

# Clique Activation Multiple Access (CAMA): A Distributed Heuristic for Building Wireless Datagram Networks

C. David Young  
James A. Stevens

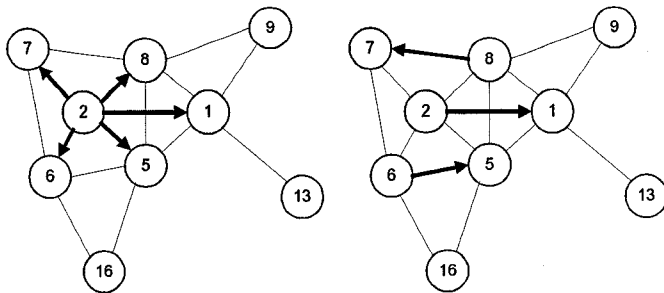
Rockwell Avionics & Communications  
Richardson, Texas

## ABSTRACT

The Soldier Phone program at Rockwell uses a combination of channel access techniques to transmit voice, video, and data over a dynamic, distributed mobile radio network. The current algorithms are briefly described and then an interesting clique based heuristic is proposed and examined in detail.

## INTRODUCTION

Traditionally, there are two primary types of channel access schemes. The first, known in the literature as "node activation", has a transmitter broadcast to all of its neighbors at once rather than individually. When a transmitter has only one intended receiver it is known as "link activation". The former allows only one active transmitter in a neighborhood while the latter can potentially have several, as depicted below.



Node Activation

Link Activation

Figure 1. Node and Link Activations

Node activation is especially well suited for broadcast messages like those used for address resolution. Link activation on the other hand lends itself better to high volume point-to-point traffic where allocations are made along the path of the traffic for the duration of a session. Thus, Soldier Phone has created a hybrid network using node activations for low volume control traffic and occasional datagrams and link activations for high volume point-to-point traffic. The combination of these channel access techniques is called ODMA (Orthogonal Domains

Multiple Access) because it makes use of both the frequency and time domains.

The Unifying Slot Assignment Protocol (USAP) [1] facilitated the implementation of the node and link activations in Soldier Phone, which will be described first. Then, in an attempt to more efficiently use channel resources, a replacement for node allocation is considered. Based on cliques, it is similar to node activation in its use of multiple receivers but is different in that it allows time slots to be shared among neighboring nodes.

It will be useful to have a way of referring to a particular choice of RF channel used in a specific time slot. Thus, the word "allocation" will refer to an ordered pair of time slot and frequency channel.

## SOLDIER PHONE SLOT ASSIGNMENT HEURISTICS

When a node chooses an allocation, USAP enforces certain constraints to avoid interference within 2 hops of the transmitter. For link activation from node  $i$  to neighbor  $j$  it must be an allocation:

- that has not already been assigned to either node
- $i$ 's neighbors are not receiving in
- $j$ 's neighbors are not transmitting in

For node activation a node  $i$  must choose an allocation:

- that has not already been assigned to node  $i$  or any of its neighbors
- none of  $i$ 's neighbors' neighbors are transmitting in

The above constraints can be modeled in graph theory as a distance 2 vertex-coloring problem; that is, in an undirected graph no vertices connected by 1 or 2 edges have the same color. In Soldier Phone the colors are applied to both time slots and frequency channels so transmitters within 2 radio hops of each other must color

their slots or channels (or both) to prevent interference at their intended receivers.

Indeed, for node activation the half-duplex Soldier Phone transceivers within 2 hops of each other must choose different slots for the sake of the neighbors they have in common. Thus, transmitters using the same slot must be at least 3 hops apart. But if they are 3 hops apart then they are allowed to have the same color so they might as well use the same frequency. This implies that node activation cannot benefit from simultaneous transmissions on multiple channels. In other words, frequency plays no useful role in coloring. Thus, the sample network shown below is colored with 7 slots (shown in parentheses).

