A Distributed Architecture for Multimedia in Dynamic Wireless Networks†

Chunhung Richard Lin and Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095-1432
{chunhung, gerla}@cs.ucla.edu

Abstract
The paper presents a self-organizing, wireless mobile radio network for multimedia support. The proposed architecture is distributed and has the capability of rapid deployment and dynamic reconfiguration. Without the need of base stations, this architecture can operate in areas without a wired backbone infrastructure. This architecture provides an instant infrastructure for real-time traffic transmission. Based on the instant infrastructure, a stable and loop-free routing protocol is implemented.

1. INTRODUCTION
Current wireless systems, such as cellular systems, have fixed network configuration and fixed base stations or servers that are linked by a wired backbone infrastructure. In some cases, such as emergency disaster relief, when the backbone is not available, this type of architecture is infeasible. The goal of WAMIS is to overcome these constraints, and to develop techniques for the design of wireless networks that are adaptable to a variety of transmission environments, network configurations, and user services (including data, voice and image) [5-8]. We propose a novel architecture which enables rapid deployment and dynamic reconfiguration of a network of wireless stations. More specifically, we propose a wireless mobile instant infrastructure concept for real-time communications. If an infrastructure exists, the wireless network will also provide access to it through multiple hops.

One of the goals of WAMIS is to support multimedia services (voice, data, video, and image). There are basic differences between the requirements for the transmission of bursty traffic (data, image) and real-time traffic (voice, video). Bursty traffic is error-sensitive, while real-time traffic is delay-sensitive. Thus, real-time sessions require bandwidth and delay guarantee, and are generally carried on virtual circuits (VC) which are established at call setup time. On the other hand, bursty traffic is carried in a connectionless (datagram) mode.

In the mobile environments, the logical link between two radio nodes is subject to frequent and unpredictable change. Therefore, the reliability of packet routing is important to guarantee the integrity of network services. As mentioned in [3], if the rate of change is very low (nearly static), some stable shortest-path algorithms such as Bellmann-Ford, Dijkstra and Floyd-Warshall [2] can solve it. If the rate of change is very high, only flooding is viable, since no algorithm can react fast enough. Our focus is on the rate of change which is not so fast as to make flooding the only option, but not so slow as to make one of the shortest-path algorithms applicable. The routing algorithm must provide the function to maintain the VC in the face of link breakages. Prior works [2-4] can only satisfy a part of the requirements. We propose an adaptive hierarchical routing protocol to meet the requirements which is implemented on top of our architecture.

The paper is organized as follows. Section 2 describes some definitions used in the paper. Section 3 presents the network architecture and discusses its properties. Based on this architecture, section 4 formally introduces the routing protocol. Section 5 concludes the paper.

2. DEFINITIONS
The following definitions as well as notations will be frequently used in the sequel.

Definition 1: (System Topology)
The system topology is a graph $G = (V, E)$, where $V$ is the set of nodes, and $E$ the set of logical edges. It is used to represent a radio network. There is only one transceiver in each node and the network operates in a half-duplex mode. A logical edge $(x, y)$ means that node $y$ is node $x$'s one-hop neighbor under the current transmitting power, and vice versa. Fig. 1 shows a topology of a multihop packet radio network.

![Fig. 1: The system topology](image)

Definition 2: (Distance of Two Nodes)
The distance $d(x, y)$ of two node $x$ and $y$ of $G$ is defined to be the minimal number of hops from $x$ to $y$.

Definition 3: (Cluster)
A cluster $C_i \subset V$ is a set of nodes, where for any two nodes $x, y \in C_i$, $d(x, y) \leq 2$. Namely, any two nodes in a cluster are at most two hops away. We define a cluster coverage $\{C_i\}$ of $V$, such that $V = \bigcup C_i$ and $C_i \cap C_j = \emptyset$, if $i \neq j$.

Definition 4: (Center and Radius of a Cluster)
The center of $C_i$ is defined to be the node $x_0$ such that $\min_x \max_y d(x, y)$, $x, y \in C_i$. $\max_y d(x_0, y)$ is called the radius of
nodes are moving and the topology is slowly changing. Other-
clustering scheme will tend to preserve its structure when a few
into small partitions (clusters)

Definition 5: (Degree of a Topology)
The degree of a topology is the number of clusters in a topology. It is denoted by $\Delta$.

Definition 6: (Repeater, Bridge and the Order of a Repeater)
For an edge $u = (x, y)$, $x$ and $y$ are called repeaters if they belong to different clusters, $u$ is called a bridge. The number of clusters which a repeater can reach in one hop is called the order of the repeater. Note that the order of a repeater includes the cluster which it belongs to. Thus, the minimal order of a repeater is 2.

