents the simulation results. Finally, Section V concludes the paper with directions for future research.

II. Related Work

Flooding [8] is a common technique that is used to search data in the network. In [3], Yuan and Wu discuss a publish/subscribe protocol that delivers events to the interested nodes in the network. The Rank-Based Broadcast (RBB) technique [2] is another resource discovery technique which ranks the resources in its database and broadcasts the top-ranked resources to its neighbors. Zhao, Ammar and Zegura [4] introduce a technique using a message ferrying approach.

TACO-DTN [11], a content dissemination system, which by virtue is time-aware in terms of subscriptions and events, is appropriate for delay tolerant networks. In this system, there are nodes which act as infostations [10] that are located at specific positions where there are high chances of interested subscribers moving. The self-limiting epidemic forwarding protocol [9] is another dissemination technique which is defined around the local scope of the node. The authors present a scheme where the forwarding of the messages is controlled by manipulating the time to live (TTL). This protocol is broadcast in nature, but there is some control over the spread of the messages in the network.

III. EFFICIENT RESOURCE-DISCOVERY (ERD) ALGORITHM

Our model consists of a fixed number of moving nodes which move in a finite geographic space. These nodes have a specific transmission range and fixed memory size. The nodes move according to the *random way-point* mobility model.

Queries, Resources, and Acknowledgments: Every node periodically exchanges queries, resources, and acknowledgments with its neighbors. A query q_i consists of (1) a unique identifier *i*, (2) the name of the source that initiated the query, (3) the name of the requested resource, (4) the number of hops that it has traveled, (5) the time it was issued t_i at, and (6) a signature. Resource r_j is the data which a node is looking for in the network. The resources disseminated by a node are based on the queries it has.

Hello messages are another prime component of our algorithm. They consist of a node identifier (nid) and an acknowledgment list. The acknowledgment list consists of (1) (nid) of the node that has requested the query q_i , (2) the name of the query q_i and (3) the timer T_i for the acknowledgment. An acknowledgement list is represented as follows: $([n_1, q_1, T_1], [n_2, q_2, T_2], [n_3, q_3, T_3], ...)$, etc. The acknowledgments in this list are discarded once their life time, a fixed time period, expires. Our algorithm executes in four distinct phases: a) Neighbor detection, b) Dynamic ranking of the queries, c) Dynamic ranking of the resources, and d) Dissemination.

A. Neighbor detection

A node periodically exchanges a Hello message with its neighbors. The Hello message contains the node identifier (nid) and the acknowledgment list. At the end of this phase, the node knows its neighbors and has the updated information about the requirements of the neighboring nodes.

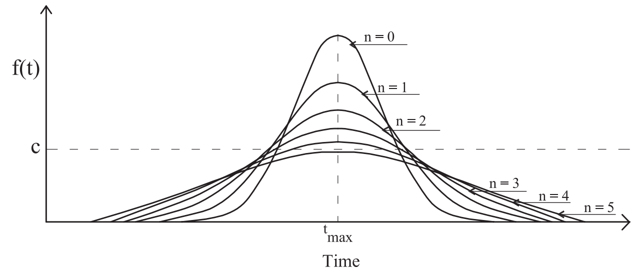


Fig. 1. Ranking function graph using the Gaussian function.

B. Dynamic ranking of the queries

At this stage, the node has queries which are not yet serviced and they need to be spread in the network. The algorithm randomly chooses a small amount of queries which are important at that point in time to avoid spreading duplicate queries in the network and to stay within the limits of the bandwidth constraint. The queries are dynamically ranked and the top-ranked queries are considered for dissemination. The ranking is done such that duplicate data is not spread in the network and significant number of nodes do not have that query. Time t_i is the time at which the query was initiated. This variable is used to determine the rank of the query q_i. t_i also indicates how long the query has been traveling in the network. The greater the value of t_i, the less important the query is because it indicates that the query is relatively old and could have been sent to many nodes. A lower value of t_i indicates that the query is relatively new in the network and should be spread quickly. Dynamic ranking of the queries is done using the following function:

$$f(t) = \frac{1}{\delta\sqrt{2\pi}} e^{-\dfrac{(t-\mu)^2}{2\delta^2}} \quad (1)$$

where, $f(t)$ is the ranking function for queries, and *t* is the time when the query was initiated. δ and μ are constants used to determine the maximum time and threshold time respectively. The Gaussian function is used to find the ranking function of the queries. This ranking function is shown in *Figure 1* and it displays the graph for different values of δ.

