An auto-Configurable, and, Adaptable, Metric-driven Cluster-head Organisation for Hybrid Multi-hop Routing

Nikos Migas and William Buchanan
DSMA, School of Computing, Napier University, 10 Colinton Road, EH10 4DT
{n.migas, w.buchanan}@napier.ac.uk

Abstract

Multi-hop ad-hoc routing is a challenging issue, because of the dynamic network topology, and limited capabilities of resource-constrained mobile devices. This paper proposes a metric-driven clustering organisation of participating nodes which provides an effective way to reduce network overhead, in contrast to traditional flooding methods. This novel method creates a network backbone for data routing and location management, which is composed of the fittest nodes. The key metrics are: mobility, buffering capabilities, throughput, network state, utilisation, and battery reserves. The paper presents the overall agent-based model, and describes the adaptability of the cluster-head metrics, and the automated reconfigurability of an ad-hoc network.

1. Introduction

Recent advances in technology provided the ground for highly dynamic, mobile, infrastructure-less networks, namely, ad-hoc networks. These are typically formed on a temporary basis, and, thus, they do not have a fixed infrastructure that nodes could rely on for location management and data routing. With this, nodes have to create their own structure in a dynamic way, and use their wireless interfaces for forwarding data packets to the next hop, along a multi-hop manner, and also participate equally in the tasks delegated to them by the ad-hoc routing protocol.

Despite their enormous benefits, such as in military operations, disaster relief, and commercial applications, the full potential cannot be reached unless certain issues are resolved. These mainly involve routing, as the lack of an infrastructure imposes a heavy burden on mobile devices that must maintain location information, and route data packets, in a multi-hop manner. Specifically, typical ad-hoc routing devices, such as Personal Digital Assistants (PDAs), are limited in respect to the available throughput, life-time, and performance, that these may provide, as routing elements [1, 2].

Traditional ad-hoc routing methods, initially designed for fixed networks, such as link-state [3] and distance vector [4], are typically based on the shortest path calculation [5], where each node applies a shortest path algorithm from itself to all other destination nodes, and transmits its data packets through the shortest route. However, routing protocols for fixed networks have not been specifically designed to provide the dynamic and self-organisation which is required for mobile ad-hoc networks [6].

In general, ad-hoc routing protocols can be categorised by the way they maintain routing information. For example, routing protocols which maintain routing information for each node on the network, at all times, can be categorised as proactive, while protocols which discover routing information only when it is required, can be categorised as reactive. A third category is hybrid, which shares common characteristics with proactive and reactive.

The main disadvantage of the proactive approach is that each device is required to store large routing tables in its memory, and often produces high network overhead by periodically exchanging frequent routing updates. Thus, proactive routing protocols introduce scalability problems, as large amounts of data are often broadcasted. The on-demand methods aim to reduce the high network overhead imposed by the proactive methods. Network nodes only react when a route is required between a source and a destination node, and there is no need to maintain routes to destinations in which they are not communicating with. The reactive method has been proven to be simpler, and more efficient and scalable, in comparison to the proactive method. However, it can introduce high-latency, as a node is required to initiate the route discovery process each time a data packet needs to be transmitted to a destination for which the node does not have a route. Hybrid routing protocols aim to increase scalability, and, at the same time, reduce the on-demand route discovery overhead. Protocols in this category normally organise the network into a number of logical structures, such as zones, trees, or clusters, where, at each structure, nodes exchange routing information proactively, while they initiate route discovery for distant nodes, that is, for nodes that do not belong to the same domain. An extensive review of multi-hop routing protocols can be found in [7].

This paper proposes a metric-driven clustering formation approach, which is a modification of the standard lowest-ID algorithm [8], and aims to reduce frequent re-clustering, as well as, protect the devices' vital resources, especially resource-constrained ones. Most importantly, the proposed approach is self-organised, and adaptable to dynamic environmental changes, in addition to dynamic changes of the internal state of participating nodes, such as battery remains, utilisation, network state, and so on.

The rest of this paper is organised as follows. Section 2 compares flat with cluster-head routing structured protocols, and evaluates the benefits of the proposed metric-driven cluster-head approach. Section 3 describes stationary and mobile agents, and their applicability to wireless networks, especially multi-hop ad-hoc networks. Section 4 presents the overall agent-based model, the cluster-head metrics, and defines the reconfigurable and adaptable nature of the proposed metric-driven routing protocol. Section 5 concludes the paper and presents the future work.

