Hierarchically-organized, multihop mobile wireless networks for quality-of-service support

Ram Ramanathan Martha Steenstrup[1]

Advanced Networking Department
BBN Systems and Technologies
Cambridge, MA 02138

Abstract. MMWN is a modular system of link- and network-layer algorithms that enables a multihop mobile wireless network to support distributed, real-time multimedia applications. In this paper, we describe three key components of this system: the clustering procedures for defining a virtual, hierarchical control structure superimposed on a large network of mobile switches and endpoints; the location management procedures for determining the current locations of mobile endpoints relative to the hierarchical control structure; and the virtual circuit management procedures for setting up and repairing virtual circuits as switches and endpoints move. We also provide performance results, obtained through simulation and analysis, which show the robustness of each of these components with respect to a broad spectrum of transmission ranges and relative mobility of switches and endpoints.

1 Introduction

The evolution of mobile computing and communications technology has been shaped predominantly by the challenges posed by user mobility and wireless connectivity to the network. In recent years, quality-of-service provision to support distributed, real-time multimedia applications for mobile users has emerged as an area of major technological focus (e.g., see [1]-[4]). Most of the solutions proposed thus far rely on two key assumptions about the network infrastructure. First, the network infrastructure has predetermined interconnectivity, excluding failure of network components, i.e., infrastructure nodes either remain stationary or move in a predictable fashion (e.g., as in a low-earth-orbit (LEO) satellite constellation). Second, the network infrastructure has sufficient capacity to satisfy virtually all traffic demands for network resources. These two assumptions serve to push quality-of-service

[1] This work was funded in part by the Defense Advanced Research Projects Agency (DARPA) under contract DAAB07-95-C-D156, as part of the Global Mobile Information Systems (GloMo) program.

issues toward the periphery of the network, specifically the wireless hop between the network infrastructure and the mobile user. While these assumptions might be reasonable in the contexts of cellular telephony or mobile internetworking, they are not valid for multihop mobile wireless networks. In such networks, there is no fixed, wired infrastructure. All nodes, including switches as well as endpoints (possessing no switching capabilities), communicate via wireless links and move along trajectories that are not necessarily planned in advance. The unpredictable movement of switches implies that node interconnectivity and link properties (e.g., capacity, error rate) cannot be predetermined. Therefore, in multihop mobile wireless networks, quality-of-service issues cannot be confined to the last hop but instead pervade the entire network.

Multihop mobile wireless networks are a practical approach to providing low-cost, rapidly-deployable, self-organizing networks, necessary for communications in situations where no infrastructure yet exists or where the existing infrastructure has been severely damaged. Use of this type of network includes but is not limited to communications for tactical maneuvering and strategic planning on the battlefield, for emergency relief in an area afflicted by a natural disaster, and for field studies conducted by a team of scientists in a remote location. In many of these situations, timely and accurate receipt of multiple types of information transmitted across the network can mean the difference between life and death. For example, in the battlefield, situational awareness information distributed among combatants can be used to pinpoint friendly and hostile forces; in a disaster area, patient vital signs and medical images transmitted from the field to a hospital can be used for remote diagnosis and rapid treatment of injuries. To be effective in these critical situations, the network must be able to support applications that require real-time interactions, many-to-many delivery, and transport of voice, video, images, and other forms of data. Many of these applications demand high throughput and have low tolerance for delay, jitter, loss, and corruption of information.

In this paper, we address the problem of quality-of-service provision in multihop mobile wireless networks. A network's ability to provide a specified quality of service between a set of endpoints depends upon the inherent performance properties (e.g., delay, throughput, loss rate, error rate) of the links and nodes, the traffic load within the network, and the control algorithms operating at different layers of the network. Wireless transmissions may be adversely affected by the distance between the ends of the link, obstacles in the environment, externally generated noise, or interference caused by other transmissions, and hence the quality (e.g., capacity, signal-to-noise ratio) of a wireless link is apt to be highly variable. Moreover, the environmental conditions influencing the quality of a wireless link at a particular point in time are not likely to be known completely or in advance. Thus, the unpredictability of link quality and switch and endpoint mobility makes the problem of quality-of-service provision difficult.

A comprehensive approach to quality-of-service provision in multihop mobile wireless networks involves adaptive control algorithms operating at many layers, from the transceivers to the applications.

Even in simple situations, control of quality of service may be required at multiple layers. For example, in a network with stationary nodes and with no traffic except for a single session between a pair of endpoints, it might be the case that none of the existing paths between the two endpoints satisfies the session's service requirements. To obtain its desired quality of service, the application might have to adapt its traffic (e.g., through compression or hierarchical encoding of information) to the quality of service that the network is capable of providing. The goal of a multi-layered approach to quality-of-service provision is to control the quality of service as perceived by the next higher layer, as environmental conditions change. Thus, adaptation begins at the lowest layers and progresses to the higher layers only when the lower layers can no longer maintain the quality of service at the desired level. This approach localizes , within the network and within the layers (to the extent possible), the effects of environmental state changes, and hence limits the quantity of network resources necessary to maintain quality of service. Moreover, fast adaptation at the lower layers reduces the amount of time that higher layers perceive degraded quality of service as a result of unfavorable changes in the environment.

Our work centers on quality-of-service control at the link and network layers. We have designed a modular system of distributed, autonomously adaptive algorithms that cooperate to support distributed, real-time multimedia applications in large, multihop mobile wireless networks. These algorithms include adaptive control of link quality; hierarchical organization of a control structure reflecting endpoint location and network connectivity and services at multiple levels of abstraction; quality-of-service route selection; and autonomous repair of multipoint virtual circuits with resource reservations. This system is called MMWN, an acronym for multimedia support for mobile wireless networks, and will be referred to as such throughout the rest of the paper. The MMWN system has been designed to support a variety of applications on a variety of radios and presupposes the existence of generic interfaces for the exchange of information with the radio and with the upper layers. It does not, however, directly address quality-of-service issues related to radio hardware design, transport protocols or applications, or interconnection of multiple heterogeneous networks. Moreover, the MMWN system is not currently designed to operate effectively in a network comprised of highly mobile nodes, i.e., endpoints that change network attachment points (or switches that change neighbor connectivity) faster than the network can detect these changes. In such a network, accurate prediction of future state is necessary for the network control algorithms to keep pace with the rapid and frequent state changes. Solutions to the network control problems associated with highly mobile nodes are beyond the scope of this paper. Finally, power conservation and security (including authentication of information, limited probability of detection, and resistance to jamming) are key concerns in multihop mobile wireless networks. While these considerations were factored into our design, they were not the primary objectives, and hence are not treated further in this paper.

In designing the MMWN system, we have built upon previous work produced by the DARPA packet-radio programs [5]-[7], particularly the SURAN packet-radio project [8]. This prior work tackled the problem of session survivability in large, military, mobile packet-radio networks, through adaptation at the link and network layers and through efficient use of scarce network resources. The problems of quality-of-service provision [9]-[11] and multipoint delivery (other than broadcast) were secondary concerns. More recently, several approaches have been proposed that address various aspects of quality-of-service support in multihop mobile wireless networks. These include efficient channel access protocols [12, 13], route selection improving reliability or capacity [14]-[16], and fast and flexible resource reservation mechanisms [17, 18]. With the exception of the MMWN system, we are aware of no other approaches that attempt to solve the general problem of quality-of-service provision (with multipoint delivery) in multihop mobile wireless networks. The complete MMWN system design is described in detail in [20]. We are currently investigating the behavior of this system and its individual components, in a variety of situations, through large-scale simulation and through implementation and experimentation with a reduced version in a mobile radio testbed. In the remainder of this section, we briefly describe the main features of the MMWN system.

The MMWN system enables the network nodes to organize themselves into a hierarchical control structure that combines elements derived from both cellular and packet-radio networks. Endpoints group themselves into "cells" around switches (called "cellheads"), such that each endpoint is within one hop of its affiliated switch. As endpoints move, they may affiliate themselves with different switches. Switches group themselves into "clusters", each of which functions as a multihop packet-radio network, and lower-level clusters may group themselves to form higher-level clusters. (The lowest-level cluster is a cell.) Two clusters are interconnected by "virtual gateways", each of which consists of one or more communicating switches in each of the two clusters. As switches move, clusters may autonomously split or coalesce, altering cluster and virtual gateway membership. This hierarchical control structure consisting of nested clusters permits a network to scale to millions of nodes, by confining network dynamics to the lowest-level of the hierarchy affected by them.

Each cluster contains a location manager that keeps track of the locations of endpoints within the cluster and that assists in locating endpoints inside and outside of the cluster. Location management permits one to trade off updating and paging overhead, on an individual endpoint basis, depending upon the endpoint's call-to-mobility ratio. Each endpoint has a "roaming level", specified with respect to the hierarchical control structure, which implicitly defines a "roaming cluster". An endpoint generates a location update only when it moves outside of its current roaming cluster. Propagation of a location update remains confined within the highest-level cluster in which an inter-cluster movement occurs. A mixture of hierarchical location queries and paging is used to locate an endpoint that has moved from its last known location.

Application performance is directly related to the quality of the network links over which the application traffic travels. To stabilize the quality of wireless links, the MMWN system provides support for adaptive link control whereby a node can adjust link parameters in response to perceived and desired link quality. The set of observable and controllable parameters and hence the degree of control possible depends upon the capabilities of the particular radios used by the nodes. Example controllable parameters might include transmit power level, forward-error-correction coding rate, chips per bit (in direct sequence spread spectrum radios), and transmissions per packet. Example observable parameters might include receive power level, bit error rate, and signal-to-noise ratio. In adjusting transmit power, one must trade off the projected improvements in the quality of the link against the estimated interference caused by increasing the power. Adaptive link control may be applied on a network-wide basis to maintain a uniform link quality throughout the network, or it may be employed on a virtual-circuit basis providing different link quality for different sessions.

The MMWN system distributes routing information in the form of link states, which contain connectivity and service information pertaining to clusters and virtual gateways at all levels within the hierarchical control structure. This approach to hierarchical link-state routing resembles that of [21] and [22] but also includes abstracted service information in addition to connectivity information as part of the link state. Link states provide more information than distance vectors about the state of the network and hence increase the chances that a route with a desired quality of service will be found by the route generation procedure. (See part II of [19] for a discussion of link-state versus distance-vector routing). Each cluster contains a QoS manager that computes the qualities of service advertised in the link-state information for the cluster. This characterization of quality of service is a function of the services provided by the lower-level component clusters, which ultimately depends upon the services provided by constituent nodes and links. In particular, the characterization depends on both the recently-used routes across the cluster and actual measurements of quality of service within the cluster. To stabilize the service characterization of a cluster in the presence of mobile nodes and wireless links, the QoS manager computes a statistical characterization of the cluster's qualities of service that reflects the mean values as well as the expected variation in these values. A cluster's QoS manager distributes link-state advertisements about the cluster both within the cluster and to all child clusters within the parent cluster. Thus, each switch receives link-state information about all clusters within its parent cluster, all clusters within its grandparent cluster, and in general all level-$n-1$ clusters within its level-n ancestral cluster. A switch attempts to compute one or more feasible routes for a session, based on the link-state information received from other clusters, the current location of the session endpoints with respect to the hierarchical control structure, and the service requirements for the session. Route selection accounts for the service constraints of the session and the network and favors minimum-hop routes through less mobile clusters, to conserve network resources and to reduce the number of changes in the route throughout the life of the session.

