HRAN - a scalable routing protocol for multihop wireless networks using bloom filters

João Trindade[ψ], Teresa Vazão[Ψ]
INESC-ID / Department of Computer Science and Engineering, Instituto Superior Técnico
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal;
Email: [ψ] jtrindade@tagus.inesc-id.pt [Ψ] tvazao@tagus.inesc-id.pt

Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, Portugal

Abstract. We propose a novel routing protocol for large scale Mobile Adhoc Networks called HRAN (Heat Route for Ad-hoc Networks). The protocol is able to scale to large composed by many nodes, requires few resources from devices and lowers the global number of control message imposed by the protocol in discovering routes. HRAN maps the paradigm of heat trails in a physical environment to the network topology. In our protocol each node emits a certain quantity of heat and as the node moves a heat trail is formed. After a node leaves a location the heat from surrounding nodes slowly dissipates. This heat information is then used to guide routing queries from the source to the destination. All heat information is represented through the use of bloom filters. We compare our proposed solution with other routing protocols for MANETS and the obtained results show that for large scale networks HRAN reduces the overhead of control messages.

Keywords: Routing Protocol, Large Scale MANETs, Bloom Filter

1 Introduction

The proliferation of wireless devices in today's world has lead to emergence of Mobile Ad-hoc NETworks also denominated as MANETs. These type of networks are becoming ubiquitous in many human activities, with an increasing number of participants in a wide range of application environments.

MANETs have specific characteristics which are not found in other types of networks, such as the limited lifetime of the node's batteries, the unpredictable signal conditions and node's mobility that lead to frequent topology variation. For these reasons they pose some novel research problems, being routing one of the most challenging as traditional protocols are not able to cope with node mobility. So, as described in various surveys [1] [2] [3], there is a wide variety of routing protocols which were developed to respond to these problems. Some of them, as the Destination-Sequence Distance Vector (DSDV) or the Optimized Link State Routing protocol (OLSR) are a direct evolution of routing protocols

that were designed for wired networks, whilst others, as the Zone Routing Protocol (ZRP) or the Fisheye Routing Protocols (FSR), were designed using new concepts specifically suited for MANET environments.

Nevertheless, as the authors of [4] [5] have shown none of them are able to provide consistent good results in different scenarios. In fact, in [6] and [7] the authors have observed that the efficiency and effectiveness of non-geographical routing protocols does not scale well to MANET composed by hundred of nodes. In most of them, the number of controls messages grows exponentially with the number of nodes, increasing the overhead and decreasing the overall network performance. Geographical routing protocols [8] have the drawback of requiring a positioning sensor, normally a GPS device, for executing their path discovery processes. In many cases the use of such equipment is not adequate as it requires a significant amount of battery resources and increases the monetary value of the nodes. Moreover, route discovery is performed by the use of an additional component, the location-service, which can introduce a significant amount of overhead.

In this work we propose a new routing protocol named Heat Routing for Adhoc Networks (HRAN) suited for MANET scenarios. HRAN is scalable to large networks, in which the nodes have limited resources (memory, processing power and bandwidth) and do not requires the use of GPS devices. HRAN relies on the use of bloom filters to store and spread topology information in an efficient way. In the HRAN protocol nodes do not explicitly rebroadcast topology information messages coming from other nodes. Instead, this information is merged by each node with its own topology data and only the result is shared. To reduce the number of messages sent by a node, whenever a route to a destination is requested, HRAN does not flood the network but uses the information present in the node's bloom filters to discover what nodes are potentially useful for the route. This process effectively guides the route query to the destination and avoids unnecessary message transmissions, saving battery power and decreasing the overhead imposed by the protocol over the network.

Previous work of other authors proposed the use of bloom filter in routing protocols [9]. In the Gradient-ascending routing via footprints (GRASP) bloom filters are used to store information of wether a node of a wireless sensor networks belongs to route or not, but it lacks for an efficient mobility support. Other proposal is the Table Attenuation Routing Protocol (TARP) [11], in which bloom filter are use to implement a proactive routing strategy. TARP does not scale to networks with high node density or mobility.