Definition 7: (Bridge Partially in a Cluster)
If one of the two end nodes of a bridge $u$ belongs to a cluster $C$, then $u$ is said to be a bridge partially in $C$, and we write $u \in \omega^+(C)$. Similarly, we define an edge totally in $C$, if both of the end nodes of $u$ belong to $C$, and the set $\omega^-(C)$. Finally, the set of edges incident to $C$ is denoted by $\omega(C) = \omega^+(C) \cup \omega^-(C)$.

Fig. 2 shows a possible clustering of the topology in Fig. 1. In particular, $C_1 = \{0, 1, 2, 3, 4\}$, $C_2 = \{5, 6, 7, 8\}$, and $C_3 = \{9, 10, 11, 12\}$, $\Delta = 3$. The centers of $C_1$, $C_2$, and $C_3$ are node 0, 5, and 9 respectively. The radii of the three clusters are all 1. Node 1, 2, 6, 7, 10, 11 are repeaters. The orders of all repeaters are 2. $\omega^+(C_1) = \{(1,6) (1,7) (2,10)\}$, $\omega^+(C_2) = \{(1,6) (1,7) (6,11)\}$, $\omega^+(C_3) = \{(2,10) (6,11)\}$.

![Fig. 2: Clustering nodes](image)

3. THE MULTICLUSTER ARCHITECTURE AND ITS PROPERTIES

Most hierarchical clustering architectures for mobile radio networks are based on the concept of clusterhead [1, 5]. The clusterhead acts as a local coordinator of transmissions within the cluster. It differs from the base station concept in current cellular systems in that it does not have special hardware and in fact it is dynamically selected among the set of stations. However, it does extra work with respect to ordinary stations, and therefore it may become the bottleneck of the cluster. To overcome these difficulties, in our approach we abandon the clusterhead approach altogether and adopt a fully distributed algorithm [7-8].

The objective of the proposed clustering algorithm is to find an interconnected set of clusters covering the entire node population. Namely, the system topology $G(V,E)$ is divided into small partitions (clusters) with independent control. A good clustering scheme will tend to preserve its structure when a few nodes are moving and the topology is slowly changing. Otherwise, there will be a lot of overheads to reconstruct clusters. Within a cluster, it should be easy to schedule packet transmissions and to allocate the bandwidth to real-time traffic. Since there is no notion of clusterhead, this permits us to avoid vulnerable centers and hot spots of packet traffic flow.

3.1. The Centralized Clustering Algorithm

We first introduce a centralized version of the clustering algorithm. Initially, node IDs are used to construct clusters in which any two nodes are two hops away at most. After initialization, the cluster structure is maintained and updated based on node connectivity. The centralized clustering algorithm is shown below. Let $\Gamma_i(x)$ be the set of one-hop neighbors of $x$.

Centralized Clustering Algorithm:

0. $i = 0$.
1. $x = \min(V)$.
2. $C_i = \{x\} \cup \Gamma_i(x)$;
   $V = V - C_i$;
   $E = E - \omega(C_i)$.
3. If $V \neq \emptyset$, then $i = i + 1$ and goto 1; else stop.

![Fig. 3](image)

In the above algorithm, $\min(V)$ returns the node with the lowest ID in the set $V$. Obviously, the algorithm constructs clusters with radius (see Definition 4) at most 1. So, any two nodes in a cluster are at most two hops away. For example, Fig. 2 shows the result of applying this algorithm to the set of nodes in Fig. 1.

![Fig. 4](image)

We simulate the clustering algorithm by placing $N$ nodes randomly in a $100 \times 100$ area and we measure some of its properties. We assume two nodes can hear each other if their distance is within a predefined transmission range. We begin studying the impact of transmission range on connectivity. In Fig. 3, we find that adjusting transmission range will increase the con-
nectivity of the system. The connectivity is defined as the fraction of node pairs which can communicate through single or multiple hops. From the point of view of connectivity, the reasonable transmission range should be more than 30 for $N = 40$, and more than 40 for $N = 20$.

Next, we study the characteristics of the clusters generated by the algorithm. To this end, we measure the average size of the clusters (i.e. the average number of nodes in a cluster) and the number of clusters (the degree of a topology, $\Delta$). They are controlled by the transmission range. In Fig. 4 and Fig. 5, the larger the transmit power, the more nodes a cluster contains, and the fewer clusters are created. Repeaters relay packets from one cluster to another. Every repeater is time-shared among a set of clusters, since it has only one transceiver which operates in a half-duplex mode. So, the order of a repeater (see Definition 6) should be as small as possible in order to achieve higher throughput (the minimal order of a repeater is 2). Fig. 6 shows the average order of repeaters versus the transmission range. From this figure, we find that the order of most repeaters is either 2 or 3 based on this clustering algorithm.