In a large mobile wireless network, differences in link-state databases maintained by two switches in the same cluster may often occur, not only as a result of the propagation delays experienced by link-state updates but also as a result of the frequent changes in link state occurring in this environment. Furthermore, such differences may persist for some time, resulting in inconsistent views among switches concerning which routes to use to reach to certain destinations. To prevent routing loops in this environment, the MMWN system provides route-directed forwarding, inserting routes in datagrams and establishing virtual circuits for stream-oriented sessions. Each route selected by a switch is specified in terms of the set of clusters through which the session's traffic will travel. This specification provides progressively less detail (i.e., is expressed terms of progressively higher-level clusters) the farther the route extends from the switch, reflecting the granularity of link-state information available to the switch for route generation. As a datagram travels along a route, switches along the way refine the route as necessary to reflect a more detailed route through their clusters. Each end-to-end virtual circuit is composed of lower-level virtual circuits, each of which corresponds to a portion of the route filled in during virtual circuit setup.

The MMWN system includes several mechanisms that enable nodes to adjust existing virtual circuits to compensate for node movement and link quality degradation. These include adaptive link control to "stretch" an existing link; local switch handoff to bypass a particular switch; rerouting of a portion of a virtual circuit; virtual circuit trees connecting to multiple possible locations of an endpoint; and multiple virtual circuits per session for fault tolerance and sufficient throughput. Moreover, to provide a session with its desired quality of service over a link, a switch has the capability to reserve resources and schedule traffic separately for each virtual circuit using the link. Finally, the MMWN system accommodates both sender-oriented and receiver-oriented multicast session establishment. Each cluster contains a multicast manager that controls access to multicast groups and sessions extant within the cluster and that quickly joins endpoints to existing sessions.

In the MMWN system, the network-layer services (e.g., location management, QoS characterization, route generation, and multicast group and session management) are provided by servers. Each node may (but need not) contain all server functions, many of which are computationally intensive or require rapid responses to frequent queries. Distributing server functionality to a small subset of network nodes reduces the power requirements for most nodes but also increases the network-layer services' susceptibility to disruption caused by movement of the servers. The MMWN system provides three mechanisms that increase the availability of network-layer services in the presence of moving servers. First, servers have generic service-specific addresses, and switches know how to direct traffic to the correct type of server based on its generic address. Second, some servers keep their current clients informed of their addresses as they move. Third, within a cluster, servers of the same type periodically exchange databases, immediately update new servers entering the cluster, and distribute their database to servers in their old cluster when they move to a new cluster (called a "service handoff").

In the remainder of the paper, following a brief overview of the proposed network architecture in section 2, we focus on three key components of the MMWN system. These are cluster formation in section 3, location management in section 4, and virtual circuit management in section 5. Section 6 presents the results of an analysis of the performance of these three components, with respect to their sensitivity to the transmission range and the degree of movement of the network nodes. We conclude with a brief summary and discussion of future work in section 7. The interested reader should consult [20] for detailed descriptions of all of the components of the MMWN system, including those not covered here.

2 Network architecture

We classify network nodes into *switches* and *endpoints*. Only switches can forward (or route) packets but both endpoints and switches can be sources of or destinations for packets. Switches as well as endpoints can be mobile.

An endpoint *affiliates* with a switch in order to connect to the network. At a given instant, an endpoint may be affiliated with no more than one switch. The set of endpoints affiliated with a switch comprises a *cell*, with the switch as a *cellhead*. If an endpoint moves out of range of its cellhead, it will attempt to reaffiliate with another switch. We note that, in our architecture, a cell does not in any way demarcate geographical boundaries.

As mentioned in the introductory section, our architecture may be perceived as a hybrid, or generalization, of two well-known mobile wireless network architectures – the cellular (or PCS) and the packet radio (or ad hoc) network architectures. The cellular architecture is a particular case of ours when switches are stationary, and the packet radio (or ad hoc) architecture is a particular case when there are no endpoints.

Switches are grouped, for the purposes of routing and location management, forming a hierarchical control structure consisting of *clusters*. A cluster is a set of switches, or recursively, a set of clusters. If a cluster C_2 is a member of a cluster C_1, then C_2 is a *child* of C_1 and C_1 is the *parent* of C_2. Every cluster has exactly one parent, and every switch belongs to exactly one cluster (i.e., clusters do not overlap). The *level* of a cluster C is the maximum, taken over all switches, of the number of parent relations a switch has to invoke to reach C. By definition, a switch and the endpoints affiliated with it form a level-0 cluster. At the highest level of the hierarchy, all clusters are contained in a *universal* cluster. An example clustering hierarchy is depicted in figure 1, along with an alternate representation as a tree. The switches s, x, q, y, and z are at level 0, clusters C, D, E, and F are at level 1, clusters A and B at level 2 and the universal cluster U at level 3. Although this example shows a balanced and binary tree, neither balanced nor binary is a requirement.

Every switch and endpoint is assumed to have a configured globally unique identifier, referred to as the *switch id* and the *endpoint id* respectively. Every cluster except the level-0 clusters (cells)

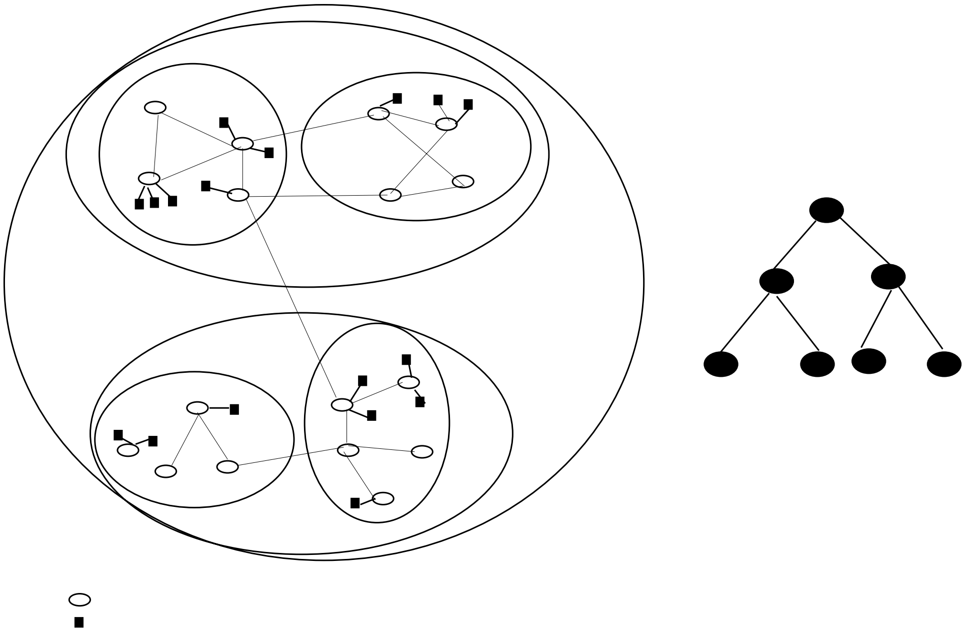


Figure 1: A virtual network hierarchy formed by clustering, pictured as nested clusters on the left and as a tree on the right (level-0 clusters are not shown in the tree). The hierarchy imposes a straightforward parent-child relationship between clusters.

acquires a *cluster id* unique among its siblings. By definition, the cluster id of a level-0 cluster is the corresponding switch id. The *address* of a cluster in relation to the hierarchy is the sequence of cluster ids C_1, C_2, ..., C_k, where cluster C_{i-1} is the parent of cluster C_i, $1 \leq i \leq k$. For example, if cluster C is a child of cluster B and cluster B is a child of cluster A, then the address of C is $A.B.C$. The address of a switch is the same as the address of the level-0 cluster it comprises. The address of an endpoint is the same as the address of the switch with which the endpoint is affiliated. Addresses are autonomously acquired and may change with time.

A switch may play two roles in relation to a cluster, *border* and *interior*. A border switch with respect to a cluster C is one that has a neighbor that is not in C. By definition, every switch is a border switch for its own level-0 cluster. A switch is an interior switch if it is not a border switch. A switch may be a border switch with respect to multiple ancestral clusters. For instance, in figure 1, switch x is a border switch with respect to B (adjacent to x) and with ancestor U (adjacent to z). The role of a switch may change during cluster reformation.

Connectivity between two adjacent clusters is provided through one or more *virtual gateways* (VGs). A virtual gateway is an aggregation of one or more *peer* border switch pairs, each of which consists of two border switches in different clusters. The border switches constituting a virtual gateway must be connected. All routes between two clusters must pass through a virtual gateway connecting these clusters. A border switch may participate in zero, one, or multiple virtual gateways, at the same or different levels. In figure 1, switches x, y, s, and q form a virtual gateway between clusters E and

F, and switches x and z form a virtual gateway between clusters A and B. Every virtual gateway acquires during the clustering process, a *VG id* unique among all virtual gateways in the parent cluster of the clusters that this virtual gateway connects.

Connectivity between two virtual gateways across a cluster is represented by a *virtual link* (VL). A virtual link represents a neighbor relation between two virtual gateways in the same cluster. It is an aggregation, recursively, of lower-level virtual or physical links and virtual gateways.

3 Autonomous clustering

The MMWN system employs autonomous clustering procedures to aggregate network nodes into logical units for control purposes, including routing and location management. The requirement for clustering, and therefore the nature of clustering, is different depending upon whether the nodes being clustered are endpoints or switches. For endpoints, clustering provides a switch that can act as an interface to the network infrastructure. Due to its resemblance to the well-known cellular architecture of wireless telephony, we call this type of clustering *cell formation*. For switches, clustering enables summarization of distant information and is crucial to the scalability of our system.

3.1 Cell formation

The primary objective of cell formation is to enable, insofar as connectivity permits, every endpoint to be *affiliated* with a switch. A secondary objective is to maintain an upper bound on the number of endpoints affiliated with a switch, in order to limit traffic concentration.

As a result of adaptive link control procedures (not described in this paper), an endpoint maintains a continually-updated list of switches with which it has acceptable-quality communication links. Each link has a *link quality vector*, which consists of values for link *attributes*, e.g., bandwidth $= 128$ kbps, bit-error-rate $= 10^{-4}$. An endpoint chooses its switch affiliation based on a function of the link quality vector, the particular function used being specific to the endpoint. This allows each endpoint to choose the nature of its affiliation link, depending upon its quality-of-service requirements.

When an endpoint is activated, it waits until the link quality vector for nearby switches is formed and executes an *affiliation protocol*. This consists of a handshake with each of the switches, in decreasing order of preference. The handshake consists of an affiliation request sent by the endpoint to a chosen switch, and a response sent by a switch if it agrees to accept the endpoint affiliation. A switch does not send a response if the number of endpoints already affiliated with it exceeds a configured limit. An affiliation handshake "fails" if either of the messages is garbled or the switch does not accept the endpoint. In such a case, the endpoint initiates another affiliation handshake with the next switch in its list. The affiliation protocol terminates successfully when an affiliation response is received from some switch. Otherwise the endpoint remains unaffiliated.

Due to mobility, the quality of the link between the endpoint and its switch may change continually. If the quality goes below a predetermined threshold, the endpoint starts afresh and executes the affiliation protocol as described above. We note that an endpoint need not be be affiliated to the "best" (e.g., closest) possible switch at all times, but rather to one that permits a link of acceptable quality. This design choice was made in order to contain control message exchanges and prevent thrashing of affiliations.

The cell formation mechanism in MMWN resembles the "handoff" mechanism in commercial cellular standards, but differs in significant respects. First, it is entirely endpoint initiated and controlled as opposed to "mobile assisted" (GSM, [23])) or "mobile independent" (AMPS, [24]). Second, it allows each endpoint to select the function of link attributes to use in judging link quality. Third, it allows for control on the cell size.

3.2 Hierarchical clustering

The clustering procedures autonomously group switches into clusters, clusters into superclusters and so on. This creates a hierarchical control structure that enables the MMWN system to scale to millions of mobile nodes, by confining network dynamics to the lowest level of the hierarchy affected.