The remainder of this article is structured as follows: Section 2 presents the background and related work, by introducing the concept of bloom filters data structure which will be used by HRAN to represent and exchange topology information. Section 3 describes our new routing protocol and introduces the concept of *Time Aware Bloom* filter. Details of the protocol's three operating phases are also included in this section. Section 4 presents the assessment of our proposal and the comparison against other known routing protocols for MANETs. The

simulation setup is described and the results are depicted and analysed. Finally, section 5 draws the conclusions and lists future work.

2 Bloom Filters

2.1 Concept of Bloom Filter

A bloom filter [12] is a probabilistic data structure that is used to test if an element is a member of a set. It is similar to a standard hash table, except that it is more space efficient and does not store the actual value, rather it maps hashed keys related to the object into a bit array.

A bloom filter returns whether *true* or *false* when asked if an object A is a subset of B ($A \subseteq B$). A return of the value *false* is guaranteed to be accurate, but a return of the value *true* has a probability of being a false positive.

In a more detailed description, a bloom filter representing the set $S = \{x_1, c_2, \cdots, x_n\}$, where n is the number of elements, is an array of m bits, initially all set to 0. $H = \{h_1, \cdots, h_k\}$ represents the independent hash functions used by the bloom filter, with each hash function having as output range values $\{1, \cdots, m\}$. Every element $x \in S$ is represented by having the bit $h_i(x) = 1$ for values of i ranging from 1 to k.

To check if an item y is present in set S then all $h_i(y)$ should have the value 1. If this is not the case then it is possible to assure that $y \notin S$. If the contrary happens, $h_i(y) = 1$ for all values of $\{i : 1 < i < k\}$, then either it is a false positive or it is a true positive. A false positive is a situation where the element is not on the set although the bloom filter indicates that it is. A true positive always informs that the element is present in the set.

In order to exemplify an instance of this data structure, Figure 1 presents a bloom filter with $m = 8$ and the set of $k = 3$ hash functions being $H1$, $H2$ and $H3$. In the example we insert two different objects, obj_A and obj_B. In the case of obj_A the result of hashing its identifier is 7 for $H1$, 3 for $H2$ and 2 for $H3$, and thus insertion of obj_A in the bloom filter is achieved by setting the bits 7, 3 and 2. To query if obj_A is present in the bloom filter one just has to check if bits 7, 3, and 2 are all set. For obj_B the result is 5 for $H1$, 0 for $H2$ and 3 for $H3$. To insert obj_B the same operation is performed, but in this case there was a bit collision, as bit 3 was already set by the insertion of obj_A. If we lookup an hypothetical obj_X which has as a result of the group of hashes the values 7, 5 and 4, then the return of the lookup would be true although obj_X was not present in the filter. This is known as a false positive and is caused by the insertion of the previous objects.

3 Heat Routing Protocol for Ad-hoc Networks

3.1 Time Aware Bloom Filter

The HRAN routing protocol requires the extension of the concept of bloom filter in order to check if a set of bits present in the structure was updated

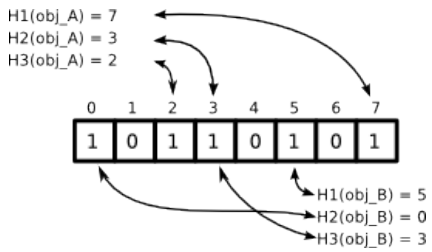


Fig. 1: A Bloom Filter Example

recently. In order to achieve this functionality, we propose a new type of bloom filter, the *Time Aware Bloom* (TAB) filter. This new structure maintains the property which states that false positives may occur but enables the removal of items which have not been updated during a predefined time period.

