A Scalable QoS Routing Scheme Using A Mobility Prediction-Based Clustering Approach for Large-Scale Ad Hoc Networks

S. Sivavakeesar, G.Pavlou

Center for Communication Systems Research
University of Surrey
Guildford GU2 7XH
United Kingdom

{S.Sivavakeesar, G.Pavlou}@eim.surrey.ac.uk

Abstract
In this paper we present a framework for dynamically organizing mobile nodes (MNs) in large-scale mobile ad hoc networks (MANETs) in order to support Quality of Service (QoS) routing. We believe that in MANETs it is essential for a MN to have an approximate mobility pattern of its neighbors in order to make QoS-enabled forwarding decisions in a scalable manner. In our scheme, each MN is expected to predict its own mobility pattern, and this information is disseminated to its neighbors using a scalable clustering algorithm. This paper presents a scalable way to predict mobility, and thus availability, of MNs, which is achieved with the introduction of geographically-oriented *virtual clusters*. We name our model the *(p,t,d)-model* that facilitates the formation of stable clusters and enables better QoS routing.

1 Introduction

It is expected that future wireless networks will evolve towards non-authority based, self-organized, large-scale MANETs, which have a significant impact on future communication models and m-business. In this work, we envisage a large-scale deployment of a multihop MANET, which is similar but complementary to mobile telephony systems (and we do not necessarily consider the creation of a spontaneous network that comes into life only for a short period of time). As such, the network model considered in this work is a longer-term ad hoc network – similar to that assumed in the 'Terminodes' project [1]. Given that such MANETs may be composed of a large number of MNs widely spread geographically, a hierarchical instead of a flat approach will scale better [3]. We, hence, adopt a hierarchical clustering approach, which is fully distributed and dynamic in nature. The primary step in clustering is the election of cluster heads (CHs) and the formation of clusters around them. The clustering technique brings in a number of benefits as stated in [3][4]. Location information may be obtained using the Global Positioning System (GPS), or a self-positioning algorithm as specified in [1]. Our mobility prediction approach is derived from data compression techniques that are both theoretically optimal and good in practice. We name our model the *(p, t, d)-model* that facilitates the formation of stable clusters and provides a framework for better and more efficient QoS routing.

2 Previous Work and Our Motivation

While many clustering techniques with CH selection have been proposed in the literature, almost none of them consider node mobility as a criterion in the clustering process effectively [2][3][4][5]. As a result, they fail to guarantee a stable cluster formation. In a MANET that uses cluster-based services, network performance metrics such as throughput and delay are tightly coupled with the frequency of cluster reorganization. Therefore, stable cluster formation is essential for better QoS. The most popular clustering approaches in the literature are the lowest identifier (Lowest-ID) and maximum-connectivity [4][5]. But these two, along with others, do not provide a quantitative measure of cluster stability. In the former, a highly mobile lowest ID CH will cause severe re-clustering; in addition, if this CH moves into another region it may unnecessarily replace an existing CH, causing transient instability. In the latter, depending on MN movement and traffic characteristics, the criterion values used in the election process can keep on varying, and hence may also result in instability. This is also the case in the Lowest Distance Value (LDV) and the Highest In-Cluster Traffic (ICT) approaches [2].

Our approach is motivated by the fact that in MANETs link bandwidth and MN transmission power are scarce, and any effective solution should take this into account and try to conserve them [3]. However, effective routing requires each MN to have up-to-date information on network topology, while keeping the control or signaling overhead as low as possible. In order to achieve a compromise between these two, accurate intelligent prediction of future state is necessary for the network control algorithms to keep pace with rapid and frequent state changes [6]. Hence, we propose a scalable mobility prediction scheme based on the accumulated past behavior history of a specific MN. In MANETs, the uncertainty about network topology arises mainly due to MN movement [3]. If, however, the movement pattern of the MNs is almost deterministic or roughly repeatable, then the lifetime of a link can be determined with relative precision. In typical mobile networks and in the type of longer-term MANET that we consider in this work, MNs exhibit some degree of regularity in their mobility pattern and this is what we are trying to exploit in our proposed solution [7][9].

3 (p, t, d)-Model

Our clustering algorithm selects a MN as CH, if it satisfies the following criteria: 1) it has the highest probability, in comparison to other MNs within the same *virtual cluster* (see section 3.1), to stay for longer time in that cluster and 2) it has the minimum distance from the respective *virtual cluster* center (VCC). The first requirement tries to ensure that a highly mobile MN is not elected as a CH. The second is to ensure that by being located close to a VCC, the CH can have a uniform coverage over a specific *virtual cluster*. This in turn ensures that in subsequent CH changes, the area covered would not be impaired. We name our model the *(p, t, d)-model.* More accurately, it is the (p_{xk}, $t_{xk,}$ d_{xk})-model, where ‘p_{xk}’ is the probability that xth MN within kth *virtual cluster* having a distance ‘d_{xk}’ from the center of that cluster stays in it for some specified time period ‘t_{xk}’ (residence-time). Any xth MN within the kth *virtual cluster*, having $p_{xk} \geq p_{max}$, for $t_{xk} \geq t_{yk} \geq t_c$ (where ‘x’ ≠‘y’) and $d_{xk} \leq d_{min}$, can become a head of that *virtual cluster* - equation (2) introduced later is used in this process. Here ‘p_{max}’, ‘d_{min}’ and ‘t_c’ are system dependent, and ‘p_{xk}’ and ‘t_{xk}’ are determined based on the mobility prediction model described below. The key ingredients of the *(p, t, d)*-model are described below.