2. Clustering formation

In respect to ad-hoc networks, clustering formation refers to the process which imposes a logical structure or hierarchy to an otherwise disoriented network, where mobile nodes are grouped into overlapping or/and adjacent clusters. A cluster-head is elected in order to provide coordination of data transmissions within its own cluster. The wireless range of a cluster-head defines a cluster, and thus every node within its transmission range belongs to the cluster. Therefore, every node within a cluster can communicate with the cluster-head, and, possibly with each other. Nodes which are situated in the edges of two or more clusters are called gateways and are responsible for inter-cluster routing. A cluster-head ad-hoc routing protocol can take advantage of the clustering formation, and, thus, efficiently minimise the flooding traffic during route discovery by allowing packets to be routed along a repeated sequence of alternating cluster-head and gateway node pair(s).

As an example, the communication complexity (CC) for route-discovery and route-maintenance in CBRP [9] is $O(2X)$ and $O(2A)$, respectively, where X is the number of clusters and A is the number of affected nodes, whereas the CC in AODV [10] is $O(2N)$, where N is the total number of nodes in the network [7]. In addition to reducing network overhead, the cluster-head structure may also provide a convenient framework for the development of important features such as wireless channel separation (among clusters), routing, and bandwidth allocation [11].

Two fundamental clustering algorithms have been initially proposed in the literature: lowest-ID [12]; and highest-connectivity [13], and they have both been later revised in [8]. According to the lowest-ID algorithm, the node with the lowest ID among its neighbours is elected as a cluster-head, that is, a unique number, such as the node's IP address, whereas, according to the highest connectivity, the node with the most mutual bi-directional links among itself and its neighbours is elected as a cluster-head. The authors in [11] have shown that, in most cases, the lowest-ID algorithm performs better than the highest-connectivity in terms of cluster stability. In addition, they proposed a modified version of the lowest-ID, which results in least cluster-head changes (LCC).

2.1. Metric-driven clustering

The standard lowest-ID and highest connectivity algorithms do not incorporate mobility factors, which can significantly influence the stability of clusters, and should thus be a key factor in clustering formation. In other words, nodes with high mobility patterns should not be chosen as cluster-heads, as their rapid movements may result in frequent cluster rearrangements. A study which proposed the usage of mobility metrics for cluster formation and selection can be found in [14]. The authors proposed a novel mobility metric, called MOBIC, which is based on the ratio of power levels due to successive receptions at each node. The calculation of the MOBIC metric does not involve a GPS, and generally assumes that any signal strength measured by a receiving node, *PxPr*, directly indicates the distance between the transmitter and receiver node pair. Although this method cannot be used to accurately measure distance in a real-life application, successive power measurements of two or more consecutive transmissions from a neighbouring node may allow the calculation of the relative mobility between the nodes. Simulations proved that the use of MOBIC, as a cluster-head metric (CH_M), can reduce the rate of cluster-head changes by 30% when compared to the standard lowest-ID algorithm.

However, mobility criteria alone do not guarantee that the elected cluster-heads will be fit enough to cope with demanding intra-cluster routing and location management. A good example lies on resource-constrained devices, such as PDAs, which typically provide low throughput, have low buffering capabilities, are battery-driven, and, can thus, easily, become performance bottlenecks [1, 2]. Thus, factors such as these should be incorporated to the cluster-head metric. With this, low performance devices with high mobility patterns will not be used as cluster-heads, unless there are no other fitter nodes. In addition, the CH_M must be adaptable to changes in mobility patterns, as well as to changes in the device's internal state, such as utilisation, battery reserves, and so on. Furthermore, the CH_M should be reconfigurable, that is, accurate weighting should be applied to each factor, according to the requirements imposed by different network topologies.

3. Intelligent software agents

Software agents are distinguished from normal programs, due to the agents' abilities to execute in distributed heterogeneous computing environments, and their ability to automate user-tasks [15]. The authors in [16] suggest that intelligent agents typically perform three functions: sense the environment they live in; reason about the environment; and perform actions that affect the environment. The authors in [17] propose that intelligent agents typically have characteristics, such as: autonomy, that is, they can operate without any direct intervention from humans or other software; social ability, that is, they communicate with other agents; reactivity, that is, they sense their environment and perform actions based on this knowledge; and pro-activeness, that is, they are goal-oriented.