A notable feature of our approach is the reliance upon and exploitation of link-state topological information available locally. (Refer to section 5.1 for a description the hierarchical link-state information used for routing within the MMWN system.) This link-state information is available "for free" to the clustering mechanism, that is, it is already collected for routing. Using this available information allows for quick convergence of the clustering algorithm – a capability difficult to accomplish using truly distributed algorithms. Moreover, as each switch has link-state information, it avoids single points of failure – a drawback of centralized algorithms. In the design of the clustering algorithm, we were influenced by the following objectives.

1. *Bound the size of each cluster.* The routing overhead (i.e., link-state distribution and route computation) depends significantly upon the number of switches in a cluster, or at higher levels, the number of child clusters in a cluster. When this number exceeds a certain threshold, splitting the cluster is likely to result in more efficient routing.

2. *Minimize the number of levels in the hierarchy, within the constraints of objective 1.* In general, the more levels there are, the more sub-optimal the routes become. Furthermore, more overhead is spent in maintaining the hierarchy. Note that this "flat-as-can-be" goal, combined with objective 1 implies that balance in size among clusters should be maximized.

3. *Minimize the volatility of inter-cluster connectivity, i.e., virtual gateways must be "stable".* Previous studies of hierarchical routing [22] have shown that the overhead of link-state distribution increases much more rapidly with the frequency of inter-cluster connectivity changes than with the frequency of intra-cluster connectivity changes. Therefore, in our mechanism, each virtual gateway is a *maximally* connected set of border switch pairs. Thus, connectivity changes within the virtual gateway do not necessitate the triggering of link-state updates.

We now describe the clustering mechanisms governing cluster and virtual gateway dynamics in more detail. For ease of understanding, our discussion is restricted to a 2-level hierarchy.

3.2.1 Cluster dynamics

Events governing cluster dynamics include switch movement, switch births (new switches), and switch deaths (e.g., failure, destruction etc.). The reaction of the clustering algorithm to these events can be one of four: a cluster is born, a cluster splits into two, two clusters merge, or cluster membership changes.

Cluster splitting and merging are controlled by three configured parameters: N_{split}, the splitting threshold; N_{merge}, the merging threshold; and N_{pref}, the preferred size for a cluster. A cluster splits into two if the number of switches in the cluster becomes greater than N_{split}, or if a *partition* is detected. A cluster (or virtual gateway) is said to be partitioned if there exists at least one pair of switches such that there is no path from one switch to the other through switches within the cluster. Switches comprising each resultant cluster form a connected network, and there is at least one virtual gateway between the clusters. Within these constraints, the algorithm attempts to balance the sizes of the component clusters in a best-effort manner. A cluster may merge with an adjacent cluster if the number of switches becomes less than N_{merge}. If and when there is a merge, an attempt is made to bring the resultant cluster size as close to N_{pref} as possible. A switch may change cluster membership if it finds itself without neighbors in its own cluster.

Switches in a cluster are ranked by order of increasing switch ids. This ranking is done in a fully decentralized fashion, independently by each switch using the link-state information collected. The highest-ranked switch in a cluster (i.e, the switch with the lowest id) initially assumes responsibility as the *cluster leader*. Cluster splitting or merging is then executed only by the leader and the result (e.g., where to split, who to merge with etc.) is sent to (the appropriate subset of) the rest of the switches. A timeout mechanism is used so that if nothing is heard from a switch of rank r, the switch with rank r-1 assumes responsibility as the *backup* cluster leader.

Cluster splitting. Periodically, a leader (or a backup), S, checks to see if the number of switches in its cluster has exceeded the splitting threshold, N_{split}, and if so initiates a split. First, S runs a splitting heuristic on the cluster topology (available from the local link-state database), partitioning it into two disjoint clusters. Then, it broadcasts information about the new cluster membership and virtual gateway membership to every switch in the cluster. Each switch receiving this information updates its cluster-related state appropriately. Virtual gateways between the two resultant clusters are born automatically as a result of this cluster splitting.

The splitting heuristic should maintain connectivity between switches in each resultant cluster, and attempt to equalize the sizes of the two clusters. Although the literature is replete with graph partitioning algorithms, there is none that targets the criteria that we are interested in. Therefore, we have designed our own partitioning algorithm that works as follows. Given the topology graph G_c = (V_c, E_c) of a cluster C, we pick a random vertex x in the graph. Then, we pick a vertex u that is maximum number of hops away from x, and then a vertex v that is a maximum number of hops away from u. The idea is to pick "seed" vertices that are as far away from each other as possible without running a all-pairs shortest path algorithm which is potentially $O(n^3)$ in time. We begin the splitting by initializing sets S_1 and S_2 to u and v respectively. The sets S_1 and S_2 are grown one vertex at a time, in an alternating manner, in such a way that S_1 and S_2 are always connected graphs. That is, first a vertex not in S_1 or S_2, but adjacent to some vertex in S_1 is sought for S_1. If such a vertex is found, it is added to S_1, otherwise S_1 remains as it is. Next, a similar addition is done for S_2, and then again for S_1 and so on until G_c is partitioned into S_1 and S_2. It is easy to see that the algorithm converges (terminates), producing two clusters, each connected. Experiments with a number of randomly generated graphs have shown that the heuristic produces excellent "balance", i.e., $|S_1| - |S_2|$ is very small, and in over 90 % cases 0 or 1.

Cluster merging. Periodically, a leader (or backup), S, checks to see if the number of switches in its cluster, C, is less than the merging threshold N_{merge}, and if so, initiates a merge. The merging procedure finds the most appropriate neighboring cluster to merge with, based on size. This requires that the switch initiating the merge know the sizes of adjacent clusters and is accomplished by including the cluster size information as part of the link-state updates. The algorithm seeks to find an adjacent cluster D at the same level as C such that the sum of the sizes of C and D (i.e., the size after possible merging) is as close to the preferred size N_{pref} as possible, and not greater than N_{split}. The result of the algorithm is either such a cluster, or an indication that no such D exists. If a candidate cluster is found, then the result is conveyed to all involved switches which update their cluster membership and role.

3.2.2 Virtual gateway dynamics

As mentioned in section 2, a virtual gateway between two clusters is a connected set of border switches in the two clusters. When clusters split or merge, or when switches move, virtual gateways may be created, destroyed, grow, shrink, split, or merge. Creation and dissolution of virtual gateways as a result of cluster splitting and merging was addressed in section 3.2.1. In this section, we consider the virtual gateway dynamics due to switch movements.

Figure 2(a)-(d) illustrates virtual gateway creation, growth, merging and splitting. A new virtual gateway between two clusters may be created when an interior switch in one cluster becomes a neighbor of an interior switch in the other (thereby making both border switches). A virtual gateway may "grow" when an interior switch in either of the clusters becomes a neighbor of a border switch in the virtual gateway. Two virtual gateways may merge if a switch in one becomes adjacent to a switch in another, and the two are peer border switches. Finally, if a virtual gateway becomes disconnected, it will split into two or more virtual gateways.

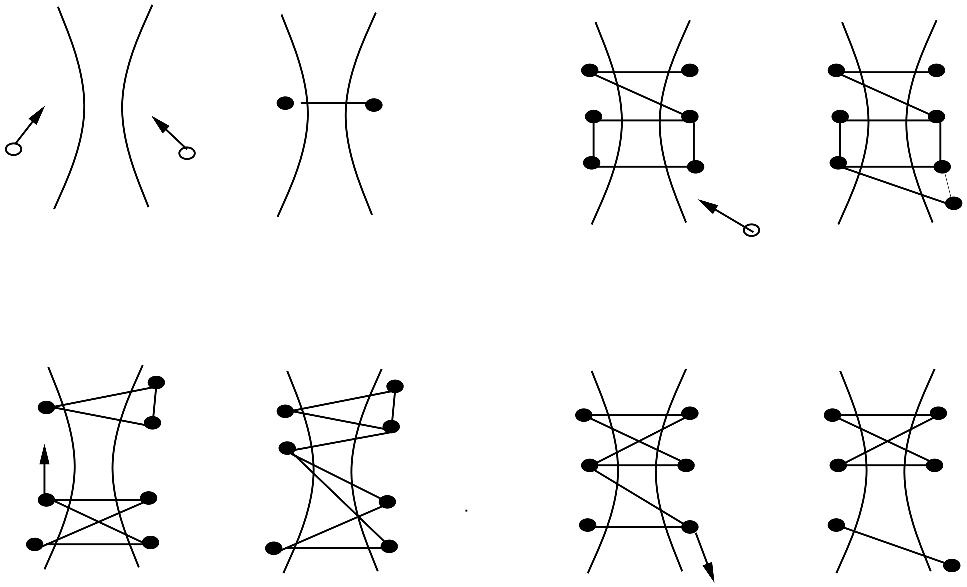


Figure 2: Virtual gateway creation, joining, merging, and splitting. Dark circles indicate border switches; light ones indicate interior switches. The arrow indicates the direction of movement of the node effecting the change.

Virtual gateway dynamics are controlled by a distributed protocol involving a handshake between two switches involved in the change. Periodically, say every T_{vg} seconds, a switch checks its state, and its neighbor's state and decides whether or not to initiate the virtual gateway protocol. The state consists of the switch id, role (border or interior) and VG id, if applicable. For instance, consider figure 2(a) where switch 4 becomes adjacent to switch 6. Each switch checks its state and its neighbor's state and finds that they are both borders, and therefore could form a new virtual gateway together. The lower-id switch, namely switch 4, initiates a request-response handshake if it wishes to form a virtual gateway. After the handshake, which includes the selection of a VG id, is complete, a new virtual gateway is born. We note that a switch may choose not to initiate a request or not to respond if it does not wish to form a virtual gateway, e.g., if it is highly mobile or if there are already too many virtual gateways in the cluster. Virtual gateway growth, merging and splitting happen in a similar manner – details can be found in [20].

Note that the period, T_{vg}, between state checks determines the sensitivity of the protocol to changes. Thus, in the worst case, an action (such as virtual gateway formation) may not be initiated until T_{vg} seconds after the event. By using a small value of T_{vg}, we can reduce the reaction time to state changes as desired, while incurring only a slightly increased processor load for running the checks. Furthermore, in a highly mobile network, we may *not* want to react to every state change. This can be accomplished by setting T_{vg} appropriately high.

4 Location management

Location management in MMWN involves the maintenance of a dynamic distributed database whose entries are *associations* between endpoint ids and addresses. An entry in this database changes if and only if one of the following three events occurs:

1. An endpoint reaffiliates with another switch, thereby changing its address. This may involve a change in any suffix of its address. For example, endpoint q in figure 3 has an address of $U.A.C.R$. If q reaffiliates with a switch V, then suffix R will change to V; if q moves to switch T in cluster D, suffix $C.R$ will change to $D.T$.

2. The endpoint's cellhead moves to a different cluster, but the endpoint continues to be affiliated with the cellhead. Here, although the endpoint does not change cells, its address changes because its switch's address changes.

3. Reformation of an ancestral cluster occurs, that is, an ancestral cluster splits into two or merges with another cluster. Since the cluster address itself may change, there may be a change in the endpoint address.

In order to provide location management, each cluster (including clusters at levels greater than zero) has a *location manager* (LM). The location manager for a cluster is a switch, elected from among all of the switches within the cluster, with the lowest id. A switch can perform the location management functionality at multiple levels. For example, in figure 1, there is a location manager in cluster F that is a location manager for both F and B.