In the graph, we vary the values of δ to represent different values n, the number of hops. In the figure, Y-axis has the ranking function and X-axis represents the lifetime of the query. There is a cut-off limit C for the ranking function for every value of n. The cut-off value C is used at the time of disseminating the queries, which we discuss later in the paper. Every graph has a t_{max}, where the ranking function $f(t)$ is maximum for that particular value of n. Ranking of the queries is done with respect to time, in a special manner. According to the above function, when a query is initiated, it has a low level of importance; gradually over time, its importance increases. Consider that query q_1 has $n = 0$ when initiated. The rank of this query is lower as the value of the ranking function $f(t)$ is less when the query is initiated. The queries that are generated before q_1 are of more importance because they need to be disseminated in the network first. Gradually, the importance of q_1 increases with respect to time and its rank rises. Thusly, the chances of q_1 being transmitted are higher when its lifetime is around t_{max}.

When a query reaches t_{max}, its importance starts decreasing because it is assumed that there will be enough copies of that query in the network by this time. Hence, the probability of q_1 being spread starts decreasing as its $f(t)$ starts decreasing. After a threshold lifetime of the query, it is discarded from the local database of the nodes. In *Figure 1*, there is a different graph for different values of n. A query which has traversed through fewer nodes is consider to have higher priority and consequently a higher value of $f(t)$ when compared to the query which has traversed through more nodes. For instance, assume there are two queries with a node- q_1 has $n = 0$ and q_2 has $n = 3$. This indicates that q_2 has traveled in the network more than q_1 and there is higher probability that q_2 has been spread to more nodes. So, while ranking the queries, q_1 should be given a higher priority over q_2. Because of this, we have higher $f(t_{max})$ for a query which has traveled to fewer nodes. After ranking the queries using the ranking function, the queries are once again scanned using the signatures of the queries to re-rank the queries. Every query has a signature containing the list of nodes which have that query.

A query q_1[A, B, C] shows that nodes A, B and C have query q_1 in their local database. Every node has the node ID of its neighbors that are received along with the Hello message. Queries which are already with the neighbors are not transmitted to both preserve precious bandwidth as well as avoid transmission of duplicate copies. This procedure is explained in Figure 2. There are two nodes, A and B, in each other's transmission range, with their own query list in their local database. Suppose node A has to transmit its top 2 queries from q_1, q_2 and q_3, which are shown with their respective signatures in Figure 2. The signature of q_1 shows that nodes A and B have it. If q_1 is transmitted to B, then it will receive another copy of the same query and will discard it. This can be avoided by not sending q_1 to B. q_1 is then pushed to a lower rank in the list instead, and q_2 and q_3 are considered to be transmitted to B. At the end of this phase, the queries are ranked using the ranking function and the signatures of the queries, as discussed previously.

C. Dynamic ranking of the resources

Resources are the data which a node in the network has requested. These resources are ranked using the queries which were ranked in the previous phase; the ranking itself is done as follows:

$$f(d_i) = \frac{\#n(d_i)}{\#(d_i)} \quad (2)$$

where, d_i is the resource or data (resource information), $\#n(d_i)$ is the number of nodes requesting d_i, and $\#(d_i)$ is the number of nodes having the resource d_i. $\#n(d_i)$ can be determined from the queries which are ranked in the previous step. Every resource has a signature attached to it which indicates how many nodes have that particular resource. For instance, r_1[A, D, E] is a resource which is held by nodes A, D and E. $\#(d_i)$ can be determined from the signature of the resource.

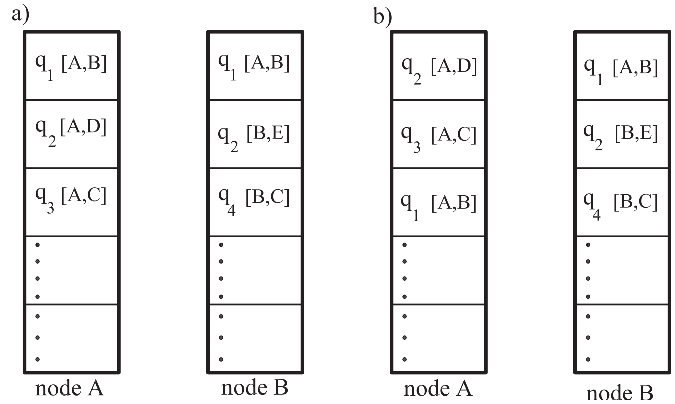


Fig. 2. State of the queries with their signatures (a) before the exchange of queries (b) after the exchange of queries.