The TAB filter is a tuple of two bloom filters, sharing the same size n and the group of k hash functions. These structures are named the *sticky* and *plain* filter and are depicted in Figure 2. The *sticky* filter is used to identify whether a bit was updated in the last δ time interval or not, whilst the *plain* bloom filter contains the bits present in the current plain filter plus the bits set to one in the previous δ time interval.

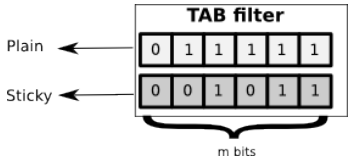


Fig. 2: Time Aware Bloom Filter

To add a new element in the TAB filter, the algorithm performs the default bloom filter operation on the *plain* and *sticky* filters. This consists on hashing the inserted element with the k hash functions group in order to get the k corresponding array positions. The bit array positions are then set to 1 on both filters.

To search if an element is in the TAB filter, a query is made only on the *plain* bloom filter. Once again this operation is similar to the default bloom filter lookup function. The element value is hashed by the group of hash functions and the respective bits are checked. If all respective bits are set to value 1 the TAB filter returns a positive value indicating the element is present in the structure.

Finally, at regular δ time intervals, the *plain* filter is erased and its content is then overwritten by the *sticky* bloom filter bit array. After this operation is

completed all the *sticky* filter positions are reset to 0 value in order to indicate a new δ time interval.

3.2 HRAN Protocol Description

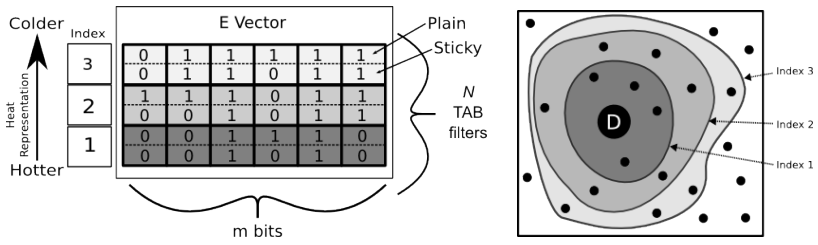
Conceptually, the HRAN protocol execution is divided into three stages, each one serving a different purpose. This first stage is named heat overlay construction and consists in each node spreading heat information to neighbour nodes. Closer nodes are represented using a stronger heat index than nodes which are farther away. When a node spreads its heat it also spreads the heat it had received from neighbour nodes. This mechanism enables the diffusion of topology information using few control messages. The second stage is named route discovery and is reactive part of the protocol which is activated whenever a node wishes to establish a new route to a destination. The route request message travels the network being guided, where possible by the previously constructed heat overlay. When it does not find any heat information regarding the destination, it acts as a random walk query. Random walks are very easy to implement, but usually lead to non-optimal paths. In order to solve this problem HRAN uses an additional mechanism which shortens unnecessary long routes iteratively. The last stage is designated as route maintenance. This stage serves to the repair broken routes with little overhead, iteratively optimizes active routes in terms of hop numbers, and finally helps to construct heat tunnels useful for guiding future queries which share the same destination.

Overlay Construction During the overlay construction stage each node spreads its topology information to the neighbour nodes using a pro-active approach. Each node stores the knowledge it has of the network topology through an array of N TAB filters which is designated as E vector. N is a static configuration parameter that represents the level of heat gradients of the heat overlay. Node ids present in E vector index position's closer to 1 are "hotter" than those closer to index N.

In Figure 3a we present an E vector capable of representing three heat gradient levels ($N = 3$), having each tab filter two bloom filter of 8 bits size each. Figure 3b shows the heat overlay of node D on a network where nodes in darker colours layers have node D's identifier in lower index positions than nodes present in brighter layers. Nodes not inside a layer do not have node's D identifier in any layer.

The construction of the heat overlay, mapping the network topology, is created by each node sending HELLO messages at regular time intervals. Each one of these messages contains a vector with $N-1$ bloom filters because a node does not transmit the bloom filter for colder gradient it knows (index N). The bloom filters contained in the HELLO message are a copy of the *plain* filters present at the TAB filters at the sending node's E vector from position 1 to position $N-1$.