We divide location management functionality into two components: *location updating*, (i.e., updating the distributed location database with the new association between the endpoint id and

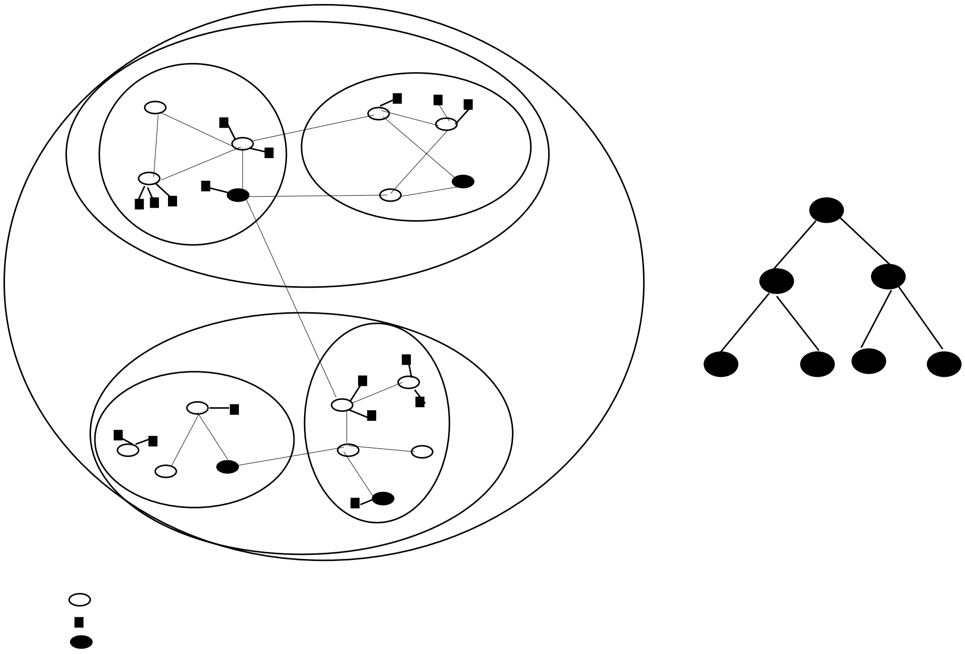


Figure 3: Reference for location management examples in text.

address), and *location finding*, (i.e., obtaining the corresponding address). A key feature of our location management scheme is that it allows for control of updating versus finding on a *per endpoint basis*, thereby accommodating endpoints with diverse call-to-mobility ratios. In our approach, each endpoint is associated with two parameters: the *roaming cluster (RC)* of an endpoint, which is the lowest-level cluster containing the endpoint such that an update is triggered if and only if the endpoint exits this cluster; and the *roaming level (RL)* of an endpoint, which is the hierarchical level of its roaming cluster. For example, in figure 3, if the roaming cluster of endpoint m is E, then no updates will be sent as long as m remains within E, even if m reaffiliates to another switch, e.g., switch P. If m goes outside of E, say to F, then a location update will be triggered. The roaming level of m in this case is 1.

The roaming level for an endpoint is configured, and its roaming cluster is derived from its roaming level and its current location. In particular, the roaming cluster is the roaming-level cluster containing the endpoint. For instance, suppose endpoint q in figure 3 is configured with a roaming level of 2. Then its roaming cluster is $U.A$. Suppose now that q moves to E. Then its roaming cluster would become $U.B$. Thus, when an endpoint exits its current roaming cluster, it will get a new roaming cluster based on its roaming level. An endpoint's roaming level may be changed dynamically in response to the call-to-mobility ratio. In general, the more mobile the endpoint is, the higher its roaming level. For this paper, however, we shall restrict ourselves to statically configured roaming levels and focus on the updating and finding schemes and their performance.

Thus, if an endpoint's roaming level is k, then an update will be triggered if and only if the endpoint exits the level-k cluster containing it. By definition, if the roaming level of an endpoint is 0, then its roaming cluster is the current cell, and an update will be triggered for every reaffiliation – this is equivalent to an "always update" scheme. Similarly, if the roaming level of an endpoint is greater than or equal to the level of the universal cluster, then no updates are triggered – this is equivalent to a "never update" scheme.

We now describe the location updating and location finding procedures. These procedures involve the updating of or searching for association entries in the LM database. Specifically, an entry is a quadruple (endpoint id, endpoint address, endpoint roaming level, expiration time). The endpoint address may be partial, i.e., at a particular LM, it may be a pointer to a child LM containing more details. The roaming level is used to control the scope of paging, as will be described in section 4.2. The expiration time is used to age out entries.

4.1 Location updating

If an endpoint changes its address due to switch reaffiliation then a location update will be sent by the endpoint whenever it exits the current roaming cluster. An update is also sent when the endpoint is activated and affiliates for the first time. As part of the (re)affiliation process, an endpoint obtains its new address. By comparing its old address and its new address, an endpoint determines whether or not it has exited its roaming cluster.

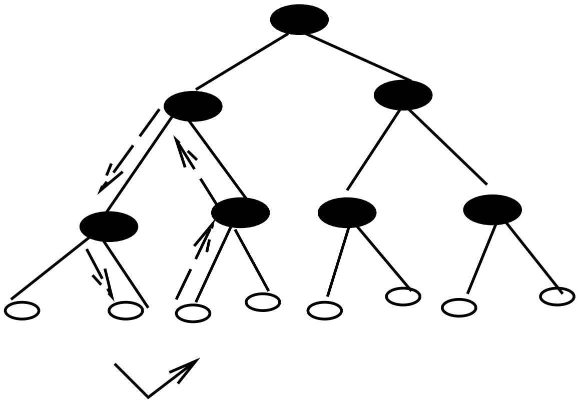


Figure 4: The updating process when an endpoint moves from switch R to switch S in the example hierarchy. "U" represents an update message, and "C" is a cancel message. The lowest common cluster of movement is A and all update propagation is within this cluster.

The location update propagates up the tree of location managers, as shown in figure 4 until it reaches the location manager in the *lowest common cluster* (LCC) in which the movement took place.

Each location manager that receives an update message creates an association entry for the endpoint if there is no entry, or changes the pointer for the association to point to the location manager from which it received the update. The location manager in the lowest common cluster additionally deletes the location manager pointed to previously. The association cancellation trickles down the tree of location managers, following the previously installed pointers, deleting an association entry for the endpoint at each location manager, if found.

We note that the location change only affects location managers within the lowest-level cluster in which the movement occurs. The remainder of the network is oblivious to the change. Since most mobility is local, this approach results in low location management overhead.

4.2 Location finding

Location finding is the process of obtaining the address of an endpoint, given its endpoint id. Our location finding procedure has two components: *location query*, which involves the following of location pointers in location managers to reach the location manager containing the association, and *location paging*, which involves the polling of switches within a given cluster in search of an endpoint. Paging is required because (and only because)[2] an endpoint may have a roaming level greater than 0, and may not have sent an update upon an address change.

A node wishing to obtain the address of an endpoint originates a location query containing three fields: the id of the endpoint whose address is required, its own id, and its own address. If the query is originated by an endpoint, then it is sent to the endpoint's cellhead. Thus, we begin the description of the finding process from a switch in possession of the location query.

The switch (which is also a location manager for its own cell) first searches its affiliation list to see whether the target endpoint is in its own cell. If it is, the location finding procedure terminates. If not, the switch forwards the query to its parent location manager. A location manager receiving a query searches its location database for an association corresponding to the target endpoint. If an entry is found, the query is forwarded to the location manager in a child cluster as pointed to in the entry. If no entry is found, then it is forwarded on to its parent location manager. Thus, the query makes its way up the tree of location managers until it finds an entry for the endpoint and then down the tree until it reaches a level-0 location manager, i.e., the final switch. There are two possibilities:

1. The endpoint is affiliated with the final switch. This is the case if the target endpoint has a roaming level of 0, forcing an update for every movement. In this case, the switch in question

[2] Paging may also be required if the movements are so rapid that by the time the updating process is completed the endpoint has already moved to a new location. Handling such cases requires predictive techniques, which are beyond the scope of this paper.

sends a reply containing the id and address of the target endpoint directly to the location manager that intially orignated the query.

2. The endpoint is not affiliated with the final switch. This is the case if the target endpoint has a roaming level of at least 1. The location query has followed the pointers installed from the endpoint's most recent update but the endpoint has since moved to a new location and not updated. In this case, the switch initiates a location page which contains the same information as the that in the query but is flooded throughout the level-r cluster containing the endpoint, where r is the roaming level of the endpoint being paged. In other words, the paging is done in the current roaming cluster of the endpoint. Note that the roaming level of the endpoint is part of the location database entry in the location manager, and hence r is easily obtained. Upon receiving a page message, a switch checks to see if it is affiliated with the endpoint. If it is, it sends a reply to the requestor as described in item 1.

4.3 Switch mobility

Thus far, we have ignored the effects of switch movements. Mobility of switches causes two problems that must be addressed. First, a switch and the endpoints affiliated with it may all together move to a new cluster. Second, the clustering hierarchy may autonomously undergo changes such as two clusters merging into one or a cluster splitting into two. In both cases, there is no reaffiliation but nonetheless there may be a change in an endpoint's address.

Cluster changing. When a switch changes clusters, it obtains, (see [20] for details), a new address. It then prepares an "aggregated" update that contains the new address and the list of endpoints affiliated with it. The handling of this update is similar to that generated due to reaffiliation (section 4.1), with the difference that the entry for each of the endpoints in the list is changed at the appropriate location manager(s).

Cluster splitting. When a cluster splits into two, the location manager of the original cluster will remain in one of the clusters and the other cluster will get a new location manager. This new location manager, however, will be "empty" to begin with, i.e., it will not have the associations for the endpoints in its cluster. This is illustrated with an example in figure 5. In this example, the lowest-id switch in a cluster assumes the role of a location manager. Thus, switch 5 is a new location manager and hence does not possess the associations for endpoints affiliated with switches 6, 7, and 8. In our protocol, the old location manager sends its entire location database to the new location manager. The new location manager thus has the associations for switches in its cluster, and more. The excess information will eventually be flushed out at the next expiration timer since it will not be refreshed.

This mechanism is a particular case of the more general service handoff mechanism mentioned in the introductory section.

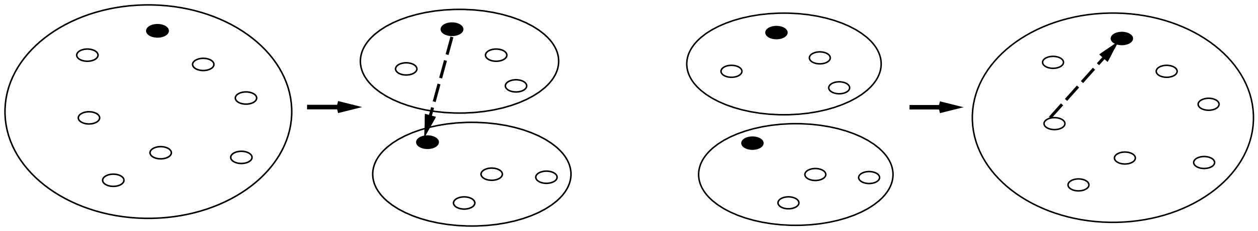


Figure 5: Effect of cluster split (left) and of cluster merge (right) on location management assuming the lowest-id switch is the location manager. After the split, switch 1 continues to be location manager and switch 5 is elected as the new location manager. The *DB-INIT* transfers state. After the merge, switch 5 stops being a location manager, but hands off its state to switch 1.