A mobile agent is an entity that inherits some, or, all characteristics of an intelligent agent, and, in addition, it has the ability to initiate its own migration, that is, its data state, code, and possibly, its execution state, and thus transfer itself to another agent-enabled host and resume execution on the new host. The authors in [18] suggest that mobile agents act on behalf of their user, as well as other software entities, that need their services, and should thus be kept fully responsible for their mobile agents' actions, as the autonomy of agent migration is an issue that may raise security concerns. The authors in [19] suggest that a mobile agent usually knows its user's preferences and problem-solving techniques in a specific context, which then efficiently implements in an automated manner. The authors in [20], as well as in [15], identified mobile agent essential models, which include: a life-cycle model; a computational model; a security model; a communication model; and, finally, a navigation model.

The mobile agent paradigm greatly differs from the traditional client-server communications model. As shown in Figure 1, in the client-server case, the client requests a service from the server, and the server replies back with the results, whereas, in the mobile agent case, the actual computation is transferred to the server computer, and all interactions are performed locally. By moving the computation close to the needed resources, mobile agents can reduce communications that would otherwise take place over the network, and thus reduce bandwidth and latency requirements. In addition, mobile agents can encapsulate protocols, execute asynchronously and autonomously, use parallel processing, adapt dynamically, and are robust and naturally heterogeneous [21].

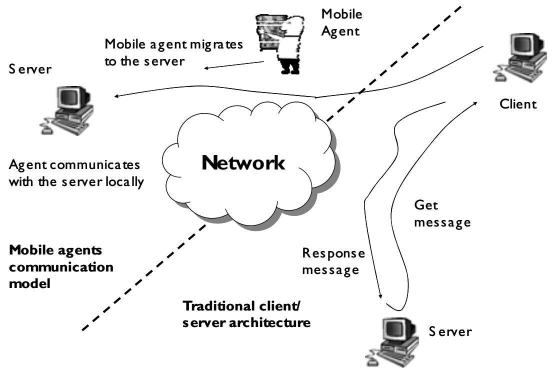
Mobile Agent
Mobile agent migrates to the server
Server
Client
Agent communicates with the server locally
Network
Get message
Response message
Mobile agents communication model
Traditional client/ server architecture
Server

Figure 1. Client-server model versus the mobile agent paradigm

3.1. Mobile agents in wireless networks

As previously discussed, mobile agents offer a set of additional features than traditional methods, which makes them ideal for a large number of applications, such as information retrieval [22], network management [23], e-commerce [24], intrusion detection [25], collaborative applications [26], and mobile computing [27]. Although, none of the proposed applications require the use of mobile agents [20], their usage can contribute to a simpler, and, more effective, development of these distributed applications [28]. Mobile, or wireless computing, is the most frequently proposed application area of mobile agent technology, due to two important features [29]:

- **Task continuation**. An agent can autonomously migrate to a server and continue its processing task, while the user can disconnect from the network.
- **Minimal connection use**. The agent can pre-process information locally, at, either the server, or the mobile device, and thus reduce network overhead which would otherwise be required for large data transmissions.

Despite the enormous benefits of multi-hop ad-hoc networks, such as the ability of a user to access information in an *anytime-and-anywhere* manner, they have a number of inherent problems, including: channel unreliability; frequent disconnections; network protocol errors; limited and variable throughput; dynamic network topology; resource-constrained nodes; and ambiguous routing. Mobile agents are robust over unreliable network links, that is, they sense the connection status, and initiate their self-migration when the destination is reachable. In addition, they can dynamically create clones, delegate tasks, and, finally, dispatch them close to the resources, and thus benefit from parallel execution and local interactions. Furthermore, as shown in [30], mobile agents can perform intelligent filtering to gathered information, and thus reduce network load, a process which could be efficiently employed to reduce the size of gathered, redundant routing information.

4. Agent-based, metric-driven clustering architecture

An overview of the proposed cluster-head model is presented in Figure 2, which is based on a small-scale network topology, and depicts: the metric-driven clustering formation; the proactive network discovery mobile agent approach; the reactive route discovery stationary agent approach; and the data structures, which are maintained at each node.

As illustrated, the network is organised into four clusters, that is, cluster *A*, cluster *B*, cluster *C*, and cluster *D*, using the nodes' IDs, which are derived by the cluster-head metrics (*see* Section 4.1). Clusters *A* and *C* intersect, as nodes G_1 and G_2 are directly linked to the clusters' cluster-heads R_2 and R_3. Similarly, clusters *B* and *D* intersect, using the node G_3, and, clusters *C* and *D* intersect, using node G_4. Although clusters *A* and *B* do not intersect, as nodes DG_1 and DG_2, are not directly linked to both clusters' cluster-heads, the pair of nodes DG_1 and DG_2 can be used for inter-cluster routing, and it is thus called distributed gateway pair. The proposed clustering formation process ensures that each connected node belongs to at least one cluster, and that each cluster is linked to its adjacent clusters, as long as there is connectivity. In addition, as a result of the clustering formation algorithm, cluster-heads can be at either 2-hops, or 3-hops away, for example, R_1 is 2-hops away from R_4, whereas, R_1 is 3-hops away from R_2.