When a node receives a HELLO message a bitwise OR operation between the bloom filters present in the messages and its E vector is performed. The iterator



(a) E vector (b) Heat Overlay

Fig. 3: Overlay construction stage

in the E vector starts at position 2 while the index that iterates over the array in the message starts at position 1. This operation, presented on Figure 4, is done in order to create heat gradients where information of nodes which are further away is inserted in the colder index positions of the E vector.

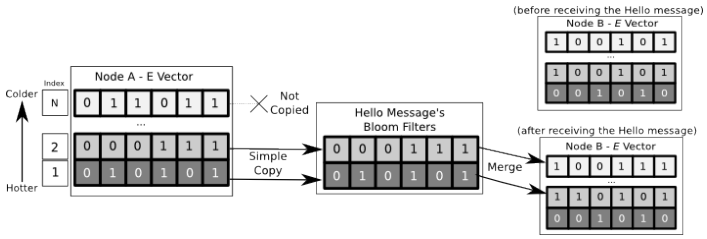


Fig. 4: Hello message exchange between neighbour nodes (for simplicity only plain bloom filters are shown)

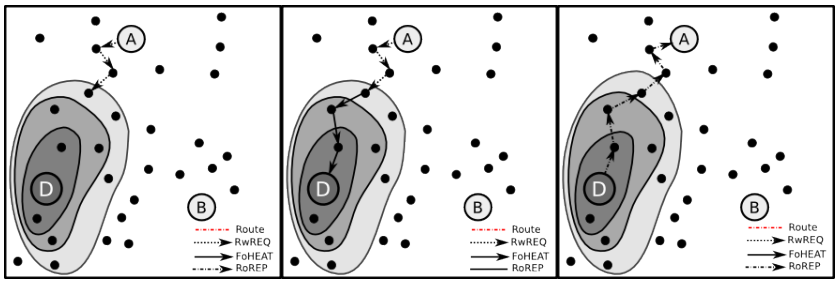
Route Discovery The second stage of the HRAN routing protocol is named route discovery. This stage is responsible for discovering a valid route from the source to the desired destination.

When a source node requests a route to an unknown destination, it initiates a predetermined number of *random walk request* queries (RwREQ) which travels through the network as is shown on Figure 5a. Upon receiving a RwREQ, the node analyses its E vector and checks if it contains the destination node's identification in any one of the *layers*. If this is not the case, the random walk continues to a next neighbour node chosen randomly. If a match is found, the random walk ends and a directed walk is started by sending a *follow heat* (FoHEAT) message with the index of the current heat level. FoHEAT messages, as shown on Figure 5b, are only forwarded by nodes that have the destination

node identifier in a E vector positions which is hotter than the current heat level passed by the FoHEAT message.

The route discovery mechanism enables the query to rapidly reach the destination with few messages exchanged because only nodes with information of the destination node's heat rebroadcast the query. Nodes do not send twice the same message, as each FoHEAT message contains a unique identifier composed by the route requester id and a sequence number. This way a FoHEAT message is not transmitted if a node has already delivered a message with the same identifier.

Once the FoHEAT query reaches the destination node, the route is given by the list of nodes present in the path log of the message. At this moment the protocols enters in the *go back mode* and the destination sends a *route reply* (RoREP) to the source using the inverted route discovered. This is shown on figure 5c. If the source node, after sending its RwREQ message receives no answer for a predefined amount of time, it falls backs to a reactive source routing protocol (like DSR). This backup procedure is performed in order to ensure the creation of heat tunnels which following queries will be able to use for reaching the same destination. Heat tunnel formation will be explained in the next stage.