Cluster merging. When a cluster merges with another cluster, one of the location managers will give up its role (since a cluster has only one location manager). The surviving location manager, however, will only have partial information about the new cluster, i.e., it will only have the associations for endpoints that were in the portion of the cluster before the merge. In figure 5, switch 5 ceases to be a location manager, while switch 1 continues to be a location manager but does not possess the associations for endpoints affilated with switches 6, 7, and 8. The resigning location manager sends its entire location database to the surviving location manager. These associations will be added to those already present in the surviving location manager.

5 Dynamic virtual circuits

In generating, selecting, and using routes for a session, a switch attempts to satisfy the session's service requirements given the qualities of service available within the network. Quality-of-service mechanisms employed include session-specific route generation, resource allocation, virtual-circuit forwarding, and adaptive control of link quality. Collectively, these mechanisms give eacg switch the capability of handling traffic in accordance with its service requirements. In this section, we focus on virtual circuit management in the MMWN system, but we also provide brief descriptions of quality-of-service routing and resource allocation.

5.1 Quality-of-service routing

In the MMWN system, the connectivity and service capabilities of the network are expressed in the form of cluster-based link states which enable on-demand route generation tailored to the specific needs of a session. For each cluster, the link-state information includes service-related attributes pertaining to inter-, trans-, and intra-cluster connectivity, as well as the set of multicast groups that have at least one member located within the cluster. Each cluster contains one or more *QoS managers* that compute and maintain the service attributes pertaining to each virtual gateway connecting two clusters, each virtual link connecting two virtual gateways across a cluster, and each *generic path* from a virtual gateway to points in the interior of the cluster. These service attributes include qualities of service (e.g., delay, jitter, probability of packet loss, unused transmission capacity, and reservable transmission capacity), as well as other properties (e.g., the number of switch hops per virtual link and the volatility of cluster connectivity) that affect the ability of the cluster to meet a session's service needs.

For each virtual link (or virtual gateway), the services available may be different depending upon the direction in which the virtual link (or virtual gateway) is traversed. Moreover, these services may change over time as a result of switch movement and environmental factors affecting the quality of the wireless links. Service information is expressed as a vector of quantities, each component of which represents a statistical characterization of a service attribute, computed over a specified time interval. Using statistical characterizations instead of instantaneously sampled values of service attributes reduces the volatility of link states in mobile wireless networks, and hence reduces the quantity of network resources required for computing and distributing link-state information. The statistical characterization of a service attribue includes the mean of sampled values and, for capacity-related attributes, the degree to which a sampled value is likely to vary from the mean. In the MMWN system, the measure of variation from the mean is not variance but simply the maximum variation from the mean of the sampled values, thus reducing the computational load. For lowest-level clusters and for virtual gateways associated with clusters at any level, the statistical characterization of each service attribute is derived from measured or configured values of switch and link characteristics. For higher-level clusters, the statistical characterization of each service attribute for the virtual links across and generic paths into these clusters is derived from those of the child clusters contained in the routes actually (or expected to be) constructed and used within the cluster.

The QoS manager generates link-state updates for the cluster, based on the service characterizations constructed, and distributes these updates to all other sibling clusters and to all child clusters within the cluster. Distribution of link-state updates is such that each switch obtains link-state information from all other switches within its parent cluster, from all clusters within its grandparent cluster, and in general from all level-$n-1$ clusters within its level-n ancestral cluster, as shown in figure 6. Thus, the granularity of link-state information advertised about a particular cluster (and available to a particular switch for route generation) depends upon the relationship between the switch and the cluster, with respect to the hierarchical control structure.

Each switch is capable of generating quality-of-service routes, on behalf of sessions extant in the network. Before generating a set of routes for a session, the switch must determine the current

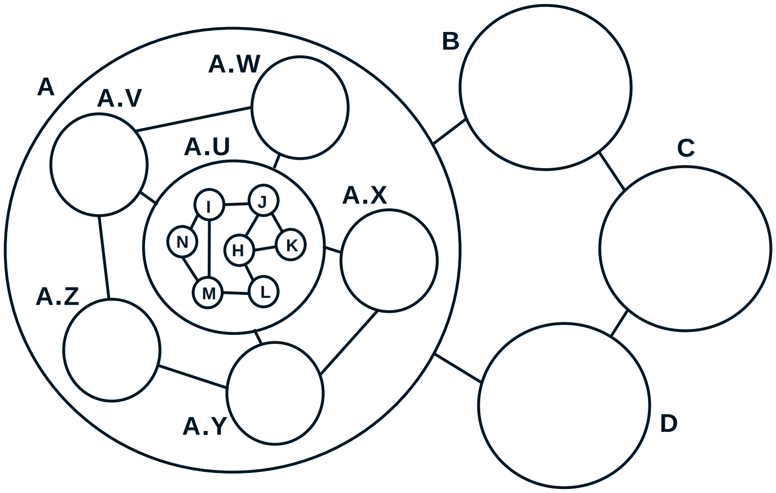


Figure 6: The hierarchical link-state connectivitiy information visible to all child nodes of $A.U$, such as $A.U.M$. The circles represent clusters, and the lines represent virtual gateways. Service information is not shown.

location of the session's endpoints, which may be specified as individual endpoints or as a multicast group. For an individual endpoint, the location manager provides information about the location of the endpoint relative to the hierarchical control structure, as described previously in section 4. For a multicast group, the link-state information for other clusters, obtained by the switch, indicates which clusters contain members of the multicast group. Once the switch determines the location of the session's endpoints, it attempts to generate one or more routes for the session, based upon the locally-available link-state information and the service requirements for the session.

The MMWN system uses a variant of Dijsktra's shortest path first (SPF) algorithm [25] to compute routes. Based on the session's service requirements, the switch picks a service attribute that will serve as the "cost" guiding the construction of the SPF tree. For example, if the session has a very low delay bound, the switch might choose delay as the route cost to optimize in the SPF computation. The remaining service requirements for the session act as constraints on the search. As shown in [26], the use of hierarchically-abstracted, instead of detailed, routing information does not significantly affect the cost of the routes selected. In this SPF search, the vertices are the virtual gateways and the edges are the virtual links. When determining whether to add a virtual link to the SPF tree, the switch determines whether adding the virtual link and its associated outgoing virtual gateway will result in a route that fails to meet at least one of the session's service requirements. The switch only adds the virtual link (and accompanying virtual gateway) to the SPF tree if doing so will not violate the session's service requirements. Whenever multiple feasible routes to a particular destination are discovered, these are ranked giving preference to routes with smaller switch hop count and lower volatility clusters and virtual gateways. Such routes use fewer network resources, enabling the network to accommodate more sessions, and are likely to experience fewer interruptions in session traffic, caused by switch movement. Moreover, all feasible routes to intermediate vertices are stored as starting points for secondary SPF calculations, if the initial SPF calculation fails to find a feasible path. (For more information about the MMWN route generation procedure, consult [20].)

5.2 Establishing virtual circuits

With the MMWN system, all data messages use route-directed forwarding whereby the switch acting on behalf of a session endpoint determines the route along which the session traffic travels. This route may be carried in data messages or may be used to establish virtual circuits (VCs) over which data messages will travel. Route-directed forwarding prevents the persistent routing loops that may form with destination-oriented forwarding as a result of forwarding decisions based on inconsistent routing information at different switches. Such inconsistencies are likely to be common in large mobile wireless networks, because of the frequent changes in link states and the propagation delays incurred by link-state updates.

In the MMWN system, each route is specified at fine granularity near the source and progressively coarser granularity toward the destination(s), according to the granularity of the link-state information available for route generation at the source. For example, given the link-state information shown in figure 6), $A.U.M$ might specify the following route to reach a destination, $C.w_j.z_k$, inside C:

$$A.U.M \rightarrow A.U.I \rightarrow A.U.J \rightarrow A.U.K \rightarrow A.U \rightarrow A.W \rightarrow A \rightarrow B \rightarrow C \rightarrow C.w_j.z_k.$$

As a message progresses along its route, intermediate switches refine the granularity of the route by generating more detailed segments of the route using their locally-available link-state information. For example, in figure 6, a switch on the B side of the virtual gateway connecting A and B would generate a route across B to fill in more detail in the route originally generated by $A.U.M$.

Virtual circuits are recommended for sessions consisting of a large number of messages or requiring reserved resources. Once a switch sets up state for a virtual circuit, it may immediately begin to forward data messages along that virtual circuit, without waiting to determine whether the virtual circuit has been set up successfully end to end. Each data message travelling over an established virtual circuit carries virtual circuit identifiers which are used by the recipient switches for rapid access of state information related to the virtual circuit and thus for efficient forwarding. The virtual circuits created by the MMWN system are flexible, accommodating switch and endpoint movements and supporting multipoint sessions with dynamic membership. Space limitations preclude an in-depth discussion of multipoint virtual circuits in this paper; refer to [20] for details on management and use of multipoint virtual circuits.

A virtual circuit management protocol establishes, repairs, and eliminates forwarding state and other traffic handling information in the switches participating in the virtual circuit. The *initiator* is the switch that initiates establishment of a virtual circuit on behalf of a session endpoint, and the *targets* are the switches acting on behalf of session endpoints to which the virtual circuit is ultimately directed. Mostly likely, a source of session traffic will be the initiator and destinations will be the targets. There is, however, nothing in the protocol that precludes destinations from becoming initiators or sources from becoming targets. Each branch of a virtual circuit has directionality, specified by the initiator, which indicates the direction(s) (downstream toward the targets, upstream toward the initiator, or both) in which data may flow and resources may be reserved, as shown in figure 7.

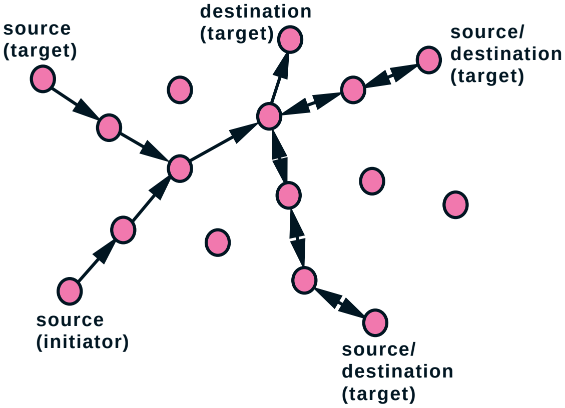


Figure 7: A multipoint virtual circuit established by the node marked initiator. Some of the target nodes are sources only, destinations only, or both sources and destinations. The arrows represent the direction of resource reservation and data flow.

In a mobile wireless network, the limiting resource is likely to be transmission capacity, and hence the switches must be able to reserve and control the use of this resource in order to meet sessions' service requirements and to accommodate as many simultaneous sessions as possible. The MMWN system enables switches to allocate and manage resources for individual virtual circuits. Here, we describe resource reservation only; the mechanisms for controlling the use of reserved resources – including traffic accounting, marking, queueing, and scheduling – are described in [20]. During virtual-circuit establishment, each switch participating in the virtual circuit reserves resources for communication with the immediate upstream and downstream hops, depending upon the directionality of this portion of the virtual circuit. For higher-level virtual circuits, resource reservation is simply an accounting operation which keeps track of the amount of resources reserved for the virtual circuit. For the lowest-level virtual circuit, resource reservation also includes a "diffusion" procedure (as well as interactions with the channel access mechanism, which are beyond the scope of this paper).

Each switch diffuses the reservation to its immediately neighboring switches (those that are not currently part of the virtual circuit), so that they can be prepared to accommodate a repair of the virtual circuit if necessary. Each neighboring switch, upon receiving a new diffused resource reservation, reduces both its actual and its advertised reservable transmission capacity (duplicate diffused reservations from different switches are detected). For each type reservable transmission capacity, actual and advertise, the quantity is reduced by a fraction of the reservation, possibly equal to zero and dependent on the priority of the virtual circuit to which the reservation pertains. These two steps together reserve transmission capacity at the neighbor and bias other virtual circuits away from using routes that include the neighbor, hence increasing the likelihood that the neighbor will be available to accommodate a repair of the virtual circuit should one be necessary. This process of diffusing resource reservations out from the current route of a virtual circuit serves a similar purpose to the "shadow reservations" proposed in [28] for cellular networks.