D. Dissemination

The objective of our algorithm is to spread the resources based on the queries that have high priority in the network at that point in time. In the previous phases, we determined the high priority queries and resources, which are dynamically ranked by the nodes. The top-ranked queries and resources need to be transmitted to their direct neighbors. A major concern while disseminating the queries and resources is the bandwidth. As there is limited bandwidth, it has to be distributed between the resources and queries efficiently. ERD makes the decision of allocating bandwidth for the transmission of queries and resources dynamically. The value of $f(t)$ [Eq. 1] and $f(d_i)$ [Eq. 2] of the query and resource respectively, are used in allocating the bandwidth. The list of queries whose $f(t)$ is greater than cut-off limit C, as in Figure 1, are considered for dissemination. In the remainder of this paper, we will refer to this cut-off C as $bandwidth_{query}$. The queries are then sorted based on $f(t)$. Similarly, for the resources, $bandwidth_{resource}$ is the threshold value for the resources. Resources with $f(d_i)$ greater than $bandwidth_{resource}$ are considered ready for transmission. We assume that queries and resources above these cut-off values are new in the network and need to be transmitted sooner.

Before transmission, however, the allocated bandwidth is consider to decide how many queries and resources can be transmitted. Suppose the allocated bandwidth allows the transmission of 6 queries and 9 resources which sums up to 15 data packets that need to be transmitted. Assume, from the sorted lists, there are only 4 queries whose $f(t)$ values are above the $bandwidth_{query}$ and there are 13 resources with their $f(d_i)$ values above the $bandwidth_{resource}$. Therefore, from the sorted list of the queries, we can see that there are only 4 queries which are new. Out of the 13 resources which qualify for transmission, only top 11 resources are selected. Finally,

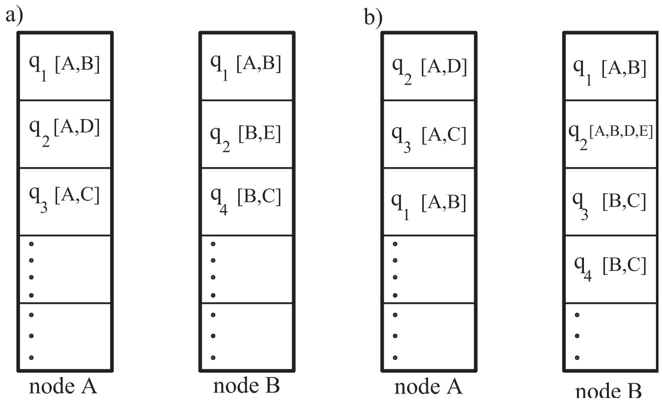


Fig. 3. State of the queries with their signatures (a) before exchange of the queries (b) merging of queries after exchange.

the node transmits 4 queries and 11 resources, which sums up to 15 data packets. This way the bandwidth will be distributed dynamically between the queries and resources, depending on the need. The bandwidth is shared in an optimized way and important data is spread in the network.

Once the queries are transmitted to the neighbors, merging the signatures of the queries is another important feature of our ERD algorithm. Nodes are not aware of which queries are stored in other nodes. For instance, in Figure 3, the signature of q_2 in node A shows that q_2 is in nodes A and D, whereas node B shows that q_2 is in nodes B and E. From this, it can be concluded that q_2 is in nodes A, B, D and E. However, A and B are not aware of this complete information. To solve this problem, when q_2 is received by node B, the signatures of the query are merged. In Figure 3, the signature merge of q_2 from nodes A and B is [A, B, D, E]. For the future spread of q_2, the merged signature will be considered that will reduce the spread of redundant data in the network.

In the proposed ERD, we assume that nodes have a limited memory and that nodes cannot afford to lose important data that is important in the network at that point in time. Therefore, when the memory limit is reached, queries and resources of lower ranks from the ranked list are discarded to make space for new data. In this way, ERD ensures that important data is not deleted from the local database of the nodes.