(a) Random Walk Mode (b) Directed Walk Mode (c) Go Back Mode

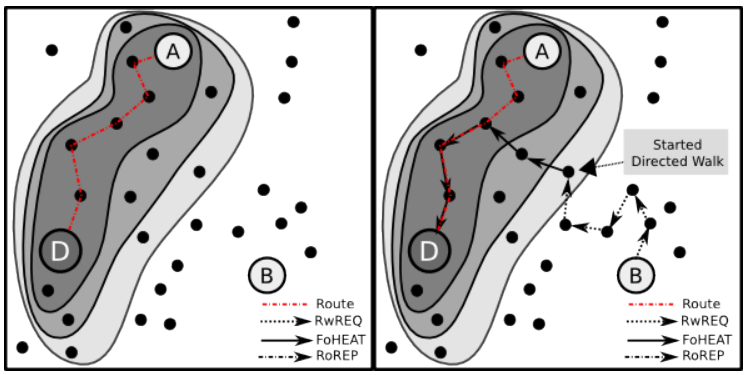
Fig. 5: Route Discovery Process

Route Maintenance Route maintenance stage includes three different tasks: heat tunnel creation, route repair and route improvement. The first one aims at creating and maintaining an heat tunnel when a route is being used to transfer data packets. This process helps future route discoveries to quickly find the destination node and supports the route repair stage. Finally, route improvement allows the HRAN protocol to iteratively improve a route as it is being used.

Heat tunnel creation When a data packet is routed by an intermediate node, a heat tunnel is created or updated with the identifier of the destination. This is accomplished by inserting the destination nodes unique identifier in the respective E vector of the intermediate nodes at position number 0. This operation

maintains heat tunnels information on the TAB filters for routes which are actively being used. Inactive routes dissipate with time as their respective heat tunnels information is not updated on the TAB filters.

An example of an heat tunnel formed around the route from node A to node D is shown on Figure 6a, in which nodes in the different shades of grey symbolise the respective amount of heat information regarding the destination. Nodes inside the darker colour layer have the destination identifier in its hotter E vector index position than those nodes on the more light grey layers. Nodes which are not inside a grey layer do not belong to the heat tunnel as they do not have the destination identifier inside their E vectors. Heat tunnels purpose is to help future route requests to quickly find a route to the destination, as is shown on Figure 6b.



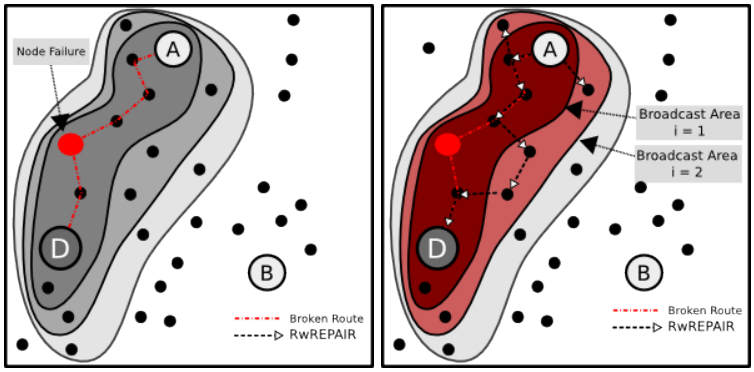
(a) Heat tunel for node D (b) New route discovery

Fig. 6: Heat tunnel creation

Route Repair When a route failure is detected by the origin endpoint of a communication, the previously created heat tunnel is used to help discover a new valid route. To start this process a *route repair* (RoREPAIR) message with the value of $i = 2$ is broadcasted to all the source node's neighbours. If node receives this message and has the destination indentifier in one of its E vector index positions which are lower or equal to i then it rebroadcasts the RoREPAIR message. When the destination receives a RoREPAIR message, it sends a RoREP message to the source using the inverted route discovered. If the source does not receive a RoREP message during a predetermined time period, it performs the same operation, increasing the value of i. This is repeated until i value surpasses the value of E array positions ($i > N - 1$), at which time the source node considers the route to be irreparable and falls back to the normal route discovery mechanism described earlier in order to find the new route. Through

this procedure, route repairs are achieved by incrementally flooding the various levels of the heat tunnel.