3.1 The Concept of Virtual Clusters

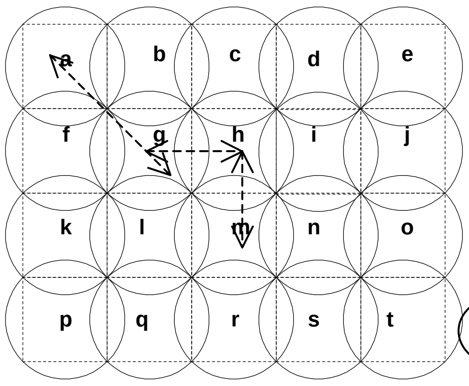

Figure 1. Concept of Virtual Clusters.

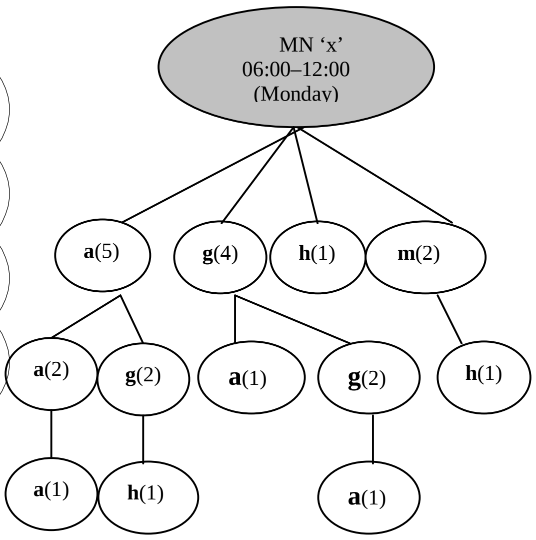

Figure 2. A Mobility Trie of a specific Mobile Node 'x'.

In order to make our mobility prediction viable, and our clustering and routing mechanisms scalable, we introduce the notion of *virtual clusters*. The idea is that a geographical area (or even the whole earth) is divided into equal regions of circular shape in a systematic way so that each MN can determine the circle it resides in if location information is known. The *virtual cluster* is ideally a circular region centered on a *virtual cluster* center (VCC) as depicted in Fig. 1. These VCCs are associated to a particular region in such a way that the resulting *virtual clusters* are nearly overlapping. These circle area regions are our *virtual clusters*; a *virtual cluster* becomes an *actual cluster* if MNs exist in it. Unlike in other clustering scheme discussed in Section 2, in our approach each *virtual cluster* is supposed to have a unique identifier based on the geographic location, which can be calculated using a publicly known function. It is necessary that each *virtual cluster* have a unique identifier for our mobility prediction algorithm to work in a scalable way. Each MN is supposed to have a complete picture of the locations of VCCs. In our context, each VCC is assumed to be away from each other by a fixed distance (which is not exactly true due to the earth's spherical shape). In order to make our mobility prediction analysis simpler, the MANET is modeled as a connected graph G = (V, E), where the vertex-set V represents the *virtual cluster*, and the edge-set E represents the adjacency between pairs of *virtual clusters*.

3.2 Mobility Prediction Model

User Mobility Pattern. Users tend to have favorite routes and habitual movement patterns, and those can be learned. Learning aids decision-making when reappearance of those patterns is detected, given that 'history repeats itself' [7][9]. Our model exploits this, and attempts to derive probabilistic prediction of particular user mobility by utilizing his accumulated movement-history. Our mobility prediction scheme is motivated by computational learning theory, which has shown that prediction is synonymous with data compression. The movement history of a MN is represented by a string '$\nu_1,\nu_2, \nu_3\ldots$' of symbols where $V = \{\nu_1, \nu_2, \nu_3\ldots\nu_n\}$ is the set of virtual clusters and ν_i denotes the virtual cluster id (VID), and such strings are generated by using a combination of time-based and movement-based tracking schemes [7][9]. The goal of our algorithm is to construct a universal predictor or estimator for the user mobility model. The proposed scheme attempts to create a dictionary of virtual cluster ids, which are treated as character symbols, and uses the dictionary to gather statistics based on movement history contexts, or phrases. The proposed mobility prediction algorithm is based on the Ziv-Lempel algorithm (LZ78) for data compression [7][9]. In addition to making predictions about the future movements of a particular MN, our model is used by each MN to predict its approximate residence-times of the virtual clusters it visits. A key aspect of our clustering and routing algorithm is residence-time as explained in Sections 3.3 and 3.4 respectively.

Mobility Prediction Scheme. The need for finding a universal variable-to-fixed coding scheme gave rise to the emergence of the LZ78 algorithm, where the coding process is interlaced with the learning process for the source characteristics [7][9]. The learning process is aided with the de-correlating process, which works by efficiently creating and looking up an explicit dictionary. This algorithm parses the input string '$\nu_1, \nu_2, \ldots\nu_n$' ($\nu \in V$) into c(n) distinct substrings $\omega_1, \omega_2, \ldots, \omega_{c(n)}$ such that for all $j \geq 1$, the prefix of substring ω_j is equal to some ω_i, for $1 \leq i < j$ [9]. Because of this prefix property, substrings parsed so far can be efficiently maintained in a multiway tree or trie [7].