Each node runs an agency, which provides the execution environment for stationary and mobile agents, whereas, a cluster-head, in addition to the agency, runs a region registry, which provides a registration service for agencies, stationary and mobile agents, which exist in its cluster. The agency and region addresses are assigned by the mobile agent system, which are, typically, two *strings*, and they consist of: the communications protocol used; the node's IP address; the port number, which is different for agencies and regions; and the name of the agency/region. In addition, each node's agency produces an identifier, which is referred as the agency identifier, that is, a unique value in a distributed environment. Furthermore, each node has a node address, which is its IP address, normally, the IP of its wireless interface. There are numerous mobile agent systems, to date, however, only a few support resource-constrained devices, such as Grasshopper ME [31], which allows the execution of agencies, and regions on PDAs.

Each non-cluster-head node registers to its local cluster-head's region registry, and receives a ticket, where, once expired, the node assumes that it is no longer in direct communication range with the cluster-head. As an example (*see* Figure 2), member node M_4 has a node-ID of 85, which represents its ability to become a cluster-head, and ranges from zero (strongest) to 100 (weakest), whereas, it has a routing-metric of 78, which represents its routing ability. In addition, its node address is set to the IP address of its wireless interface, whereas the agency address incorporates the additional information that was previously mentioned.

Each node is required to broadcast its Neighbouring Node Table (NNT), which consists of the nodes, that is, the node address and agency/region addresses, a node can hear from, including itself, along with the respective node-IDs, routing-metrics, and roles. Then, each node builds a 2-hop Neighbouring Node Table (2-hop NNT), which is deduced from incoming NNTs, and consists of the 2-hop topology, from the viewpoint of the node, which provides location information about cluster-heads that are 2-hops away. For example, DG_1 can deduce from the NNT broadcasted by DG_2, that DG_2 is directly linked to R_2, and, also, that DG_2 is a distributed gateway, which can provide inter-cluster routing services for cluster *A* and *B*. However, R_1 does not know the existence of cluster *A*, yet, as this information cannot be deduced by the NNT broadcasted by DG_1, and, thus, DG_1 is required to broadcast its 2-hop NNT, so that cluster-head R_1 can learn about cluster *A*. Specifically, each distributed gateway is required to broadcast its 2-hop NNT, in addition to, its NNT, and thus cluster-heads can obtain knowledge of each adjacent cluster-head, that is, cluster-heads which are 3-hops away. The information contained in 2-hop NNT is: the cluster-heads' node addresses; the cluster-heads' agency addresses; the distributed gateways' node addresses; the distributed gateways' agency addresses; and the respective node-IDs and routing-metrics.