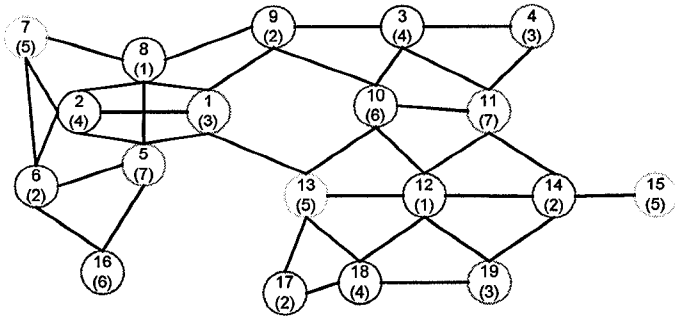


Figure 2. Sample Node Activation Coloring

### A CLIQUE BASED ACTIVATION HEURISTIC

To achieve higher efficiency, it might be desirable to allow multiple transmitters in a neighborhood to share the same slot to utilize it more effectively. One such channel access technique would have the nodes sharing a slot simply take turns transmitting in the slot. This is easy to implement and scales to arbitrarily large groups sharing the same slot (albeit at the expense of arbitrarily large latencies.) Another more efficient technique is called "communication groups" (CGs), which defines a random access non-persistent CSMA protocol[2]. It allows groups of fully connected nodes to contend for a slot via minislots at the beginning. A minislot is assigned to each of the members of a CG and its order is rotated with respect to the others to provide fairness.

In order for a group of transmitters to share a slot, they must all be neighbors of each other, that is, they must form a clique or a complete subgraph. The example network is redrawn in the next figure to show all possible cliques.

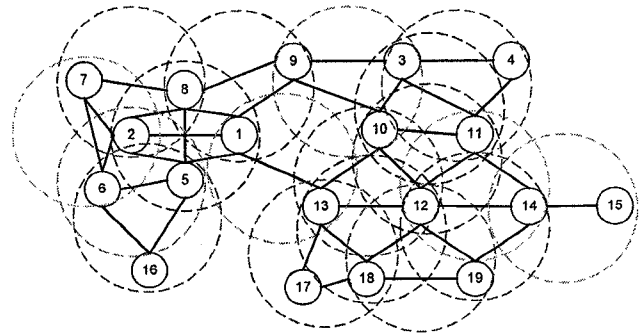


Figure 3. Cliques in Sample Network

In this example, observe that if clique 7-2-8 uses one color, clique 8-1-9 must use another, and 9-10-3 yet another to prevent interference. That is, if node 7 were to transmit at the same time on the same channel as node 9, node 8 would miss node 7's transmission. Thus, if one views the network from the clique perspective, it is still a distance 2 vertex-coloring problem. This is represented in the graph below where each clique becomes a vertex and the edges show the connectivity between the cliques. The letters identify the cliques while the numbers below them represent colors. Observe that this graph can be colored with no more than 10 colors, which is still more expensive than the node activation heuristic requiring 7. A later optimization will reduce that to 5.

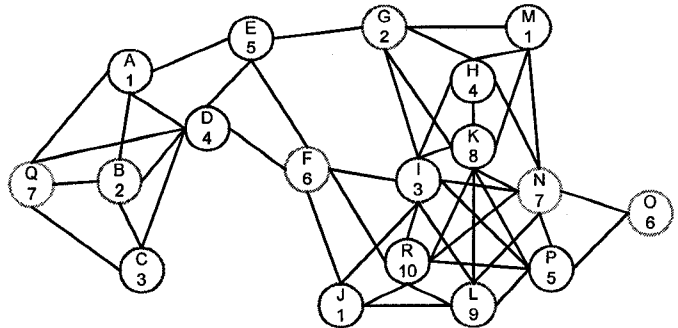


Figure 4. Clique View of Network

A potential advantage to the clique heuristic is that frequency could play a useful role in coloring. That is, while neighboring cliques must be on different time slots, cliques 2 hops apart can be on the same slot as long as they are on different channels. However, for many topologies this will make no difference since a lower bound on the number of slots required to color every clique is the maximum degree of any clique vertex plus 1. For instance, clique I has 9 neighbor cliques so at least 10 slots are required. With this many slots it is possible to color the entire graph without resorting to different frequencies. Therefore, as in the node activation, it might make sense to use a single channel and leave the other channels reserved for link activation.