Since each MN maintains its mobility database at a specific time in terms of a trie, we name such a trie *Mobility Trie* [7]. This trie is a probabilistic model corresponding to the dictionary of the LZ78 algorithm. Each leaf except the root in the *Mobility Trie* preserves the relevant statistics that can be used to predict the probability of following events. When a MN is switched on for the first time in a specific day of a week, the encoder (or predictor) of that MN would initialize the root of the trie according to the time and VID, and should be able to calculate the probabilities of all possible future events associated with that MN. In our model, an event occurs due to either time-based or movement-based updates. On seeing the actual event, the predictor of each MN walks down the trie and is ready to predict again. When an event is not in the *Mobility Trie*, a prediction fault is generated and the trie is updated by adding a leaf. In addition to representing the dictionary, the *Mobility Trie* can store statistics for contexts explored. A path from the root to any leaf ω in the trie represents a context. Fig. 2 shows an example trie formed while parsing the movement-history "aaagagggggaaghhmmhgaaaa…" – obtained from Fig. 1 – as "a, aa, g, ag, gg, gga, agh, h, m, mh, ga, aaa, …". The commas separate the parsed phrases and indicate the points of trie updates. As the process of incremental parsing progresses, larger and larger phrases accumulate in the *Mobility Trie*. Consequently estimates of conditional probabilities for larger contexts start building up. Intuitively, it would gather the predictability or richness of higher order Markov Models [9]. In other words, by modeling the sequence of events (symbols) generated during a specific time duration (for e.g. a specific day of a week) as those generated by a stationary r^{th} order Markov source, and predicting next events using the mobility prediction scheme derived from the LZ78 algorithm, we can predict not only which *virtual cluster* a MN will visit but also the approximate residence-time in it. Similarly, the blending probabilities, associated with the occurrences of next possible VIDs on the path segment to be reported by the next update event can also be calculated if the movement history context is known [7][9]. This technique is used by every MN ‘x’ to calculate the state probability (p_{xk}) for it to stay in *virtual cluster* ‘k’.

‘T_e’ at which update events are triggered based on the time-based updating, and the radius (R) of *virtual cluster* are two important parameters, and thus determine the prediction accuracy and hence the performance of our model. A compromise decision is necessary, when selecting values for these two parameters. Also an assumption is that all MNs have a common time reference (e.g. by using GPS).

3.3 Clustering Algorithm and Protocol

In this scheme, the decision as to the next CH to be selected depends on whether the current CH is available or not. If there is a primary CH or deputy CH available, they will make the decision on behalf of all MNs after getting the members’ Ω values (see equation (2) below). On the other hand if any of them does not exist, because of abrupt failure or error in prediction, then MNs within the *virtual cluster* will elect one as their CH in a distributed manner. For this algorithm to work well, each MN should maintain its past history in terms of a *Mobility Trie*. This trie is needed for every MN ‘x’ to determine its residence-time (t_{xk}) in *virtual cluster* ‘k’, and the state probability for it to stay in ‘k’, (p_{xk}), if the location information and time of the day are known. If however, a MN is unable to construct its trie, it can still use its distance in the criterion calculation, since $t_{xk} = 0$ (see equation (2) below) in this case. The MN that has the highest Ω can become the CH. In forming clusters, the CH has to make sure it can cover the whole area of the *virtual cluster*. Some of the terms used in this paper, such as ‘*HELLO*’ message, adjacent cluster, and gateway MNs, are similar to those specified in [4][5].

Each *HELLO* message, periodically broadcast by the CH – say every CH_HELLO_INTERVAL – carries the ID of the *virtual cluster* (VID) it covers, the VCC, the cluster’s radius and the neighbor-table, the latter being the set of MNs in the cluster [5]. Whenever a new MN receives this message from a CH, it can send a *JOIN* message immediately, if it is within the *virtual cluster*. The new MN includes in the *JOIN* message its approximate residence-time (t_{xk}) within its current *virtual cluster* ‘k’, its location information, and its *Mobility Trie* corresponding to the next ‘T_{mt}’ (system parameter) minutes. Whenever a CH receives a *JOIN* message, it checks first if the MN is within its *virtual cluster*. If it is, the CH includes it in the cluster, and appends its information to the neighbor-table. The exception to this case is when the MN’s expected residence-time within the cluster is minimal. If, on the other hand, the MN is not within the *virtual cluster*, it will simply not be included. In either case, the MN has to wait for at least the next two successive CH_HELLO_INTERVAL periods to check whether it has been included. If not, it has to re-transmit the *JOIN* message. From the periodic neighbor-table that a CH broadcasts, each member of a cluster can build its own neighbor-table. A MN can be a member of up to four maximum adjacent *virtual clusters*. This specific MN would then behave as gateway or forwarder between those clusters [2][5]. Having become a member, each MN within a particular *virtual cluster* is supposed to disseminate a *HELLO* message to its respective CH periodically – say every MN_HELLO_INTERVAL, where MN_HELLO_INTERVAL > CH_HELLO_INTERVAL. In the *HELLO* message, each member specifies if it is acting as a gateway or as an ordinary node. These control messages are relayed by intermediate MNs only within the *virtual cluster*. On the other hand, periodic *HELLO* messages by CHs are unicast by gateways between CHs of adjacent *virtual clusters* to an extent that can be limited for scalability. This is to enable CHs to get the topology information of adjacent clusters. Given that each CH knows the predicted residence-time of each MN within its cluster, it deletes the entry associated with a particular MN exactly 't_{to}' (system parameter) seconds after its residence-time expires. This effect will be reflected in every *HELLO* message a CH broadcasts periodically. Also after having become a member of a cluster, each MN can dynamically increase its MN_HELLO_INTERVAL until the new CH election process is triggered, given that it knows the predicted residence-time of the CH. This is economical with respect to both bandwidth and transmission power.