Figure 7 shows the route repair procedure on a network, starting at the route fail instant and ending when the new route is discovered.



(a) Intermediate node failure (b) Route repair message with i=2;

Fig. 7: Route Repair process

Route Improvement Due to the random process of route discovery, in the HRAN protocol it is not guaranteed that discovered routes are the best paths in terms of number of hops. To improve the routes HRAN uses a process which enables the iterative and gradual approximation of the discovered routes to the optimal ones. Also, as nodes move there is the possibility that a potential better route between the source and the destination is created.

HRAN's route improvement process uses the heat tunnel described previously to discover new and faster routes. When a route is being used, HRAN keeps a counter of the number of packets sent by the source node and when a threshold value is reached the source node sends a *route improvement* (RoIMP) message containing the destination ID. Each node which receives this message and has in its E vector the destination, rebroadcasts it. This process happens recursively until the destination node receives the RoIMP message. When this happens a RoREP message is transmitted through the inverse path. The source node then uses the path reported by RoREP as the new route. By using this mechanism RoIMP messages are only broadcasted inside the heat tunnel. Route improvement can therefore occur using a limited number of transmitted messages and only impacting nodes which are close to the path between the source and the destination. Similarly to FoHEAT messages, RoIMP messages contain an unique identifier in order to prevent messages from being sent twice by the same node.

4 Simulation Results

We used the Qualnet version 5.0 software, due to its ability of simulating MANETs and the relatively large number of routing protocols it includes.

HRAN was implemented in Qualnet in order to be compared with some of the most representative and studied MANET's routing protocols. Several parameters of HRAN protocol were tuned and configured as follows: each bloom filter was configured with a size of $m = 128bits$ and three hashing functions; and the E vector had $N = 3$ heat gradients. HELLO packets were generated each 8s, with a jitter of 800ms to reduce HELLO message collisions.

4.1 Routing protocol comparison

To assess the scalability of HRAN protocol we used small, medium and large networks, in which we vary the number of nodes. The coverage area for all maps was $400m^2$ with nodes distributed at random places. The small size network had 50 nodes, the medium network had 100 nodes and the large network had 200 nodes. In all cases, nodes are randomly disposed on a square map and pairs of source/destination were randomly selected, representing 30% of the network nodes.

CBR/UDP traffic was used in all the simulations, generating data packets of 512 Bytes, at a rate of 2 packets/s. As we want to assess the behaviour of the routing protocols, the MANET must operates in proper conditions without collisions that lead to unwanted packet losses. Thus, a small data generation packet was used.

Figure 8 presents the amount of messages of each type sent by the HRAN protocol when new routes are established. As it is possible to notice the proactive facet, represented by the number of sent HELLO messages, is independent of the network size. The remaining messages types, which are sent only when a new route is requested, seize the heat overlay which was previously spread. This algorithm allows a small number of sent messages even as network size increases.

In the following tests we compared HRAN with three other routing protocols: the reactive protocol AODV, the proactive routing protocol OLSR; and the hybrid protocol ZRP. The overhead was measured as depicted in Eq 1. The results are depicted on Figure 9a.

$$r = \frac{\text{control packets}}{\text{data packets}} \tag{1}$$

As state in the figure, all protocol present a significant overhead, as the amount of data traffic is significantly low (2 packets/s). HRAN outperforms both OLSR and ZRP in all cases and AODV for the large network case. Thus, the reactive nature of this protocol does not seems to be good enough to maintain the overhead results when the network size increases. This results are caused by the small size and reduced number of messages used by HRAN to spread routing information.