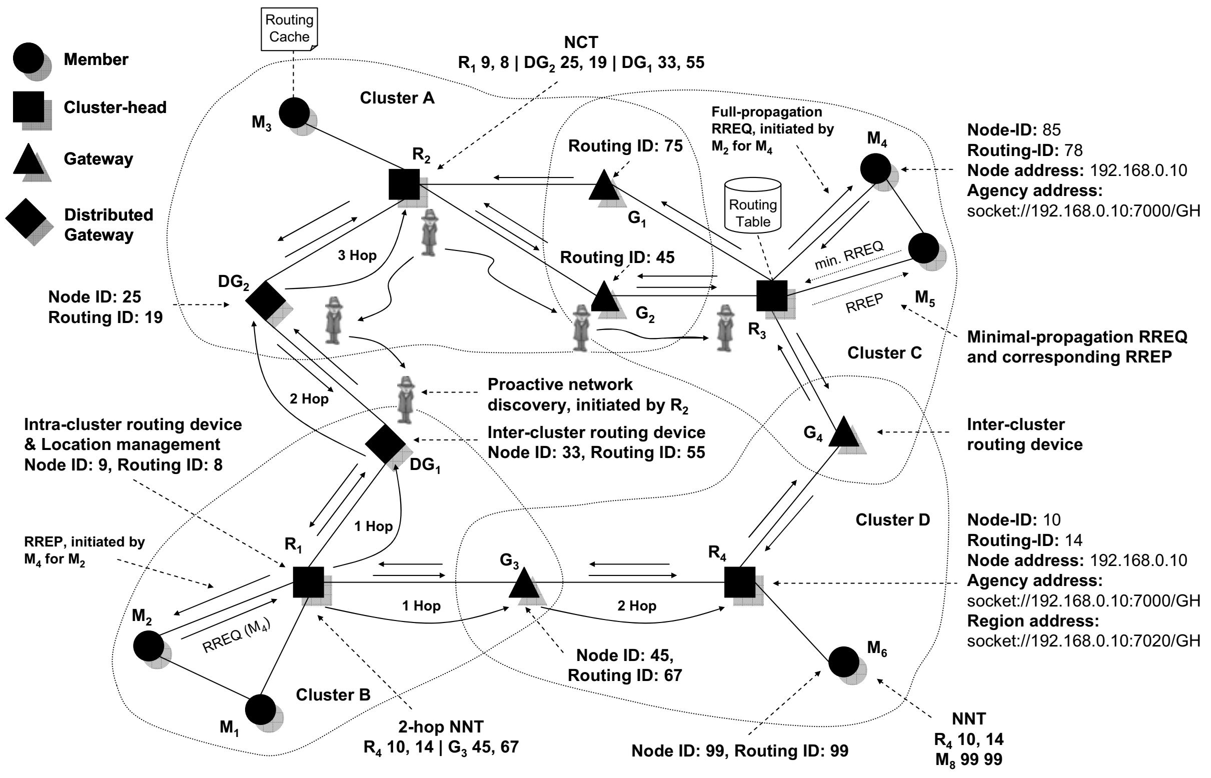
Member
Cluster-head
Gateway
Distributed Gateway
Routing Cache
NCT
R1 9, 8 | DG2 25, 19 | DG1 33, 55
Cluster A
M3
R2
Routing ID: 75
G1
Full-propagation RREQ, initiated by M2 for M4
Routing Table
M4
Node-ID: 85
Routing-ID: 78
Node address: 192.168.0.10
Agency address:
socket://192.168.0.10:7000/GH
min. RREQ
RREP
M5
3 Hop
Routing ID: 45
DG2
Node ID: 25
Routing ID: 19
G2
R3
Minimal-propagation RREQ and corresponding RREP
Cluster C
2 Hop
Proactive network discovery, initiated by R2
Intra-cluster routing device & Location management
Node ID: 9, Routing ID: 8
Inter-cluster routing device
Node ID: 33, Routing ID: 55
DG1
G4
Inter-cluster routing device
1 Hop
Cluster D
Node-ID: 10
Routing-ID: 14
Node address: 192.168.0.10
Agency address:
socket://192.168.0.10:7000/GH
Region address:
socket://192.168.0.10:7020/GH
RREP, initiated by M4 for M2
R1
G3
R4
M2
RREQ (M4)
1 Hop
2 Hop
Node ID: 45,
Routing ID: 67
M6
M1
Cluster B
2-hop NNT
R4 10, 14 | G3 45, 67
Node ID: 99, Routing ID: 99
NNT
R4 10, 14
M8 99 99

Figure 2. An overview of the agent-based, metric-driven model

As previously mentioned, the proposed hybrid method incorporates an on-demand route discovery process, which is based on stationary agents, and a proactive network topology gathering process, which is based on mobile agents. Both approaches coexist, and execute, in parallel, and thus mobile agents gather the network topology asynchronously, and provide the routing information to their cluster-heads, whereas, a node that requires a route, assuming that the route does not exist in its cluster-head's routing table, can independently initiate a route-request. As an example of the reactive approach, M_5 requests a route to M_3 from its cluster-head R_3, and the cluster-head responds back and provides the routing information, whereas, M_2 requires a route to M_4, but its cluster-head R_1 does not have this information, and thus, M_2 initiates a full-scale route-request. As shown in Figure 2, the route-request traverses only key nodes, such as cluster-heads, gateways, and distributed gateway pairs, and, in addition, it travels through all possible paths, which excludes the paths that involve multiple gateways. For example, the route-request travelled from R_2 to R_3 through G_2 only, as there is no need for the route-requests to traverse all possible paths, as this is left to the corresponding route-replies. Furthermore, the route-request was routed through G_2 and not G_1, as G_2 has a lower metric than G_1, and thus it provides a more reliable path. M_4 creates and transmits a route-reply for each incoming route-request, and specifies the reverse route that each route-request took, which consists of the cluster-head list only, as the gateway and distributed gateway information is not required, as each cluster-head knows its adjacent cluster-heads and the way to reach them. The route-replies gather the complete routing information, which includes all possible routes to M_4, as well as, the routing-metrics of each node along these routes, and, thus, M_4 can determine the most optimal routing path, in relation to the type of traffic it requires to transmit.