The unique aspect of our protocol is that, before a particular CH_i becomes unavailable, it has to trigger the "*CH Changeover Event*". This happens exactly 't_{ce}' (system parameter) seconds before the time at which the present CH has been predicted to leave the serving *virtual cluster*, unless CH_i fails abruptly. Whenever a CH broadcasts "*CH Changeover Event*" message within its *virtual cluster*, each member MN should perform the clustering criterion calculation process. When this event occurs,

- Each MN has to calculate its distance from the center (VCC) of a particular *virtual cluster*. Assuming an MN with an identifier 'x', whose location co-ordinates at time 't' are (x_{xk}(t), y_{xk}(t)), in the kth *virtual cluster*, whose center's Cartesian co-ordinates are (x_{ck}, y_{ck}), its distance at time 't' can be calculated by :

$$d_{xk}(t) = \sqrt{(x_{xk}(t) - x_{ck})^2 + (y_{xk}(t) - y_{ck})^2} \qquad (1)$$

- Each MN has to predict how long it will remain in the present *virtual cluster*, which it can obtain from its *Mobility Trie*. Let the resident-time of a MN with an identifier 'x' within kth *virtual cluster* be 't_{xk}', and the state probability for the xth MN to stay in kth *virtual cluster* be 'p_{xk}'.

Based on the above, each MN 'x' is required to calculate, the clustering criterion factor Ω_x which is given by:

$$\Omega_x = \begin{cases} p_{xk}\left[\dfrac{t_{xk} - t_{th}}{d_{xk}(t)}\right] & \forall \quad d_{xk}(t) \neq 0 \\ p_{xk}\left[t_{xk} - t_{th}\right] & otherwise \end{cases} \qquad (2)$$

The formula, given by equation (2), tries to ensure that the resulting clusters are more stable, and have uniform coverage by respective CHs. Ω_x is proportional to expected residence-time and the probability to stay in a *virtual cluster*, and inversely proportional to distance from respective VCC. 't_{th}' is the threshold value (system dependent) for the residence-time. At the end of calculation, each MN will disseminate a Ω-message to the CH. Based on the information received from its members, the CH will select the MN that has the highest Ω value as the new primary CH. It will also select two assistant (deputy) CHs for reliability purposes – again based on Ω-values. The present CH will then broadcast this information using a *SUCCESSOR* Message. As soon as the new CH receives this, it will assume its status as the new primary CH, and so will the two assistants.

If the first assistant CH sees that it has not received a *HELLO* message from the primary CH during the last two consecutive CH_HELLO_INTERVAL periods, it will take over as the primary CH informing its deputy as its first assistant CH. If, however, the second assistant has not received any *HELLO* message either from the primary or first assistant during the last four consecutive CH_HELLO_INTERVAL periods, it will assume duty as the primary CH. In case an ordinary MN notices no *HELLO* message from any CH during six consecutive CH_HELLO_INTERVAL periods, the first MN to notice this will assume duty as the temporary CH, and it will immediately trigger the "*CH Changeover Event*". Accordingly, each MN will become aware of other MNs' **Ω-**values. In this case, the MN that has the highest value for **'Ω'** will be elected as the new primary CH in a distributed manner. Deputy CH election will follow and this information will be broadcast through a *SUCCESSOR* message. The new CH will then start broadcasting *HELLO* message as usual. If however, an ordinary MN has not received any of the above control messages for more than eight consecutive CH_HELLO_INTERVAL periods, then it will elect itself as the CH. In this algorithm, if more than two MNs have the same value for **'Ω',** the one with the lowest ID will be selected as the new CH. Unlike in any other clustering algorithm, our algorithm has another unique feature in the sense that whenever a CH leaves the *virtual cluster* it has served, it will loose its CH status. In this way this algorithm ensures that no other visiting MN will cause transient instability.

With the *Mobility Trie* information that a CH receives from its new member MN, the CH becomes aware of future mobility patterns of its cluster members at least for the next 'T_{mt}' minutes. In case any member MN continues to stay within the same *virtual cluster* beyond 'T_{mt}', then such MN is expected to unicast its new *Mobility Trie* corresponding to the next 'T_{mt}' to the CH 't_{mt}' seconds before the original 'T_{mt}' expires. The same procedure is expected instantly from every member MN of a cluster, whenever CH changeover occurs.

3.4 QoS Routing

QoS support in networks whose topology varies with time requires addressing: 1) connection-level issues related to path establishment and management to ensure the existence of connectivity between the source and the destination and 2) packet-level performance issues in terms of delay bounds, throughput, and acceptable packet loss rates [3]. Most of the current MANET routing protocols and QoS models have difficulties in addressing these two requirements because of the following reasons. Most of them fail to capture and use neighbor-availability information when constructing routes between a source-destination pair and, as such, they cannot guarantee a route that can last for a reasonable amount of time. If the constructed route fails, a route reconstruction process will be required, which will result in packets loss, and hence it will have bad transient effects on QoS. In addition, frequently changing routes could increase the delay jitter experienced by applications. Most current on-demand routing protocols tend to use a route until a link breaks. In order to minimize such effects, the route construction process should carefully select appropriate neighbor MNs in such a way that the constructed routes would survive longer. The FORP approach [6] tries to do that by making predictions based on neighbors' velocities but it suffers from a general problem with most MANET routing protocols: they tend to use flooding as their routing strategy and, as such, they do not scale. Also, none of the existing routing protocols have an answer for the situation where only the source and destination (and not the intermediate MNs) move simultaneously.