The existence of at least one path between a pair of nodes is important in WAMIS. The number of repeaters and the number of bridges between an adjacent cluster pair will affect the number of paths. In Fig. 7, we note that more than 50% of nodes are repeaters if the transmission range is over the interval $(30, 80)$. In Fig. 8, we can find the average bridges between an adjacent cluster pair are more than 5 if the transmission range is greater than 40.

Our cluster structure provides a very robust infrastructure, which is not easily disrupted by mobility. We measure the stability of the multicenter architecture by assuming that nodes move and by counting how many nodes switch from one cluster to another (or construct a new cluster) within a time tick. For example, Fig. 9(a) is the original cluster. After one time tick, Fig. 9(b), the original cluster structure is destroyed, since $d(1, 4) = d(0, 4) = 3 > 2$. In the situation of Fig. 9(b), the node with the highest connectivity (in case of a tie, the lowest ID) (node 2 in this example), and its neighbors $\Gamma_1(2)$ (node 0, 1, 3) do not switch cluster. The other nodes (node 4) must leave the current cluster. Fig. 10 shows the stability of this cluster updating procedure. The mobility model we employ is that the direction of node motion is uniformly distributed over the interval $(0, 2\pi)$, and the distance over $(0, e)$ at each time tick. From Fig. 10, we note that the average number of nodes which switch clusters per time tick is relatively small. It leads to better stability than the scheme reported in [5]. It also provides easier adaptation to topological changes.

The detail implementation of this centralized algorithm in a distributed manner is shown in [7].

4. ADAPTIVE ROUTING FOR REAL-TIME TRAFFIC

When real-time traffic is considered to transmit over the dynamic network, the objective of a routing protocol is to keep communication going. Routing optimality (e.g., shortest path) is of secondary importance. Furthermore, the routing protocol must be capable of establishing new routes for real-time sessions quickly when a topological change destroys existing routes.
We choose a hierarchical scheme over the multicluster infrastructure for routing. There are three main reasons that we use a hierarchical routing protocol. First, we attempt to reduce traffic by hiding information about the content of a cluster. Since every node has full information of its locality, it knows all the details about how to route packets within its own cluster. Second, hierarchical structure can scale to large populations. Third and most importantly, the multicluster infrastructure is quite stable in the mobile environment (see Fig. 10) and multiple links (bridges) connect two adjacent clusters (see Fig. 8). Therefore, hierarchical routing on top of the multicluster infrastructure is efficient and reliable.

4.1. The Two-level Hierarchical Routing Protocol

There are two logical phases in our routing protocol: route construction and route maintenance. The construction phase establishes an initial set of routes. The maintenance phase maintains loop-free routing in the face of arbitrary topological changes; it rebuilds new routes quickly when topological changes destroy all existing feasible routes.

4.1.1. Construction Phase

We assume every node has no global connectivity information. After clustering, a node, say $x$, keeps some information of its locality: the nodes in its cluster ($\text{cluster}(x)$) and 1-hop neighbors. Since every node has the information of its locality, it knows all the details about how to route packets to destinations within its own cluster, but knows nothing about the internal connectivity and the nodes in the other clusters. In order to discover a route between node $x$ and $y$, $x$, first, locates $y$’s cluster, and, second, select a cluster-level route out of the many possible that connect $x$’s cluster and $y$’s cluster.

Let the set of repeaters in a cluster which receive the packet from the upstream cluster be the input repeaters, and the set of repeaters which forward the packet to the downstream cluster be the output repeaters. For example, in Fig. 11, $x$ is an input repeater and $y$ is an output repeater. An input repeater can easily plan a path to an output repeater according to its local information. To implement hierarchical routing, every node has to maintain for each repeater in its cluster a list of nodes to which the repeater has one or more routes; this is known as the internal forward database (IFDB). Furthermore, every repeater has to maintain for each adjacent cluster a list of nodes to which the cluster has routes; this is known as the external forward database (EFDB).

Assuming that the IFDBs and EFDBs contain the intended destination, routing can be implemented as follows: For every cluster, if a packet is received from an input repeater $x$, then if the destination ID of the packet is in $\text{Cluster}(x)$, the packet is routed to the destination within the cluster. Otherwise, $x$ selects a repeater as the output repeater from its IFDB according to the destination ID and then forwards the packet to the repeater. Upon receiving the packet, the output repeater selects one suitable adjacent cluster from its EFDB as the downstream cluster and forwards the packet. For example of Fig. 12, IFDB of node 1 may be $((4, \{8, 15\}), (3, \{8, 16\}))$. So if the destination is node 15, then node 1 selects node 4 as the output repeater. EFDB of node 6 may be $((C_2, \{1, 18\}), (C_3, \{1, 4\}), (C_5, \{17\}))$. If the destination is node 4, node 6 forwards the packet to $C_3$.