## ALGORITHM

Cliques can be created using a list of neighbors and a list of each neighbor's neighbors, which requires additional overhead compared to USAP. To generate the cliques a node is a member of it considers all combinations of its node id with those ids of its neighbors (using its neighbor list) and examines each combination for complete connectedness (using each neighbor's list of neighbors). One such algorithm involves a depth-first walk of the tree rooted at the node id whose branches represent a systematic growth of every possible clique by adding neighbors in every possible order (i.e. every combination and permutation). Since this has time complexity  $O(n!)$ , two optimizations are made to trim fruitless branches and reduce this to  $O(2^n)$ :

- 1) Create a branch only when adding a neighbor of all the current clique members.
- 2) Neighbors are added in increasing order of node id and a branch that has already been generated by another permutation of the same neighbors is trimmed.

As an example, this algorithm has been applied to the following network to find the cliques that node 2 belongs to.

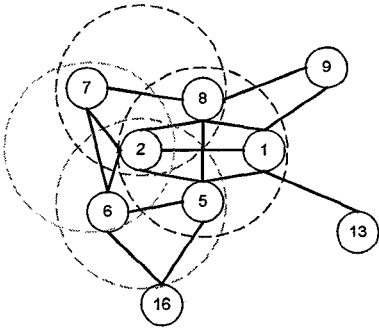


Figure 5. Cliques of Node 2

The resulting tree is shown in the following figure with the cliques boxed. Some cliques are in parentheses that were previously found and other branches are underlined that were previously traversed. Thus, the cliques containing node 2 are found in the following order:  $\{1,2,5,8\}$ ,  $\{2,5,6\}$ ,  $\{2,6,7\}$ , and  $\{2,7,8\}$ .

## PROOF OF CORRECTNESS

First, an induction proof ignores the optimizations and uses the tree structure resulting from the algorithm. If the root is considered level 1, the root with each of its neighbors level 2, and so on, the following assertion can be made:

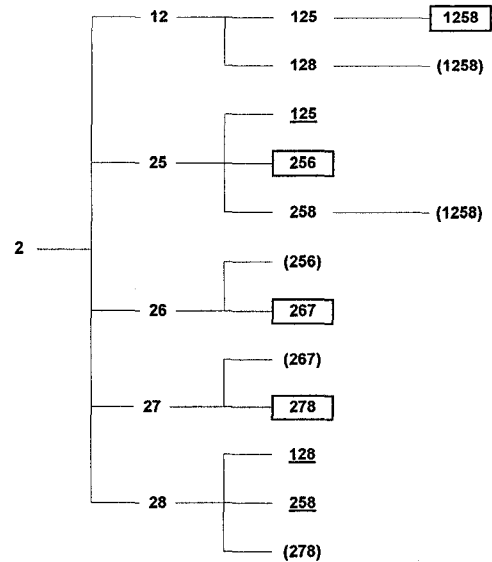


Figure 6. Clique Search Tree

**Lemma:** At each level  $L$  all cliques of size  $L$  are found.

[IB] At level 1 the root node itself forms a trivial clique. Also, at level 2 the root node and each of its neighbors in turn form all cliques of size 2 that include the root node.

[IS] Assume at level  $n$  that all cliques of size  $n$  have been found. This assumption means that the cliques at level  $n+1$  must be supersets of the cliques at level  $n$ . Now, observe that the children nodes at level  $n+1$  are formed by adding each neighbor not already part of the parent clique in turn to each parent clique at level  $n$ . Since this generates all possible supersets of size  $n+1$ , all possible cliques of size  $n+1$  must be found.

**Theorem:** The CAMA algorithm for finding cliques finds all possible cliques which the root node is a member of.

This is proved by observing that all nodes of the tree can be visited using standard techniques, for example, a preorder tree walk. Since the height of the tree is limited by the number of neighbors the root has, there are a limited number of nodes to visit and by the Lemma, all cliques must be found. Optimization 1 does not change this result because it just trims subtrees that result from adding neighbors that are not eligible members of the current clique. Optimization 2 does not change this result because it just trims subtrees that are identical to a subtree that has already been traversed. The subtrees are identical because the only inputs to the algorithm are the members of the root clique.

## IMPLEMENTATION

The implementation is a traditional recursive depth-first tree walk with the addition of the two optimizations. A neighbor id is represented as a bit position in an array so that ORing can create a set using the member id's to allow fast manipulations and comparisons of neighbor lists and cliques. The label on each branch is in fact the bit map of the current clique.