The virtual circuit management protocol employs two types of messages: *setup* and *accept*. Setup messages are used to establish new virtual circuits and repair existing ones. Each setup message carries information pertaining to the virtual circuit including the service requirements (such as desired throughput and priority), the endpoints (specified individually or as a multicast group), the route (which may be a single path in the case of a point-to-point session or a tree in the case of a multipoint session), and the virtual circuit identifier, which remains fixed along the length of the virtual circuit. Accept messages are used to inform the initiator of successful establishment or repair of a virtual circuit in order to terminate future attempts to set up the virtual circuit. These messages also contain the route and virtual circuit identifier, and serve to update switches in case the route was repaired between sending the setup message and receiving the accept message. Information about the current route for a virtual circuit is used by a switch to determine how to repair that virtual circuit following a perceived connectivity failure.

We expect virtual circuits to be used predominantly by stream-oriented sessions, and thus there should exist no large gaps between data messages sent over a virtual circuit. A large time interval during which no data messages are perceived indicates either that the session has terminated or that there is break in the virtual circuit. In either case, the appropriate response for a switch detecting such a gap is to eliminate state associated with a virtual circuit, thus releasing resources which may then be allocated to other virtual circuits. Furthermore, in a mobile network, rapid elimination of stale state reduces the number of unnecessary repairs (i.e., repairs initiated by switches that believe they are part of an active virtual circuit, when in fact that virtual circuit is no longer used).

With the MMWN system, a single end-to-end virtual circuit actually comprises a hierarchy of component virtual circuits, one for each cluster and each virtual gateway traversed. Each of these component virtual circuits is generated by a switch that refines the details of a portion of a route, during the establishment of the top-level virtual circuit. Starting with the route specified above from $A.U.M$ to a destination, $C.w_j.z_k$, the component virtual circuits and their routes might appear as follows. The lower-case clusters are not visible to $A.U.M$ and are included in routes generated by switches along the way as the virtual circuits are established.

$$\begin{aligned}
VC_1: &\quad A.U.M \rightarrow A.U.I \rightarrow A.U.J \rightarrow A.U \rightarrow A.W \rightarrow A \rightarrow B \rightarrow C \rightarrow C.w_j.z_k \\
VC_2: &\quad A.U.M \rightarrow A.U.I \rightarrow A.U.J \\
VC_3: &\quad A.U \rightarrow A.W \\
VC_4: &\quad A.W.x_1 \rightarrow \ldots \rightarrow A.W.x_n \\
VC_5: &\quad A \rightarrow B \\
VC_6: &\quad B.y_1 \rightarrow \ldots \rightarrow B.y_m \\
VC_7: &\quad B \rightarrow C \\
VC_8: &\quad C.w_1 \rightarrow \ldots \rightarrow C.w_j \\
VC_9: &\quad C.w_j.z_1 \rightarrow \ldots \rightarrow C.w_j.z_k.
\end{aligned}$$

A new session (or higher-level virtual circuit) can be multiplexed onto an existing virtual circuit, provided that in doing so the existing virtual circuit can satisfy the service requirements of the new session (or higher-level virtual circuit) and can continue to satisfy the service requirements of the existing sessions. Multiplexing additional sessions onto existing virtual circuits has several advantages. First, it reduces the amount of session-specific state that must be retained in switches throughout the network. Excepting the switches at the ends of the virtual circuit at which the multiplexing occurs, none of the other switches in the virtual circuit need to maintain state about the new session. Second, it reduces the amount of time in establishing forwarding state in the network, by taking advantage of existing virtual circuits rather than setting up new ones. Third, it reduces the amount of work in repairing virtual circuits. Only the lowest-level virtual circuit affected by a connectivity failure must be repaired. Other higher-level virtual circuits multiplexed onto the repaired virtual circuit are automatically repaired as a result of repairing the lower-level virtual circuit. Note, however, that as a result of this multiplexing, each data message travelling via a virtual circuit must carry a stack of virtual circuit identifiers to be pushed and popped as the message is forwarded toward its destination.

Sessions requiring low loss rates or high throughput may use multiple independent virtual circuits to carry their traffic. For a session with a low tolerance for loss, a switch acting on its behalf may enlist multiple virtual circuits, established over maximally disjoint routes, in order to minimize the instances and duration of interruptions resulting from switch movement or degradation of link quality. Depending upon the session's loss tolerance and the loss rates of the virtual circuits, the switch might either transmit all messages over a single virtual circuit at a time, holding the others in reserve in case the primary one fails, or transmit simultaneously multiple copies of the message, one over each virtual circuit. The destination is responsible for eliminating duplicate messages. In a mobile wireless network, the limited capacity of the wireless links may preclude the existence of a single virtual circuit with the capacity necessary for a high-throughput session. For such a session, a switch acting on its behalf may enlist multiple virtual circuits established over routes that collectively provide the capacity required, forwarding different messages along different virtual circuits to maximize throughput. The destination is responsible for reassembling the messages in the correct order.

5.3 Repairing virtual circuits

When an endpoint or a switch moves and participates in a session using a virtual circuit, the virtual circuit must be able to adapt to the movement so as to minimize interruptions in the session. Nodes may use adaptive link control to prolong the existence of a link as the nodes move apart. If the nodes continue to separate, the virtual circuit may be forced to take an alternate path to maintain connectivity, because further boosting of transmission power through adaptive link control may not be possible or it may cause undue interference to other transmissions. The MMWN system provides switches with several mechanisms that enable them to repair existing virtual circuits in response to node movement. Repairing portions of virtual circuit rather than reconstructing entirely new virtual circuits has two advantages. First, it reduces the amount of time required to reestablish forwarding state for a session, hence minimizing the duration of any session interruptions. Second, it reduces the quantity of network resources required to reestablish forwarding state for the session, thus enabling the network to accommodate more sessions at one time. Virtual circuit repair is always initiated by the switch on the upstream side of a perceived connectivity failure, so that multiple switches do not simultaneously attempt repairs related to the same failure. While the virtual circuit is being repaired, traffic continues to flow down the old portion of the virtual circuit, in order to minimize the interruptions in the session. As soon as the virtual circuit repair is complete, traffic is allowed to flow over the new portion of the virtual circuit. The MMWN system provides several different mechanisms that enable virtual circuits to cope with mobile nodes, each of which is described briefly below.

Switch movements. To accommodate switch movements, the MMWN system provides switches with three means of repairing existing virtual circuits. Two of these, handoff and local reroute, are available only for repairing lowest-level virtual circuits. End reroute may be used to repair any virtual circuit. Each repair includes a setup message generated by the switch initiating the repair and an accept message generated by the switch at the point where the repaired route rejoins the old route for the virtual circuit. The setup message actually travels to the target, even though the rejoining point may be upstream of the target, in order to convey information about the new route to all downstream switches. Similarly, the accept message actually travels to the initiator, to convey the new route to all upstream switches.

On a lowest-level virtual circuit, when a switch, C, on the upstream end of a link detects a failure to reach the downstream switch, D, it begins the virtual circuit repair process. First, C attempts a *handoff*, which is the simplest and fastest type of repair. Handoff entails finding a neighboring switch (or a set of neighboring switches if D was a bifurcation point for a multipoint virtual circuit), which is also a neighbor of the switch, E, immediately downstream from D. If C is successful in locating such a neighbor, I, it repairs the virtual circuit through I to E, as shown in figure 8(a). Otherwise, if the handoff attempt is not successful, C then attempts a *local reroute* to E. Local reroute differs from handoff in that it involves route generation. C tries to obtain a route, not exceeding a specified number of hops, from itself to E. Limiting the hop count helps prevent repaired virtual circuits from becoming unduly long, thus incurring large delays and using network resources inefficiently. If C is successful in obtaining a route to E, it repairs the virtual circuit according to the newly-generated route segment, as shown in figure 8(b). Otherwise, if the local reroute attempt is unsuccessful, C executes an end reroute in which it attempts to find a route from itself to the terminating switch of the virtual circuit, as shown in figure 8(c). Failure to repair a virtual circuit results in a break in all higher-level virtual circuits multiplexed over this lower-level virtual circuit. Such failures are detected by the switches participating in the higher-level virtual circuits, upon elimination of state associated with the lower-level virtual circuit. In response to such a failure, the upstream switch for each affected higher-level virtual circuit then initiates an end reroute of that virtual circuit.

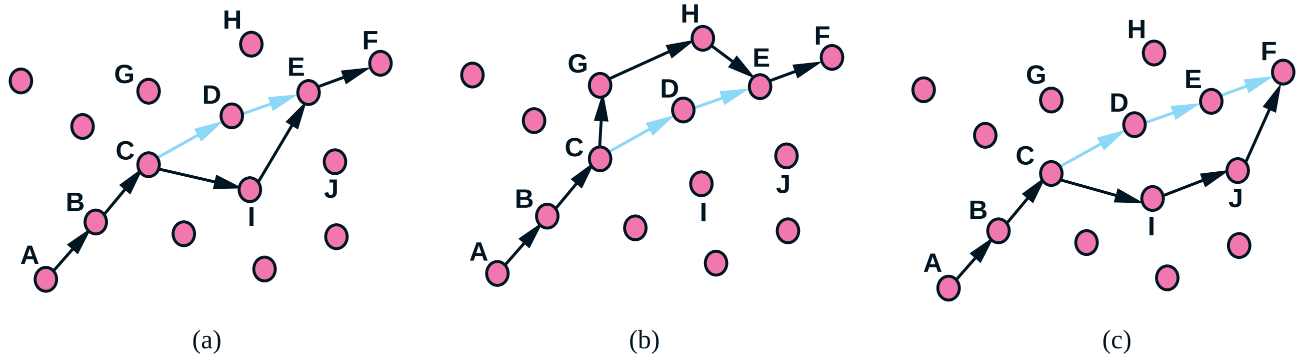
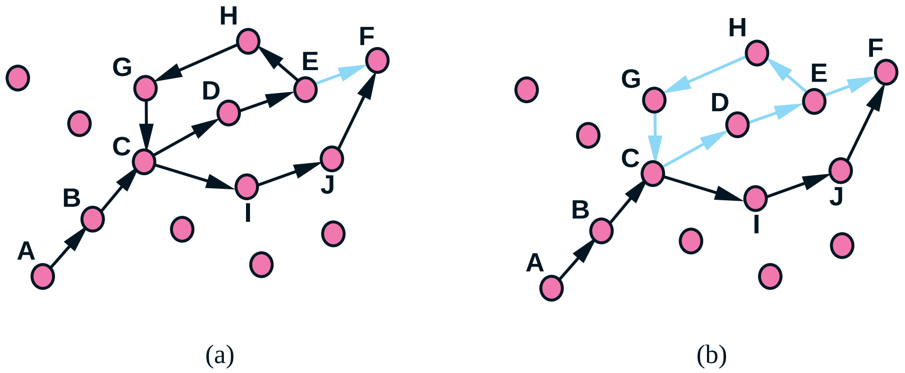


Figure 8: Repair of a virtual circuit using handoff (a), local reroute (b), and end reroute (c). The light arrows represent the portion of the path requiring repair. The dark arrows represent the path following successful repair.