When a source desires routes to a destination and its IFDB and EFDB do not include the destination, it broadcasts a control packet, the discovery packet to locate the destination’s cluster. After initiating a discovery packet, the source waits for reception of a reply packet. When the reply packet comes back from a downstream cluster, each node in the current cluster updates its IFDB. In addition, the repeaters which connect to the downstream cluster update their EFDBs. For example, in Fig. 13, node 1 would transmit packets to node 9. Node 1 initiates the discovery packet. Once the discovery packet reaches node 6, the reply packet is sent out by node 6 since node 6 knows node 9 in its cluster.

The discovery packets is flooded through the network in search of clusters which have such a route. By flooding, we mean a cluster-to-cluster broadcast such that no cluster broadcasts any individual discovery packet more than once. Each discovery packet has an unique sequence number which allows a node to remember if it has previously broadcasted or received the discovery packet. Clusters without routes that receive a
discovery packet forward the packet, while clusters, maybe not the destination cluster, with routes broadcast a reply packet. The reply packet is flooded back towards the source cluster (as well as towards to all nodes in the network) which initiates the discovery flood. From Fig. 3, we can find the reasonable transmission range from the point of view of connectivity is about 40. Thus, the average number of clusters in the network is about 6 (see Fig. 4). It is not too large. As a consequence, the flooding does not severely suffer a discovery/reply packet explosion and does not cause severe congestion.

Let the cluster-level topology be described to a graph in which a vertex represents a cluster and a link between two vertices means both clusters are adjacent. Each link of the graph is assigned a direction according to the direction of the transmission of the reply packet. The vertex sending the reply packet is the downstream vertex, and the vertex receiving the reply packet is the upstream vertex. Routes are built depending on the order of the reply packet transmission, but the final directed graph is always loop-free. Thus, the cluster-level topology would be an acyclic directed graph (ADG) with a special destination vertex.

4.1.2. Maintenance Phase
In addition to the route construction phase, it is necessary to have a route maintenance phase to cope effectively with link failures of existing routes. In the following discussion, we focus on a special destination node and let its ID be DID.

Let a repeater, say \( x \), in cluster \( C_1 \), and let \( C_j \) be a downstream cluster of \( C_1 \). When \( x \) loses its last route to DID due to the failure of link \((x, y)\), \( y \in C_j \), \( x \) deletes DID from the entry \( (C_j, \{\text{DID}, \cdots\}) \) in its EFDB. Then \( x \) sends an update packet to the other nodes in its cluster. Upon receiving such an update packet, a node, say \( z \), deletes DID from the entry \((x, \{\text{DID}, \cdots\}) \) in its IFDB. In addition, if \( z \) is a repeater and it finds every repeater in its cluster has no route to DID (i.e. IFDB of \( z \) does not contain DID), it sends an update packet to its upstream clusters. Note that the maintenance phase only allow to operate deletion to both EFDB and IFDB, while the construction phase only allow to operate insertion to both.

We refer to the VC of a real-time session as the primary route which is selected among feasible routes according to some criteria. The maintenance phase is to maintain the primary route. For example of Fig. 14, suppose that the primary route of a real-time session is from cluster \( C_0 \) to cluster \( C_n \). Once the last link between \( C_1 \) and \( C_2 \) goes down, \( C_1 \) selected another downstream cluster \( C_2' \) to route packets to \( C_n \). If \( C_1 \) loses all of downstream clusters, \( C_0 \) chooses another feasible downstream clusters \( C_1' \) for routing. As the source cluster \( C_0 \) loses the last link to its last downstream cluster, it has to initiate the route construction mechanism to rebuild the ADG for \( C_n \).

A cluster may be destroyed in the face of arbitrary topological change and then the reclustering algorithm is triggered. Routes through such a cluster may be affected. The example in Fig. 15 shows the effect of reclustering and illustrates how to establish a new route when necessary. Fig. 15(a) shows a part of the primary route in which cluster \( C_2 \) forwards packets from \( C_1 \) to \( C_3 \). If \( C_2 \) is reclustered and split into two clusters, there are two possible effects to the primary route. First, shown in Fig. 15(b) and (c), the new clusters can still connect \( C_1 \) to \( C_3 \). In this case, the primary route in cluster level need not be rebuilt, while the route within \( C_2 \) may be changed. Second, shown in Fig. 15(d), there exists no route from \( C_1 \) to \( C_3 \) through the new clusters generated from \( C_2 \). Thus, \( C_2 \) in the figure has to seek a feasible downstream cluster, or backtracks to \( C_1 \) to find another route. The primary route is always kept loop-free in the maintenance phase.

5. CONCLUSIONS
In this paper, we provide a distributed multicluster architecture for transporting real-time traffic in a multihop dynamic radio networks. This architecture is not constrained by a fixed infrastructure and, rather, it can be deployed in an environment without infrastructure at all. A hierarchical routing protocol over the architecture is stable and loop-free in the face of topological change. Real-time sessions can use this protocol to set up and maintain virtual circuits.

REFERENCES