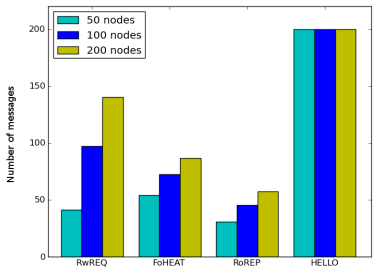
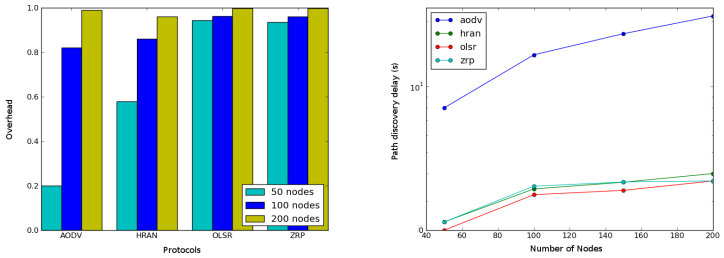


Fig. 8: HRAN messages

Finally, Figure 9b shows the performance of four simulated protocols in terms of the time required to discover new routes to destinations. When analysing this metric we notice that the reactive nature of AODV does not allow it to quickly find new paths. By the contrary, HRAN is able to provide new routes in lower times due to the use of bloom filter to store topology information which effectively guides routing queries to reach the destination.



(a) Protocol Overhead (b) Total Route Delay

Fig. 9: Protocols Comparison

5 Conclusion

This paper proposes a novel hybrid routing protocol specifically designed for large scale MANET that relies on the concept of heat tunnels to discover and maintain active routes. The HRAN protocol spreads topology information across the various nodes by using bloom filters. These special structures are small in size and capable of storing a large number of binary information. As the proposed

algorithm requires the removal of values from these structures if they are not updated during a certain time, the *time aware bloom* (TAB) filter structure was developed.

The simulation results demonstrated a case scenario where the HRAN protocol considerably reduced the number of routing messages necessary to find a route, whilst keeping the time to discover new routes acceptable and the number of hops almost optimum. As future work we intend to dynamically adjust HRAN parameters and to improve the random walk performance without introducing significant complexity in the protocol. Also new mechanisms which enable a trade-off between the number transmitted messages inside a heat tunnel and route discovery effectiveness will be studied.

References

1. Jayakumar, G., Gopinath, G.: Ad hoc mobile wireless networks routing protocols - a review. Journal of Computer Science **3** (2007) 574–582
2. Junhai, L., Danxia, Y., Liu, X., Mingyu, F.: A survey of multicast routing protocols for mobile ad-hoc networks. Communications Surveys & Tutorials, IEEE **11** (2009) 78–91
3. Mauve, M., Widmer, J., Hartenstein, H.: A survey on position-based routing in mobile ad-hoc networks. IEEE Network **15** (2001) 30–39
4. wu Chin, K., Judge, J., Williams, A., Kermode, R.: Implementation experience with manet routing protocols. ACM SIGCOMM Computer Communications Review **32** (2002) 49–59
5. Mohseni, S., Hassan, R., Patel, A., Razali, R.: Comparative review study of reactive and proactive routing protocols in manets. In: Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on. (2010) 304 –309
6. Wang, Y., Dong, L., Liang, T., Yang, X., Zhang, D.: Cluster based location-aided routing protocol for large scale mobile ad hoc networks. IEICE Transactions **92-D** (2009) 1103–1124
7. Boukerche, A.: Performance evaluation of routing protocols for ad hoc wireless networks. Mob. Netw. Appl. **9** (2004) 333–342
8. Karp, B., Kung, H.: Gpsr: Greedy perimeter stateless routing for wireless networks. In: Annual International Conference on Mobile Computing and Networking (Mobicom), ACM (2000) 243–254
9. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. Internet Mathematics **1** (2005) 485–509
10. Lim, J.J., Shin, K.G.: Gradient-ascending routing via footprints in wireless sensor networks. Real-Time Systems Symposium, IEEE International **0** (2005) 298–307
11. Gilbert, R., Johnson, K., Wu, S., Zhao, B.Y., Zheng, H.: Location independent compact routing for wireless networks. In: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking. MobiShare '06, New York, NY, USA, ACM (2006) 57–59
12. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13** (1970) 422–426