Repair of virtual circuits may result in loops in the virtual circuit. These are detected and eliminated by the switches along the virtual circuit, using their knowledge of the current state of the virtual circuit's route. As depicted in figure 9(a), following a connectivity failure to F, E executes an end reroute which creates a loop through C. C detects this loop by noticing that it appears twice in the route carried in the repairing setup message generated by E, and it subsequently excises this loop, as shown in figure 9(b). (For further details on loop detection, refer to [20].)

A virtual circuit that is a component of a higher-level virtual circuit, and whose initiator (or target) is a member of a virtual gateway, may be repaired, even if the link from the initiator (or to the target) fails. Each member, x, of a virtual gateway keeps track of all virtual circuits that it initiates and terminates. It also distributes this information to all other peers on both sides of the virtual



(a) (b)

Figure 9: Formation (a) and elimination (b) of a loop resulting from an end reroute. The light arrows represent the portion of the path requiring repair. The dark arrows represent the path following repair.

gateway, so that if connectivity to x were to fail, other peers could repair the affected virtual circuits. Moreover, when the target of a virtual circuit is a member of a virtual gateway, the penultimate switch on that virtual circuit is allowed to repair the virtual circuit by selecting an alternate target within the virtual gateway. A portion of a virtual circuit is depicted below, which initially consists of three lower-level virtual circuits, as follows:

$$\begin{array}{ll} VC_1: & A \rightarrow B \rightarrow C \rightarrow D \\ VC_2: & D \rightarrow E \\ VC_3: & E \rightarrow F \rightarrow G \rightarrow H. \end{array}$$

When the link between C and the virtual gateway member, D, fails, both VC_1 and VC_2 are affected. C chooses I as the new target for VC_1, and I becomes the new initiator for VC_2, as shown in figure 10(a). When the link between virtual gateway members, D and E, fails, both VC_2 and VC_3 are affected. D chooses J as the new target for VC_2, and J becomes the new initiator of VC_3.

Endpoint movements. The simplest mechanism for accommodating endpoint movement is extension of an existing virtual circuit to the endpoint's new switch, which is a neighbor of the endpoint's previous switch. This approach results in very fast repair of a virtual circuit, but if applied repeatedly can result in long routes and hence large delays and inefficient use of network resources. Therefore, when the length of a route thus extended exceeds a specified hop count, the initiator attempts to obtain a shorter route to the target.

A more complicated mechanism for accommodating endpoint movement is the use of a multipoint virtual circuit connecting the set of expected locations of the endpoint, thus eliminating the setup delay when the endpoint moves among these locations. This latter approach is similar to the

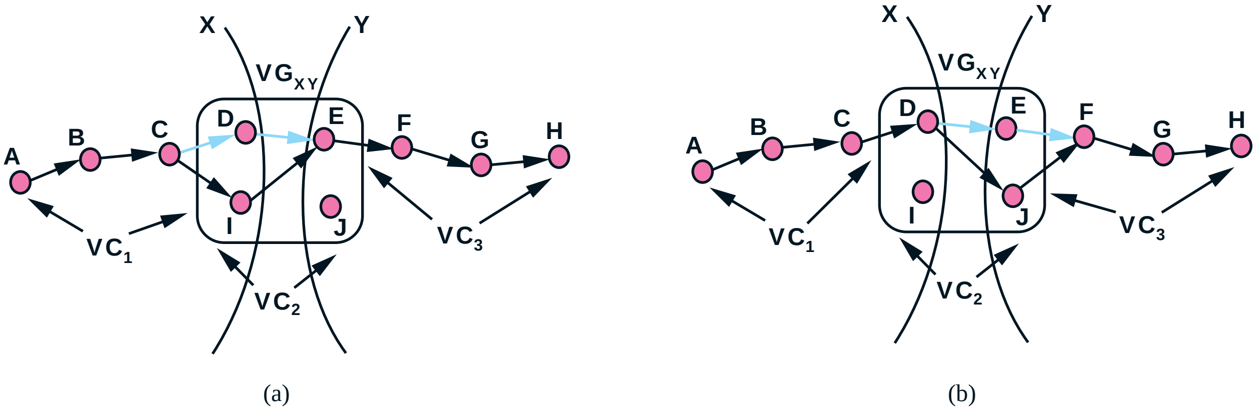


Figure 10: Repair of a virtual circuit, when the failure occurs on a link to a virtual gateway (a) and when the failure occurs on a link across a virtual gateway (b). The light arrows represent the portion of the paths requiring repair. The dark arrows represent the paths following successful repair.

"virtual connection trees" proposed in [27]. In establishing this multipoint virtual circuit, the initiator can choose one of two options. The first approach is to reserve the required resources on all branches and to send a copy of each message down each branch simultaneously. This approach yields a high probability of uninterrupted communication when an endpoint moves, but it consumes a large number of network resources that are thus unavailable for other traffic sessions. Thus, it should only be used when the endpoint is highly mobile and the session traffic is of critical importance. The second approach is to reserve the required resources on the branch to the current location of the endpoint and a fraction (possibly zero) of the resources on the other branches and to send each message down only the active branch. Reserving a small fraction of the necessary resources on the other branches increases the likelihood that the full quantity of required resources will be available on those branches should they become active. A branch becomes active when an endpoint moves to a new location on the multipoint virtual circuit and requests the necessary resources. If it is not possible to reserve the required resources along the newly active branch, the initiator attempts to find an alternate route for the session. Meanwhile, the current branch is used until a better alternative is found. If the endpoint moves to a location not currently reachable through the multipoint virtual circuit, the virtual circuit is extended from the endpoint's old location to reach its new location.

6 Experimental results

In this section, we present results of large-scale simulations of the clustering, virtual circuit management, and location management components of the MMWN system. The main purpose of these simulations was to determine the sensitivity of each of these system components to the transmission range and mobility of network nodes, in order to define an effective operating region and to estimate the overhead of these procedures within this region. Ultimately, these results indicate how well these procedures are likely to work in an actual multihop mobile wireless network. The simulation models were constructed using the Maisie simulation language [29], a C-based simulation language that can be used for sequential and parallel execution of discrete-event simulation models. All simulations were executed on a Sun SPARC workstation running SunOS 4.1.3.

6.1 Network and mobility models

In the simulated environment, an instance of a multihop mobile wireless network is generated by placing a set of S switches and E endpoints in a circular area with radius R within a square field of $L \times L$ meters. Each switch and endpoint is positioned randomly and independently and has a transmission range of T meters. We assume free-space propagation with a threshold cutoff, i.e., two nodes can communicate with each other if and only if their Euclidean distance is less than or equal to T. A shortcoming of this model is its inability to capture the effects of barriers, foliage, multipath interference, etc. We have adopted this model for our simulations because its computational simplicity results in significantly-reduced simulation times.

The set of switches in the network is divided into G groups with S_i switches in group $i, 1 \leq i \leq G$. The movement of each group as a whole and of each node within each group follows an *exponentially correlated random mobility* (ECRM) model. This model is described by the following equation:

$$b(t+1) = b(t) \cdot e^{\frac{-1}{\tau}} + s \cdot \sigma \cdot \sqrt{1 - e^{\frac{-2}{\tau}}} \cdot r \qquad (1)$$

where

$b(t)$ is the position (r, θ) of a group or a node at time t,
τ is a time constant that regulates the rate of change,
σ is the variance that regulates the variance of change,
s is the speed of the node, and
r is a gaussian random variable.

Using this model, the movement of each group as a whole is controlled independently of the movements of other groups and nodes within the group. At each time step, a group moves a random distance in a randomly selected direction. The τ and σ variables, specified separately for the distance and direction, control the nature of the movement. In general, smaller values of τ result in more random movement, and larger values of σ result in more variation from a given direction. Node movements are specified in an analogous manner. Within a group, all nodes have the same set of τ and σ variables; nodes in different groups may have different sets of variables. This mobility model permits representation of typical node movements in a tactical network, such as the maneuvers of a battallion consisting of several squads of soldiers. The squads may be represented as groups that move as a whole; the soldiers within a squad, while exhibiting some randomness of movement, are aligned with the overall movement of their squad.

6.2 Observations

For our experiments related to clustering and virtual circuit management, we simulated a mock reconnaisance exercise typical of a tactical environment. There were 4 groups (A, B, C,and D) of 6 switches each, in an area of 6×6 km. The movement of each group and each switch within a group follows the ECRM model described above. Group A is stationary while groups B, C, and D move east, northeast, and north, respectively, for a period of time, then reverse directions (i.e, west, southwest, and south) for a period of time, and then reverse direction again, and so on. We experimented with group speeds of 15 km/hr (4 m/sec) through 72 km/hr (20 m/sec), and with transmission ranges of 1 km through 3 km. Simulation data was collected over a period of 600 seconds of simulation time, sufficient to establish adequate confidence in the simulation results.

Figure 11 shows the number of cluster splits, merges, and their sum (i.e., cluster reformations) as a function of the transmission range for speeds of 8 m/sec and 16 m/sec. This data set was obtained with $N_{split} = 6$, $N_{merge} = 3$, and $N_{pref} = 4$ (refer back to section 3.2.1 for a description of these parameters). The frequency of cluster reformations, no more than 2 per minute even for switch speeds up to 16 m/sec, is well within acceptable limits of overhead in terms of computation and information distribution. As the transmission range of the switches increases, the number of merges decreases, because there are fewer isolated clusters of containing less than 3 nodes and hence needing to merge. As expected, increasing the switch speed increases the number of cluster splits and merges.

For our experiments related to virtual circuits, we considered only lowest-level virtual circuits, so that we could observe the behavior of all three types of repairs – handoff, local reroute, and end reroute – the first two of which are only possible at the lowest-level. Hence, the clustering procedure was not invoked for these simulations. During the course of the simulations, 7 different sessions were initiated, one each to group A from groups B, C, and D and one each from A to B, B to C, C to D, D to A. Each session lasted for the duration of the simulation and generated data packets at the constant rate of one per second, regardless of whether a virtual circuit was yet in place. None of the sessions required resource reservation.

Figure 12(a) shows the dependence of the frequency of virtual-circuit repairs on the transmission range for rapidly moving swithces. Figure 12(b) shows the fraction of packets dropped as a function of transmission range for different switch speeds. For all three types of repairs – handoff, local reroute, and end reroute – the frequency first increases with increasing transmission range and then decreases. The

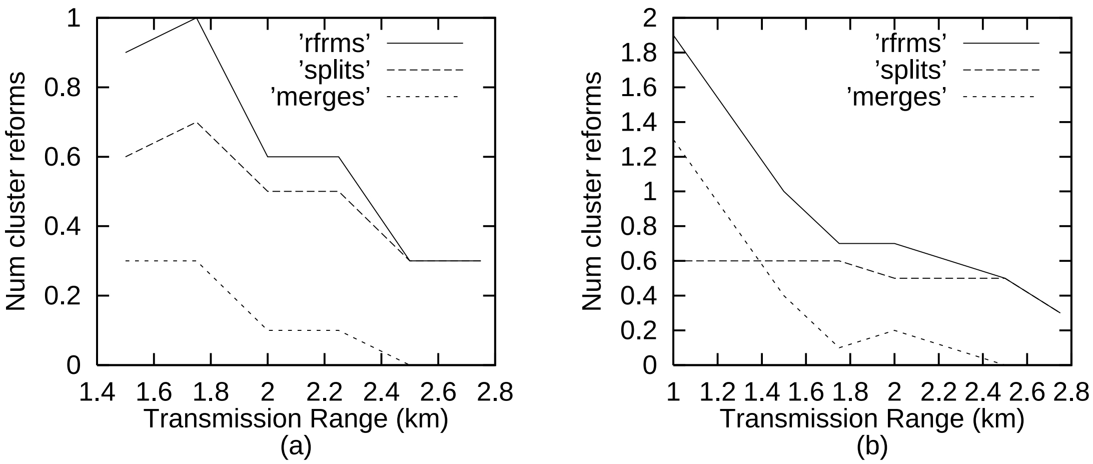


Figure 11: Cluster splits, merges, and reformations per minute versus transmission range for switch speed equal to 8 m/sec (a) and switch speed equal to 16 m/sec (b).

reason for this behavior is as follows. At low transmission ranges, the network is poorly connected, hence fewer virtual circuits are successfully established, and consequently there are fewer virtual circuits to repair. The poor connectivity at low transmission ranges is also reflected in figure 12(b), in this case as a high fraction of dropped packets. As transmission range increases, more virtual circuits are successfully established, and hence the number of repairs increases. At the high end of the transmission ranges simulated, the connectivity is good enough so that switch mobility causes fewer disruptions in virtual-circuit connectivity, and hence fewer repairs are required.