In MANETs, information arriving late due to latency can drive network routing into instability. However, in our model each MN is supposed to make movement predictions, and such information is disseminated using our clustering mechanism. Even if this information reaches late, it still has value to the recipient. We propose here a distributed QoS routing scheme that selects a longevity route with sufficient bandwidth resources to support real-time, potentially adaptive applications. Our goal is to provide a seamless service by reacting before the connectivity breaks. This is built on the INSIGNIA framework with important modifications and extensions in order to improve performance [13]. We explain below how our routing strategy attempts to satisfy the above connection-level and packet-level requirements, and how it leads to performance improvements. The aim of our routing protocol is to construct a route between a source-destination pair, which has the higher probability to survive longer, and have sufficient resources available. The task of resource reservation is the other key ingredient. Since it has been experimentally confirmed that routing protocols that do not use geographic location in the routing decision are not scalable, our routing protocol is position-based [10]. In this approach, a source is supposed to know the location information of a destination by using location services [10]. We opt for the Grid Location Service (GLS) as it is easy to incorporate it in our clustering scheme, and it is considered as the efficient means by which a distributed database of geographic positioning information can be created, maintained and queried [11]. Due to scalability issues, our protocol avoids the use of flooding. Hence, the ‘Greedy Packet Forwarding’ (GPF) strategy is considered suitable in our approach in making forwarding decisions [10]. In GPF, the sender of a packet includes the approximate position of the recipient in the packet. When an intermediate MN receives a packet, it forwards the packet to a neighbor towards the general direction of the recipient. In considering whether to choose a neighbor node, ‘y’ as a forwarder (in the route-establishment process only), a source or any intermediate MN ‘x’ of our algorithm considers the following two criteria (let both ‘x’ and ‘y’ be in *virtual cluster* ‘k’):

1) Whether the node ‘y’ has the highest value for the ‘selection criterion’ (Γ_{XY}) with respect to ‘x’, as given by equation (3).

$$\Gamma_{XY} = (p_{xk}) * (t_{xk}) * (p_{yk}) * (t_{yk}) * (M_X^{rel}(Y)) \qquad (3)$$

Notations above have the usual meaning, and $M_X^{rel}(Y)$ is given by (4) below.

2) Whether the node can satisfy the ‘compass-routing’ algorithm with the ‘nearest with forward progress (NFP) in mind [10].

In forwarding a packet, the compass-routing algorithm selects the neighbor closest to the straight line between sender-destination pairs, such that it tries to minimize the spatial distance a packet travels. In this way, compass-routing tries to minimize unnecessary interference. In NFP, the packet is transmitted to the nearest neighbor of the sender that is closer to the destination [10]. In our scheme, NFP is used in conjunction with compass-routing in order to make sure that any unpredictable micro (i.e. intra-*virtual cluster*) movement of any intermediate node does not result in any unnecessary link breakages [12].

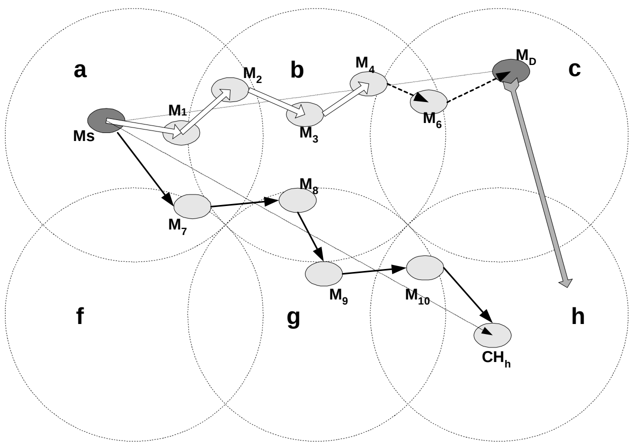

Figure 3. Intelligent Path Establishment.

As in INSIGNIA, when a source node wants to establish a flow to a distant destination, a route has to be established first, and reserved for that flow [13]. Hence a source is expected to send a route request (REQ) packet. In addition to specifying the MAX-MIN bandwidth requirements of an adaptive application, the source is expected to include fields for 'LIFETIME INDICATOR' and the destination's approximate location. This 'LIFETIME INDICATOR' field is initially set to residence-time value of the source, and enables all the MNs, which constitute the whole route from a source to a destination, to know the bottleneck lifetime (t_{bot}) of the route established as explained below.

In choosing the next forwarder, the two requirements specified above are considered instead of flooding. After selecting the appropriate forwarder, the REQ packet is unicast from the sender to the forwarder. If the selected forwarder is able to support the source-specified MAX-MIN bandwidth, it will allocate resources accordingly, and will unicast the REQ to another forwarder. At the same time, the former is required to unicast an *ACCEPTED* control packet to the sender. If, on the other hand, the selected forwarder is not able to support the source specified bandwidth requirement, it is required to unicast a *REJECTED* control packet to the sender. In the latter case, the sender has to look for another appropriate forwarder. With the *ACCEPTED* and *REJECTED* message types, the latency involved in the path construction process is minimized. If the REQ packet reaches the destination successfully, the latter will inform the status of flow establishment to the source using QoS reporting [13]. In order to distinguish 'fresh' messages from 'stale' ones, it is wise to maintain sequence numbers as in AODV, although the 'loop' problem is not as severe as in the AODV case [14]. Loop-freedom is achieved in our approach, because our routing strategy does not rely on flooding. In the same fashion to INSIGNIA, each node the REQ packet traverses is expected to indicate in the REQ packet its predicted residence-time at its present location in the 'LIFETIME INDICATOR' only if its own residence-time is less than that appears in the REQ packet. Also, such an intermediate node has to indicate the possible bandwidth it can support in the 'BANDWIDTH INDICATOR' [13]. As soon as the path is established, the source and the destination are expected to exchange their respective *Mobility Tries* corresponding to the next

‘T_{mt}’ minutes. This *Mobility Trie* information has to be updated periodically, if the session between a source-destination pair is likely to exceed ‘T_{mt}’ minutes. Therefore, the system parameter ‘T_{mt}’ should take an optimum value, as too small a value requires frequent exchanges, increasing control traffic overhead. On the other hand, too big a value necessitates sound prediction.