The mobile agents propagate in a similar manner to route-requests, that is, they visit each key node in the network by cloning themselves, until a dead-end is reached, that is, each key node has been previously visited, or, until the horizon is reached, that is, the maximum number of hops defined by the cluster-head which initiated the proactive network discovery. Then the mobile agents return back to their originating mobile agent systems, where they submit the gathered routing information to their parents, that is, each node's NNT, and then kill themselves. The mobile agents gather this information from the cluster-heads they visit, and before self-migration they intelligently filter relevant information, so as to reduce the size, in a similar manner as to the one described in [30].

4.1. Cluster-head metric

As previously discussed, the cluster-head metric incorporates various key factors which represent the ability of a node to undertake cluster-head responsibilities. These include: intensive algorithmic calculations; buffering, throughput, network error state, CPU utilisation, memory usage, battery reserves, and mobility patterns. Factors such as nodal utilisation, battery reserves, and mobility are especially important in cluster-head selection, as they can significantly influence the cluster stability and reliability, and thus more weighting is applied to them than the remaining factors. Nevertheless, throughput, buffering, and network error state are also important, as a cluster-head is often required to perform intra-cluster routing for its members. The weighting system has been initially derived through experimentation, and was then verified through simulations, which demonstrated that it provides a good configuration. As previously discussed, mobility information is assumed to be directly fed into the cluster-head metric from MOBIC, while the values of the remaining factors are retrieved as presented in [2].

The cluster-head metric is thus variable, as it relates to changing parameters, such as battery reserves, and so on. With this, it is possible that two devices can calculate the same node-ID, as a result of very similar hardware characteristics and mobility patterns. However, the lowest-ID algorithm in which the proposed clustering formation is based, strictly forbids duplicate values. In order to guarantee that each device will calculate a fair and unique node-ID, a function which generates a unique ID from a non-unique value is used. The most attractive approach is to multiplex the node's MAC address, which is guaranteed to be unique for each device, with the calculated node-ID, in such a way, so as to guarantee the node-ID uniqueness and most importantly leave the actual value virtually unmodified. An alternative approach would be to allow nodes with the same node-ID to re-execute the election process, as many times as necessary, so as to eliminate the duplicate values. This is likely to be dissolved rapidly since a node-ID is linked to various frequently changing factors. However, this approach may result in increased delays, network overhead, and implementation of coordination efforts.

4.2. Adaptability

The cluster-head metric is based on standard performance tests which are executed in advance, as well as on varying parameters, such as mobility and battery reserves. Thus, the resulting metrics are not fixed, and may considerably vary as these parameters change. Specifically, a point to infinitive value (∞) is defined, which is allocated to key parameters that have reached a critical level. For instance, in case that the battery level of a node drops below a certain threshold, the battery parameter points to infinitive, which results in the overall cluster-head metric to point to infinitive as well. Thus, in case that a cluster-head undertakes a critical change, and, consequently the cluster-head metric points to infinitive, the device discards its role and moves to *undecided* state. In this manner, network nodes are protected from running out of vital resources, and thus the routing protocol dynamically adapts to devices' critical changes.

Another example of the protocol's dynamic adaptation can be found in the proactive network discovery process. Accordingly, mobile agents that are using the underlying topological information to propagate, with purpose to retrieve routing information for the defined horizon, can dynamically alter their itinerary in case of the destination being unreachable, after waiting for a predefined amount of time. This problem could occur in situations where the ad-hoc network undertakes various frequent changes, such as cluster reorganisation, nodal movements, and so on. To resolve this, a mobile agent could dynamically find an alternative path by examining the currently visited node's topology information, such as the node's NNT, 2-hop NNT, and NCT. If the agent succeeds, it dynamically alters its previously stale routing information with the new one, otherwise it kills itself. There is also scope to allow mobile agents to initiate reactive route discovery processes in order to find a viable routing path that leads to the desired destination. However, it may be omitted because of concerns that this could lead to increased network overhead, especially in situations where the ad-hoc network is highly mobile.

4.3. Reconfigurability

As every entity in the proposed method is deployed as either a stationary or a mobile agent, reconfigurability can be achieved by external on-demand reconfiguration agents. Specifically, authenticated mobile agents carrying protocol updates in their payloads can be dynamically dispatched in the ad-hoc network in order to automatically update or replace certain protocol components, or to reconfigure the protocol in such a way so as to adapt in various environmental changes.