For our experiments related to location management, we simulated endpoints as well as switches, clustered in a two-level hierarchy. There were 20 switches and 4 endpoints, with no groups. The switches moved very slowly (1 m/sec), and endpoint speeds ranged from 18 km/hr (5m/sec) to 72 km/hr (20 m/sec).

Figure 13(a) shows the total overhead incurred by the location management scheme as a function of call frequency. Here, the total overhead is computed as the number of location management messages transmitted over each hop, anywhere in the network, due to location updating or finding procedures, over the entire simulation. Figure 13(b) shows the total overhead as a function of endpoint speed. In both cases, there is a "cutoff point" after which changing the roaming level results in increased efficiency. As shown in figure It is more efficient to use a roaming level of 1, for a call frequency of less than 0.25 calls per minute (as shown in figure 13(a)) and for endpoint speed greater than 17 m/sec (as shown in figure 13(b)).

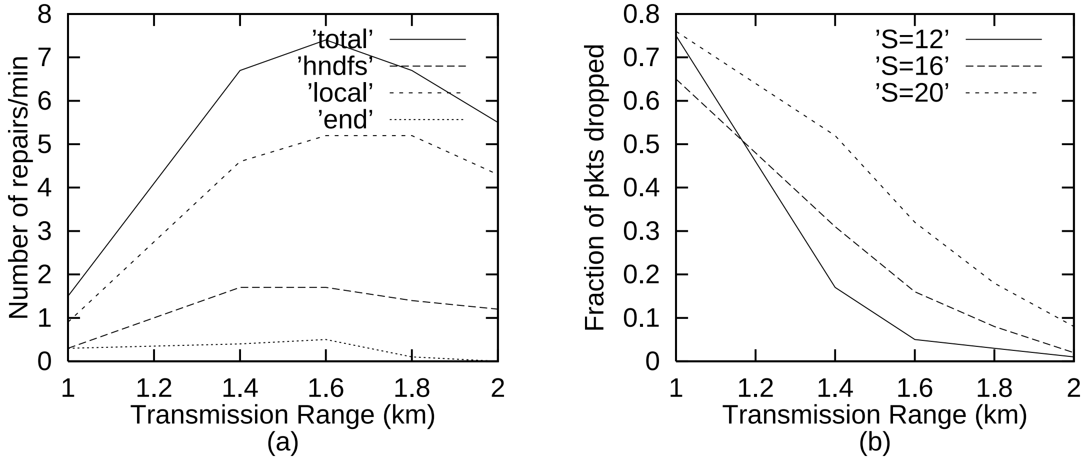


Figure 12: Virtual circuit handoffs, local reroutes, end reroutes, and total repairs per minute versus transmission range for switch speed equal to 16 m/sec (a). Fraction of packets dropped versus transmission range for switch speeds of 12 m/sec, 16 m/sec, and 20 m/sec (b).

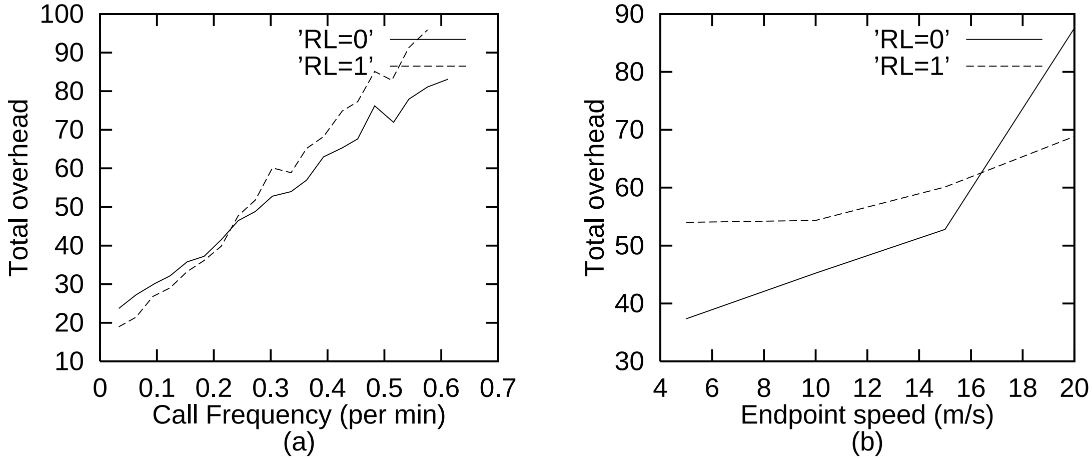


Figure 13: Total overhead of location tracking versus call frequency with endpoint speed equal to 15 m/sec (a) and and versus endpoint speed with call frequency equal to 0.3 calls/min (b). In each case, the roaming level (RL) of the endpoints were set equal to 0 or 1.

7 Concluding remarks

The MMWN system has been designed to support distributed, real-time multimedia applications in large, multihop mobile wireless networks. It consists of a set of modular components which enable the network to adapt to node mobility and link-quality fluctuations, at the link and network layers. In this paper, we have presented three key components of the MMWN system: the clustering and cell formation procedures for creating a virtual hierarchical control structure over the physical network, thus enabling efficient representation of network connectivity and services at multiple levels of abstraction; the location management procedures for determining the location of endpoints with resepct to the hierarchical control structure, allowing for tradeoffs of location updating and paging costs depending upon an individual endpoint's call-to-mobility ratio; and the virtual circuit management procedures for establishing (multipoint) virtual circuits for stream-oriented sessions according to the session service requirements and for rapidly repairing existing virtual circuits as switches and endpoints move throughout the network. Our simulation results have shown that effective operating regions for these three components fit well with environmental conditions typically experienced in a tactical network. In addition to our on-going simulation work, we are also in the process of implementing and testing the MMWN system on actual radios within a mobile wireless testbed. The results of both of these efforts will provide valuable insight into the performance of multihop mobile wireless networks in the context of quality of service provision.

References

[1] M. Schwartz, Network management and control issues in multimedia wireless networks, IEEE Personal Communications (June 1995) 8-16.

[2] C.R. Lin and M. Gerla, Multimedia transport in multihop dynamic packet radio networks, *Proc. IEEE GLOBECOM* (1995) pp. 209-216.

[3] K. Lee, Adaptive network support for mobile multimedia, *Proc. of ACM MOBICOM*, Berkeley, CA (1995) pp. 62-74.

[4] A. Alwan et al., Adaptive mobile multimedia networks, IEEE Personal Communications (April 1996) 34-51.

[5] R.S. Kahn, J. Gronemeyer, J. Burchfiel, and R. Kunzelman, Advances in packet radio technology, Proc. of the IEEE 66(11) (1978) 1468-1496.

[6] J. Jubin and J.D. Tornow, The DARPA packet radio network protocols, Proc. of the IEEE 75(1) (1987) 21-32.

[7] N. Shacham and J. Westcott, Future directions in packet radio architectures and protocols, Proc. of the IEEE 75(1) (1987) 83-99.

[8] G. Lauer, Advanced protocols for the SURAN packet radio network, *Proc. of the SHAPE Packet Radio Symposium* (1989).

[9] J. Stevens, Spatial reuse through dynamic power and routing control in common-channel random-access packet radio networks, SURAN Program Technical Note (SRNTN) 59 (1988). Available from the Defense Technical Information Center.

[10] D. Beyer et al., Packet radio network research, development and application, *Proc. of the SHAPE Packet Radio Symposium* (1989).

[11] N. Shacham and E. Craighill, Dynamic routing for real-time data transport in packet radio networks, *Proc. of IEEE INFOCOM*, Las Vegas, (1982) pp.152-158.

[12] I. Chlamtac and A. Faragó, Making transmission schedules immune to topology changes in multi-hop packet radio networks, IEEE/ACM Trans. on Networking, 2(1) (1994) 23-29.

[13] C.L. Fullmer, J.J. Garcia-Luna-Aceves, Floor acquisition multiple access (FAMA) for packet-radio networks, *Proc. ACM SICOMM* Cambridge, MA (1995) pp.262-273.

[14] R. Ogier, V. Rutenburg, and N. Shacham, Distributed algorithms for computing shortest pairs of disjoint paths, IEEE Trans. Information Theory, (March 1993) 443-455.

[15] M.B. Pursley and H.B. Russell, Routing in frequency-hop packet radio networks with partial-band jamming, IEEE Trans. on Communications 41(7) (1993) 1117-1124.

[16] M.B. Pursley, H.B. Russell, and P.E. Staples, Routing multimedia packets in a frequency-hop packet-radio network, *Proc. IEEE MILCOM* (1996) pp. 220-224.

[17] A. Acampora and M. Nagshineh, Control and quality-of-service provisioning in high-speed microcellular networks, IEEE Personal Communications (Second Quarter 1994) 36-43.

[18] M. Gerla and J.T.-C. Tsai, Multicluster, mobile, multimedia radio network, Wireless Networks 1(3) (1995) 255-266.

[19] M. Steenstrup, ed., *Routing in Communication Networks* (Prentice-Hall, Englewood Cliffs, NJ, 1995).

[20] M. Bergamo, R.R. Hain, R. Ramanathan, and M. Steenstrup, MMWN preliminary design (draft), ftp.bbn.com/pub/ramanath/mmwn-design.ps.

[21] G. Lauer, Hierarchical routing design for SURAN, *Proc. of the ICC* (1986), pp. 93-101.

[22] W.T. Tsai, C.V. Ramamoorthy, W.K. Tsai, and O. Nishiguchi, An adaptive hierarchical routing protocol, IEEE Trans. on Communications 38(8) (1989) 1059-1075.

[23] M. Mouly, M.B. Pautet, The GSM system for mobile communications, M. Mouly, 49 rue Louise Bruneau, Palaiseu, France, (1992).

[24] W.R. Young, Advanced mobile phone service: introduction, background, and objectives, The Bell Systems Technical Journal, 58 (1979) 1-14.

[25] E.W. Dijkstra, A note on two problems in connection with graphs, Numer. Math., 1 (1959) 269-271.

[26] l. Kleinrock and F. Kamoun, Hierarchical routing for large networks, Computer Networks 1 (1977) 155-174.

[27] A.S. Acampora and M. Naghshineh, An architecture and methodology for mobile-executed hand-off in cellular ATM networks, IEEE J. on Selected Areas in Communications 12(8) (1994) 1365-1375.

[28] D.A. Levine, I.F. Akyildiz, and M. Naghshineh, The shadow cluster concept for resource allocation and call admission in ATM-based wireless networks, *Proc. of ACM MOBICOM*, Berkeley, CA (1995) pp. 142-150.

[29] R.L. Bagrodia, Designing efficient simulations in Maisie, *Proc. of the 1991 Winter Simulation Conference*, Phoenix, AZ (1991) pp. 243-247. (Also, see http://may.cs.ucla.edu/projects/maisie.)