The key aspect of our routing algorithm is that since the source and destination know about each other’s *Mobility Tries* and the bottleneck lifetime (t_{bot}) of the current route, they can predict each other’s movement and establish paths proactively before the present path actually breaks. Suppose as shown in Fig. 3, a source ‘Ms’ from ‘a^{th}’ *virtual cluster* has initially established a path successfully to the destination, ‘M_D’, which is initially in the ‘c^{th}’ *virtual cluster*. From the *Mobility Trie* the source obtained from ‘M_D’, the former gets to know that after ‘t_1’ minutes ‘M_D’ would move to ‘h^{th}’ *virtual cluster*. Hence exactly ‘t_{rc}’ (system parameter) seconds before ‘t_1’ minutes, the source would initiate a route construction process for the same flow up to the cluster head ‘CH_h’ of the ‘h^{th}’ *virtual cluster*, while the current route is still active. Once the source predicts the destination to be in ‘h’, ‘Ms’ can start using the second route. Since this one is constructed up to the CH of the new *virtual cluster*, it is the responsibility of that CH to route the packet to the destination. If the prediction works well, by the time ‘Ms’ starts using the second route ‘M_D’ would have already become member of cluster ‘h’, and routing becomes easy for the cluster head ‘CH_h’. For this purpose, CHs are required to reserve some bandwidth in order to accommodate this re-routing process. Another unique feature is that even if packets destined for ‘M_D’ arrive in the ‘c^{th}’ *virtual cluster* after ‘M_D’ has moved to ‘h’, the cluster head ‘CH_c’ would be able to forward the packets to the intended recipient. This is possible, because the cluster head ‘CH_h’ is supposed to know the *Mobility Trie* of its previous member ‘M_D’ until such entry becomes stale in its neighbor-table. Similarly, even if both the source and the destination move simultaneously, both of them can still continue their session by proactively constructing routes. This is possible because the source and destination exchange periodically their respective *Mobility Tries*. If, however, the ‘bottleneck lifetime’, is due to an intermediate node, exactly t_{RRD} seconds before the expiration time (t_{bot}) of the current route, a local route repair process has to be initiated to the same destination (if a path is still required) – as explained by the following example. In the initial path between ‘Ms’ and ‘M_D’, if we assume that M_4 is the bottleneck node in terms of lifetime along the path (i.e. having the lowest lifetime), local route repair has to be initiated by M_3 well before M_4 moves and breaks the route. Depending on the local conditions prevailing at that time, M_3 will select an appropriate forwarder. Once the local repair has been performed successfully, it is important to determine and disseminate the new bottleneck lifetime and bandwidth along the path. Since we use a ‘soft-state’ approach as in INSIGNIA, the resources would be released automatically from the previous route when the ‘soft-state’ timer expires [13]. The second route may be either completely disjoint or part of the first route depending on various conditions such as the extent of movement by the source, destination or both, neighbors’ availability and communication overhead.

Each MN’s routing table is derived from its own neighbor-table. Tables are periodically updated, and stale entries are removed depending on whether a neighbor’s residence-time has expired or the MN has not heard anything from its neighbors for a NEIGHBOUR-TIMEOUT-INTERVAL. MN macro-mobility (between *virtual* *clusters*) is handled through the use of *Mobility Tries* as explained; however, MN micro-mobility (within a *virtual cluster*) is handled through power measurements from neighbors as explained below. We use it as the secondary mechanism to make sure that link breakages due to micro-mobility are kept as low as possible. Under Friis' free space propagation model, the signal power detected, say 'RxPr', at the receiving MN is indicative of the distance between the transmitting and receiving MN pairs. Since it is very difficult to calculate the exact distance between two MNs without wasting bandwidth, we try to use the MOBIC model that defines a relative mobility metric, $M_X^{rel}(Y)$, at a MN 'x' with respect to 'y' as [4]:

$$M_X^{rel}(Y) = 10\log_{10}\left(\frac{Rx\,\mathrm{Pr}_{Y\to X}^{new}}{Rx\,\mathrm{Pr}_{Y\to X}^{old}}\right) \qquad (4)$$

Every MN 'x' determines the above mobility metric for each neighbor 'y' by making subsequent power measurements. A negative value for $M_X^{rel}(Y)$ indicates that 'x' and 'y' are moving away from each other, and a positive value indicates that they are moving towards each other. This micro-mobility pattern is taken into consideration as an input in the route construction process. In this way, in our QoS routing scheme routes are constructed with nodes' macro and micro mobility patterns in mind.

4 Evaluation Through Simulation

4.1 Evaluation of the Clustering Protocol

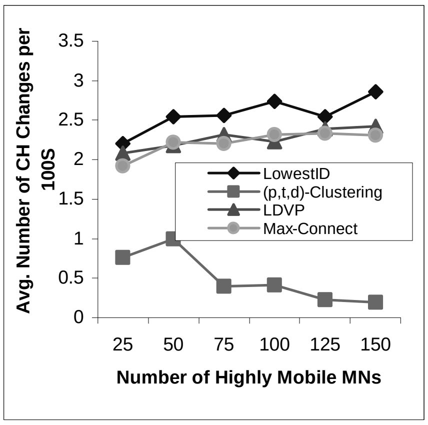

Figure 4. Clustering Instability as a function of Number of Nodes.