The main advantage of this strategy is that it eliminates the need of manual update installations. Particularly, in a routing protocol implemented with traditional mechanisms, one would have to gather every mobile device in the network and update it separately, while the devices should have to be cut-off from the network. This approach may not seem particularly difficult when dealing with a small-scale ad-hoc network, possibly consisting of a few dozens of devices, however, in a large-scale network with possibly a couple of hundred or even thousand of devices this is unrealistic. The proposed agent-based method provides the fundamental mechanisms to achieve efficient and effective reconfiguration, without needing to shut down the network, or, even reboot a single device. This lies in the agent's inherent characteristics, such as ease of installation and removal, agent communication, and mobility.

As an example, assuming that, a recently developed mobile agent is dispatched to replace an existing stale clustering agent. The agent is dispatched to the nearest node, and finds its way to the cluster-head. Once the agent arrives, it clones itself and leaves for the next cluster-head. The cloned agent delivers the credentials of its developer and its version number to the guard stationary agent. The guard agent then verifies the authenticity of the mobile agent and compares its version to the version of the current clustering agent. If the authentication is successful and the version number is later than the existing one, the guard agent removes the previous clustering agent, and installs the newer one. In this fashion, the up-to-date mobile agent could visit every single cluster in the network, and leave its clone. Updates are then disseminated from the cluster-head towards its members.

Another example is a mobile agent which carries an update towards the initiation frequency of network discovery mobile agents, as well as the horizon that these agents must propagate. For instance, assuming that, by default, each cluster-head initiates such a process every fifteen seconds, and the horizon is defined as the complete ad-hoc network. However, for certain scenarios, such as in highly mobile ad-hoc networks, these measurements may be inefficient, as they may result in increased number of hops, taken by the network discovery agents, and, consequently, to increased network overhead. Thus, the reconfiguration mobile agent could instruct each cluster-head to delay the initiation of the network discovery process, or, even discard it until instructed otherwise.

4. Conclusions

This paper presented the overall model of the proposed metric-driven, cluster-head organisation for hybrid multi-hop routing, which utilises stationary and mobile agents. As shown in the model, the stationary agents are responsible for the on-demand nature of the protocol, as they can, typically, rapidly react on route demands, while, the mobile agents are responsible for the proactive nature, as they are, typically, slower in retrieving routing information, because of migration, and serialisation time requirements. Most importantly, this paper demonstrated the use of adaptive metrics, which are used for clustering formation and data routing. The metrics incorporate various key factors, such as nodal mobility, utilisation, network state, throughput, and battery reserves. Accordingly, fitter nodes are chosen to form the network backbone, which is responsible for routing and location management, whereas, weaker nodes have no such responsibilities. With this, scalability is increased, as opposed to flat-structured routing protocols.

We are currently evaluating the performance of the metric-driven, clustering method and its routing behaviour using various performance metrics. These include: cluster-head changes, gateway and distributed gateway changes, cluster size, and connectivity.

References

[1] Migas, N., et. al., 2005. Metric Evaluation of Embedded Java-Based Proxies on Handheld Devices in Cluster-Based Ad Hoc Routing. 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05). Washington D.C., USA. pp. 147-154.

[2] Buchanan, W. J., et. al., 2004. Analysis of an Agent-based Metric-Driven for Ah-hoc, On-Demand Routing. Ad Hoc Networks. In Press, Corrected Proof. Available online 3 July 2004.

[3] McQuillan, J. M., et. al., 1980. The New Routing Algorithm for the ARPANET. IEEE Transactions on Communications. Vol. 28. No. 5. pp. 711-719.

[4] Bertsekas, D. and Gallager, R., 1987. Data Networks. Prentice-Hall, Inc. pp. 297-333.

[5] Schwartz, M. and Stern, T. E, 1980. Routing Techniques Used in Computer Communication Networks. IEEE Transactions on Communications. Vol. 28. No. 4. pp. 539-552.

[6] Perkins, C. and Bhagwat, P., 1994. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Proceedings of the ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications. London, UK. pp. 234-244.

[7] Abolhasan, M., et. al., 2004. A review of routing protocols for mobile ad hoc networks. Ad hoc Networks. Vol. 2. pp. 1-22.

[8] Gerla, M. and Tsai, J. T.-C., 1995. Multicluster, mobile, multimedia radio network. Wireless Networks. Vol. 1. No. 3. pp. 255-265.