IV. Simulation & Results

A. Simulation Environment

In our simulation, the mobile nodes move in the network and spread the resources and queries. We adopt the *random way-point* mobility model in our simulation study. Each node moves in a 0.5×0.5 mile square area with a random speed and in a random direction. Each node has a fixed lifetime after which it perishes from the network. The lifetime of each node is randomly chosen from [100 - 600] time units, before the start of the algorithm. The whole simulation runs for 1,000 time units. Each node can hold up to 45 resources and these resources are essentially numbers from [0, 45]. Similarly, each node can hold up to 40 queries selected from [0, 40]. For our

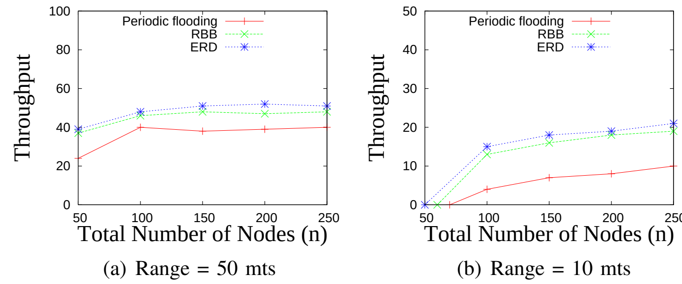


Fig. 4. Throughput vs. Density of the network.

simulation, $t_{max} = 60$ seconds, where $f(t)$ is maximum i.e. 0.1. We consider the cut-off value C to be equal to 0.06.

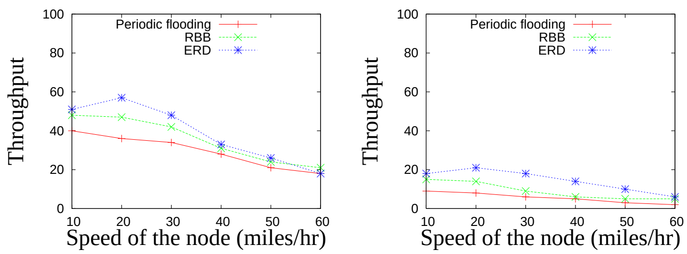
The performance measure function we consider in our simulation is the throughput, which is the average number of matching resource messages received by a moving object.

B. Simulation Results

In periodic flooding, a node does not dynamically select the flooding period based on the network condition. Therefore, the flooding period can either be too long, which can leave the network in an idle state, or too short in which case the traffic in the network will be very high. In the RBB technique, the resources are ranked and the top-ranked resources are broadcast. During every broadcast, only the native query of the node is transmitted and other queries in the local database of the node are not spread. In RBB, the native queries are transferred only when nodes come in direct contact with each other. This reduces the spread of the queries. In our model, both queries and resources are ranked prior to transmission. The transmission period is decided on the basis of the new neighbors. The dynamic ranking of the queries and resources on the basis of the current demand in the network spreads the most promising data in the network.

The results are shown in Figure 4(a), where the number of nodes is varied and the throughput is noted for each of the three algorithms- periodic flooding, ERD, RBB. It can be observed that as the number of nodes increases, the throughput also increases. In the graph, ERD performs better than both periodic flooding and RBB algorithms. The performance of periodic flooding is the worst because redundant resources are broadcast in the network. Hence, the throughput is affected since nodes don't receive their requested resource. The performance of RBB is better than periodic flooding, because only the native query is being transmitted along with the top-ranked resources. In Figure 4(b), we compare their performance by reducing the transmission range of nodes to 10 meters. With a smaller transmission range, the node comes in contact with fewer neighboring nodes thereby resulting in less resources and queries being spread. Consequently, there is a decline in the performance when the transmission range is reduced.

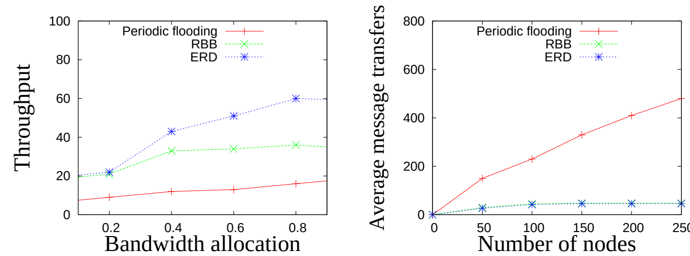
Figures 5(a) and 5(b) show the performance of each algorithm as the node speed is varied. The transmission range of the node is fixed at 50 meters in Figure 5(a). According to the



(a) Range = 50 mts

(b) Range = 10 mts

Fig. 5. Throughput vs. Speed of the nodes.