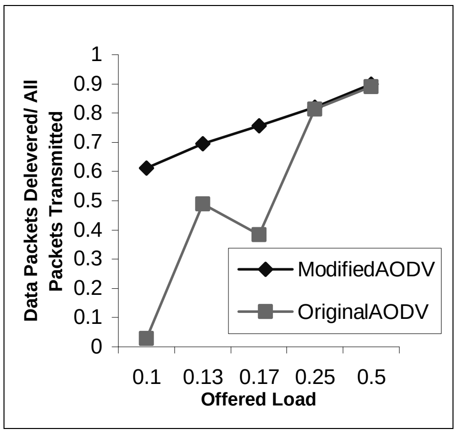

Figure 5. The number of Data Transmitted per Total Packets Transmitted as a function of Offered Load.

The simulation work attempts to compare the performance of our clustering algorithm with the Lowest-ID, maximum-connectivity (Max-Connect), LDV clustering algorithms, in terms of the stability of clusters formed. The y-axis of Fig. 4 shows the frequency of CH changes by each MN, and hence measures the (in)stability associated with each clustering algorithm. (The lower the frequency of CH changes, the more stable the cluster is). As it can be seen from Fig. 4, the *(p, t, d)*-clustering algorithm leads to more stable cluster formation. (Due to space limitations, only a few simulations are shown here).

We performed our simulations using the GloMoSim simulation package [8]. In simulating the *(p, t, d)*-model, some MNs take a regular mobility pattern, whilst others always take a random pattern. In order to simplify the simulation, each leaf of the *Mobility Trie* associated with each MN has two branches only. This again simplifies the calculation of 'p_{xk}' of equation (2). The distance between any two VCCs is 200m, and the radius R of a *virtual cluster* is 142m. Lowest-ID, LDV, and maximum-connectivity clustering algorithms form 2-hop clusters. Since it was necessary to ensure that clusters formed by all the schemes cover approximately equal area, the transmission range of each MN is set to 71m. A terrain-area of 600m X 600m with nine *virtual clusters* was considered in our simulations.

4.2 Evaluation of the Routing Protocol

Initial Approach. In our initial attempt, we incorporated our ideas into the AODV routing protocol [14]. Here our prime task is to construct longevity routes. Link lifetime plays a significant role in the path discovery process in order to construct a route that has a higher probability to survive longer. In our approach, the source is required to include fields for lifetime and probability of stability (PoS) in the RREQ message of the route discovery process. The source includes its residence-time (t_{yk}) and the probability to stay at its current location (p_{yk}) in the lifetime field and the 'PoS' field of the RREQ respectively. As in the normal AODV case, this message is broadcast by the source. The decision factor for every intermediate node 'y' to decide whether to or not to forward the received RREQ is given by the following expression:

λ_y = (Residence-time of 'y' at its present location) * (probability of 'y' to stay at its present location)

$$= (t_{yk}) * (p_{yk}) \qquad (5)$$

As this RREQ traverses each MN 'j', the latter should include its residence-time (t_{jk}) at its current location (*virtual cluster*) in the lifetime field and the probability to stay (p_{jk}) in the 'PoS' field, only if its λ_j is less than the equivalent 'λ' of RREQ. Each intermediate node will forward the RREQ only if its own λ-factor and TTL of the received RREQ are greater than zero, provided that the node has not received the same RREQ before. In case, an intermediate node receives the same RREQ (same originator IP address and RREQ ID, but with different λ-factor) for the second time via a different route within at least the last PATH_DISCOVERY_TIME, it will check whether it has a valid entry (fresh) for the originator in the routing table. If it has a valid entry, the node will check whether the recently received RREQ has a higher value for the λ-factor than that in the route table. If it is the case, the route table will be updated with details from the recently received RREQ, and the node will forward the RREQ to other nodes; otherwise the recently received RREQ will be simply discarded. Finally when it reaches the destination, the latter can deduce the 'bottleneck-lifetime' of the whole route and its corresponding probability of stability. In this approach, there is a trade off between route optimality and route stability. A route that has a larger value for λ-factor (i.e. higher 'bottleneck-lifetime' and higher 'PoS') will remain active longer, but it might not necessarily be the route with the smaller hop count. If a node receives a RREQ for a destination, and either has a fresh enough route to satisfy the request or is itself the destination, the node generates a RREP message. If it is the destination, it includes the 'bottleneck-lifetime' of the selected path in the 'lifetime' field of the RREP message. As in the original AODV, the RREP is unicast toward the originator of the RREQ. Within the next PATH_DISCOVERY_TIME after the node has generated the first RREP, the node may receive the same RREQ via different paths (routes) and becomes aware of the 'bottleneck-lifetime' t_{bot_i} and 'PoS' p_i of each route 'i'. The node waits for t_{wait} seconds, after it has generated the first RREP message, in order to generate the second RREP for a more stable route. Within the waiting time (t_{wait}), the node may receive a number of RREQ messages that reach the former via different paths. The node generates the RREP for the second time for the same RREQs that have traveled via different routes, only if the λ-factor of one of the recently received RREQs is higher than that of the first received RREQ. On the other hand, when multiple routes have the same λ-factor, the route with the minimum number of hops will be selected. With the RREP message, all the nodes that are members of the selected route and especially the source node, become aware of the bottleneck-lifetime and the stability of the selected route. This in turn enables the source to maintain its routes proactively as described below. Exactly t_{RRD} seconds before the expiration time (t_{bot_i}) of the selected route 'i', the source has to initiate a route re-discovery process to the same destination, if it is required. The source-destination pair can continue to use the existing route until the new route is proactively constructed. Once this happens, seamless handoff will take place from the old to the new route. Another important point here is that whenever an intermediate node generates a RREP message, it has to include the 'bottleneck-lifetime' of the whole route from the source to the destination. Also t_{wait} should take an optimal value.