[9] Jiang, M., et. al., 2001. Cluster Based Routing Protocol (CBRP). Internet-Draft. draft-ietf-manet-cbrp-spec-01. Work in progress.

[10] Perkins, C. E., et. al., 2003. Ad hoc On-Demand Distance Vector (AODV) Routing. Internet Draft. draft-ietf-manet-aodv-13.txt. Work in progress.

[11] Chiang, C.-C., et. al., 1997. Routing in Clustered Multi-hop, Mobile Wireless Networks with Fading Channel. Proceedings of IEEE Singapore International Conference on Networks (SICON). Singapore. pp. 197-211.

[12] Ephremides, A., et. al., 1987. A design concept for reliable mobile radio networks with frequency hopping signalling. Proceedings of IEEE. Vo. 75. No. 1. pp.56-73.

[13] Parekh, A. K., 1994. Selecting routers in ad-hoc wireless networks. Proceedings of the SBT/IEEE International Telecommunications Symposium.

[14] Basu, P., et. al., 2001. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. In International Workshop on Wireless Networks and Mobile Computing (WNMC2001). Scottsdale, Arizona, USA. pp. 16-19.

[15] White, J., 1997. Telescript technology: An introduction to the language. General Magic Inc. White paper.

[16] Hayes-Roth, B., 1995. An Architecture for Adaptive Intelligent Systems. Artificial Intelligence: Special Issue on Agents and Interactivity. Vol. 72. pp. 329-365.

[17] Wooldridge, M. J. and Jennings, N. R., 1995. Agent Theories, Architectures, and Languages: a Survey. Intelligent Agents; Workshop on Agent Theories, Architectures, and Languages. Berlin, Germany. pp. 1-39.

[18] Tripathi, A. R., et. al., 2000. Experiences and Future Challenges in Mobile Agent Programming. Microprocessors and Microsystems. Vol. 25, No. 2. pp. 121-129.

[19] Karjoth, G. and Posegga, J., 2000. Mobile Agents and Telcos' Nightmares. Annales des Telecomunication, special issue on communications security. Vol. 55. No. 7. pp. 29-41.

[20] Harrison, C. G., et. al., 1995. Mobile Agents: Are they a good idea. Technical Report. IBM T.J. Watson Research Centre. New York, USA.

[21] Lange, D. B. and Oshima, M. 1999. Seven good reasons for mobile agents. Communications of the ACM. Vol. 42, No. 3. pp 88-89.

[22] Cardi, G., et. al., 2000. Agents for information retrieval: Issues of mobility and coordination. Journal of mobility and coordination. Vol. 46, No. 15. pp. 1419-1433.

[23] Gavalas, D., et. al., 2001. Mobile software agents for decentralised network and systems management. Microprocessors-and-Microsystems. Vol. 25, No. 2. pp. 101-109.

[24] Tianfield, H., 2001. Enterprise Federation and Its Multiagent Modelization. E-Commerce Agents, Marketplace Solutions, Security Issues, and Supply and Demand. pp. 295--322.

[25] Zhang, R., et. al., 2001. Multi-agent Based Intrusion Detection Architecture. Proceedings of 2001 International Conference on Computer Networks and Mobile Computing, IEEE Computer Society. Los Alamitos, USA. pp. 494-501.

[26] Wong, D., et. al., 1997. Concordia: An Infrastructure for Collaborating Mobile Agents. In Proceedings of the first International Workshop on Mobile Agents (MA97). Berlin, Germany. pp. 86-97.

[27] Migas, N., et. al., 2003. Mobile Agents for Routing, Topology Discovery, and Automatic Network Reconfiguration in Ad-Hoc Networks. 10th IEEE International Conference and Workshop on the Engineering of Computer Based Systems. Huntsville, USA, pp. 200-206.

[28] Puliafito, A., et. al., 2000. MAP: Design and implementation of a mobile agents' platform. Journal of Systems Architecture. Vol. 46, No. 2. pp.145-62.

[29] Kotz, D., et. al., 1997. Agent TCL: Targeting the needs of Mobile Computers. IEEE Internet Computing. Vol. 1, No. 4. pp. 58-67.

[30] Migas, N., et. al., 2004. Migration of Mobile Agents in Ad-hoc, Wireless Networks. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04). Brno, Czech Republic. pp. 530-535.

[31] IKV++, Inc., 2003. Grasshopper mobile agent system. Grasshopper Documentation. Available from <http://www.grasshopper.de>. Last visited 21/07/2005.