(a) Bandwidth

(b) Message transfers

Fig. 6. Studying the optimal behavior of ERD.

results, as the speed increases, the performance declines as the time spent by a node in the transmission range of another node is reduced. As the speed decreases, the nodes remain in each others transmission range for a longer time period and hence, can exchange the messages. Therefore, the performance is better initially but when the speed is increased the time spent by a node in the transmission range of another node is less. In Figure 5(b), we reduce the transmission range to 10 meters and we can see that ERD gives better results. However, the overall performance declines because the node comes across relatively fewer neighbors during movement in the network when the transmission range is shorter.

In Figure 6(a), we vary the bandwidth allocated to the nodes and check the performance of the algorithms. The greater the number of messages transmitted, the higher are the chances that the node gets its required resource. In RBB, as bandwidth increases so does the broadcast of the number of resources in the network. ERD outperforms RBB because it also spreads the queries along with its native query. Figure 6(b) shows that ERD drastically decreases the number of query messages transferred during a search. The ERD manages to keep the message transfers almost at a fixed level as the decision is taken dynamically before every transmission. Thus, by using the network bandwidth efficiently, ERD is able to scale well as network size increases.

C. Simulation summary

From the above simulation results, it can be observed that the performance of our ERD algorithm is better than the flooding algorithm and the RBB technique. The dynamic ranking of the queries and resources by ERD gives better throughput. The spread of top queries helps to inform the requirements of other nodes in the network. ERD utilizes the bandwidth in an optimized manner, which results in better performance. The nodes will dynamically decide how many queries and resources need to be transmitted based on their importance. This results in spreading only the important data in the network. The performance of ERD is better than both periodic flooding and the RBB algorithm as it transmits more important data based on the overall requirement of the network. Hence, ERD gives optimal results in dense networks and utilizes the bandwidth both effectively and efficiently.

V. Conclusions

In this paper, we proposed ERD, an algorithm which optimally disseminates the queries and resources in the network without spreading redundant data. In ERD, nodes dynamically select the queries and resources, which are transmitted to their neighbors. This decision is made based on novel techniques for prioritizing and disseminating mobile information, as well as novel techniques for effective bandwidth utilization in mobile data dissemination. The bandwidth is utilized in an optimized way by dynamically distributing it among queries and resources. Important queries and resources are given more preference and the less important ones are discarded from the memory of a node. We compared the ERD algorithm to periodic flooding and RBB algorithms; the results show that ERD significantly outperforms the other two algorithms. These results show a lot of promise for the Efficient resource discovery algorithm and we plan to further evaluate this algorithm under various scenarios of mobility patterns and resource distribution.

References

[1] Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile environment. Proc. of ACM MoBiDE, 2001.

[2] O. Wolfson, B. Xu, H. Yin and H. Cao. Search-and-discover in mobile P2P network databases, Proc. of IEEE ICDCS, 2006.

[3] Q. Yuan and J. Wu. DRIP: A dynamic voronoi regions-based publish/subscribe protocol in mobile networks. Proc. of IEEE INFOCOM, 2008

[4] W. Zhao, M. Ammar and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. Proc. of ACM MOBIHOC, 2004.

[5] Y. Huang and H. Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. Proc. of MDM, 2003.

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

[7] R. Zhang and Y. Hu. HYPER: a hybrid approach to efficient content-based publish/subscribe. Proc. of IEEE ICDCS, 2005.

[8] R. Oliveira, L. Bernardo, and P. Pinto. Flooding techniques for resource discovery on high mobility MANETs. Proc. of IWWAN, 2005.

[9] A. E. Fawal, J. V. Le Boudec and K. Salamatian. Self-limiting epidemic forwarding. Technical report Ref. LCA-REPORT-2006-126, 2006.

[10] R. H. Frenkiel, B. R. Badrinath, J. Borres and R. D. Yates. The infostations challenge: balancing cost and ubiquity in delivering wireless data. IEEE Wireless Communications, 7(2):6671, 2000.

[11] G. Sollazzo, M. Musolesi and C. Mascolo. TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. Proc. of ACM MobiOpp, 2007.

[12] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. IEEE Trans. on Communications, 1984.