Initial Evaluation. In the evaluation process, we simulated and compared our schemes against the original AODV [14] using GloMoSim. Due to space limitations, only a single metric is considered here: number of total packets transmitted per data packet delivered. This metric is the number of data packets delivered to destinations divided by the number of all packets (i.e. data and control) transmitted. Fig. 5 shows the channel access efficiency as a function of load, and again performs better compared to the standard AODV at a moderate load. Our scheme would outperform even better at heavy loads, if proper load balancing mechanism is employed.

5 Conclusions and Future Work

In this paper we presented new clustering and routing schemes that make use of intelligent mobility prediction and location information. To facilitate this, we introduced the '*virtual cluster*' concept. Associating clusters to geographic locations dynamically results in many benefits. The stability improvement, however, depends on the accuracy of our mobility prediction. We have demonstrated that our clustering scheme results in more stable clusters than those of other well-known schemes. This in turn results in performance improvements, and this way of making intelligent mobility prediction is beneficial in environments that rely on the accurate capturing of context. We have also performed an initial evaluation of routing extensions to AODV that take into account our mobility prediction model for constructing longevity routes. First results are encouraging, showing better performance than AODV except in high loads; this can become better with load balancing. On the other hand, we have not yet properly evaluated the QoS routing protocol presented in section 3.4; we are in the process of doing this. In our work, we envisaged a large-scale deployment of non-authority based longer-term ad hoc networks, and the proposed clustering and routing protocols work effectively in such environments. Our proposed scheme could be possibly also applied to short-term spontaneous ad hoc networks by capturing context information in such environments; we also plan to investigate this in the future. Our findings should be helpful for the design and implementation of intelligent location-aware ad hoc networks.

References

[1] L.Blazevic, L.Buttyan, S.Capkun, S.Giordono, J-P.Hubaux and J-Y.Boudec, “Self-Organisation in Mobile Ad Hoc Networks: The Approach to Terminodes”, IEEE Communications Magazine, June 2001, pp 166 –173.

[2] J.Habetha, A.Hettich, J.Peetz, and Y.Du, “Central Controller Handover Procedure for ETSI_BRAN HiperLAN2 Ad Hoc Networks and Clustering with Quality of Service Guarantees”, Mobile and Ad Hoc Networking and Computing (MobiHOC), 2000, pp. 131 – 132.

[3] B.McDonald, and F.Znati, “A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks”, IEEE Journal on Selected Areas in Communications, vol. 17, August 1999, pp 1466 –1487.

[4] P.Basu, N.Khan, and D.C.Little, “Mobility Based Metric for Clustering in Mobile Ad Hoc Networks”, Workshop on Distributed Computing Systems, 2001, pp. 413 –418.

[5] M.Jiang, J.Li, and Y.C.Tay, “Cluster Based Routing Protocol (CBRP) Function Specifications”, IETF Draft, August 1999.

[6] W.Su, S-J.Lee, and M.Gerla, "Mobility Prediction and Routing in Ad Hoc Wireless Networks", International Journal of Network Management, vol. 11, no. 1, John Wiley & Sons, Jan.- Feb. 2001, pp. 3 – 30.

[7] F.Yu, and V.C.M.Leung, “Mobility-Based Predictive Call Admission Control and Bandwidth Reservation in Wireless Cellular Networks”, IEEE INFOCOM, 2001, pp. 518 – 526.

[8] X.Zengu, R.Bagrodia, and M.Gerla, “GloMoSim: A Library for Parallel Simulations of Large-scale Wireless Networks”, Proceedings of the 12th Workshop on Parallel and Distributed Simulations, May 1998.

[9] A.Bhattacharya, and S.K.Das, “LeZi-Update : An Information-Theoretic Approach to Track Mobile Users in PCS Networks”, 5th Annual ACM Int'l Conference on Mobile Computing and Networking (MobiCom'99), August 1999 , pp. 1-12.

[10] M.Mauve, J.Widmer, and H.Hartenstein, “A Survey on Position-Based Routing in Mobile Ad Hoc Networks, IEEE Network, vol. 15, no. 6, Nov./Dec. 2001, pp. 30 – 39.

[11] J.Sucec and I.Marsic, "Location management for hierarchically organized mobile ad hoc networks," Proc. IEEE WCNC 2002, March 2002, pp. 603-607.

[12] W-I.Kim, Y-J.Suh, and S.An, "A Reliable Route Selection Algorithm in Mobile Ad-hoc Networks," Journal of KISS: Information Networking, vol. 29, no.3, Jun. 2002, pp.314-323.

[13] S-B.Lee and A.T Campbell, “INSIGNIA: In-Band Signalling Support for QoS in Mobile Ad Hoc Networks”, Proc of 5thInternational Workshop on Mobile Multimedia Communications (MoMuC,98) , Berlin,Germany, October 1998.

[14] C. E. Perkins, E. M. Belding-Royer, and S.R. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, draft-ietf-manet-aodv-12.txt.