The first optimization (adding only eligible neighbors to the current clique) results from ANDing together the neighbor lists of the current members. The second optimization (trimming subtrees that have already been traversed) results from adding the neighbors in order of increasing id. This guarantees that subtrees of a particular height will be searched in order of the roots' labels, making it possible to recognize duplicate subtrees by merely keeping track of the largest label encountered at a particular level.

Thus, the pseudocode of the algorithm for finding the set of cliques  $C$  which node  $i$  belongs to is as follows:

```
//
// Start off with the set of cliques C empty except
// for the trivial clique consisting of i alone. Use
// array L to keep track of the largest label found
// at each level. Call the function FindCliques() to
// generate the cliques.
//
set C = {{i}}
set array L[1..max # neighbors] = {{}, {}, ..., {}}
FindCliques({i})
//
// A recursive procedure for adding new members to
// the given clique S
//
FindCliques(set S)
{
  set N = {1..1}
  ∀n ∈ S
    N = N ∩ {neighbors of n}
  ∀n ∈ N in order of smallest n to largest
  {
    // Enlarge S by adding the neighbor
    S' = S ∪ {n}
    // If the label is larger than any encountered
    // at this level, enlarge this clique
    // and recurse into this subtree.
    IF S' > L[|S|]
    {
      L[|S|] = S'
      // Delete S from C if it exists and add the
      // enlarged S back in
      C = C - {S}
      C = C ∪ {S'}
      // Try to enlarge clique even more
      call FindCliques(S')
    }
  }
}
```

## TIME COMPLEXITY

In the process of finding all maximal size cliques, this algorithm generates all cliques of smaller sizes. This is equivalent to summing binomial coefficients, yielding  $2^n$  such cliques in a fully connected network of size  $n$ , which is the worst case running time for this algorithm. Thus, the time complexity is at least  $O(2^n)$ . This is not surprising since the problem of finding a clique of maximum size in a graph is NP-complete. This does not prevent the algorithm from being useful for the topologies typically encountered by Soldier Phone, where a neighborhood of 16 nodes or less is normal. Denser networks can be handled by limiting the number of neighbors that any node will accept.

## TIME SLOT ASSIGNMENT

Assuming that CAMA will be restricted to a single channel, the slot assignment is as follows. It occurs in 3 synchronized phases at each node:

- 1) measure neighbor qualities
- 2) distribute the results from phase 1 and calculate the cliques and choose slots
- 3) use the slots from phase 2

Note that in phase 2 each member of a clique independently chooses the slot to be used for that clique, so that they must have exactly the same information to arrive at the same choice. This information is just that required to enforce the distance 2 vertex-coloring of the clique graph. It is shared between nodes as part of the net management packet (NMOP) broadcast periodically by each node to all of its neighbors:

- 1) the set of neighbors (N) along with their measured qualities from phase 1
- 2) the set of this node's cliques (C) with the chosen slots (CS)
- 3) the set of this node's neighbors' cliques (NC) with the chosen slots (NCS) (exclusive of those already listed in item 2)

In order to choose a slot, the members of the clique must generate the set of slots that are already in use within 2 clique hops. It does this by combining its CS with the CS's and NCS's from all of its neighbors. If a clique has not yet been assigned a slot, the nodes of this clique pseudo-randomly choose from among the available slots.

This random choice of slots will occasionally result in exceptions to the distance 2 vertex-coloring rule. Exceptions can also result from the measures taken to limit the size of neighborhoods in dense networks. However, the conflicts will be obvious because a node will either end up choosing the same slot for two cliques or a neighbor will report a CS with the same slot for a different clique. When a node detects this condition, it should drop the conflicting slot choice from its CS, which will cause the other nodes in the affected clique to drop the slot also. Then the next phase 2 will cause a new pseudo-random choice.

### OPTIMIZED TIME SLOT ASSIGNMENT

Trying to assign slots to all cliques in no particular order increases the risk of running out of slots and leaving the net partitioned. To decrease the chance of this happening, the slots can be assigned in a certain order to try to include all nodes in at least one clique before assigning slots to the remaining cliques. The idea is to first assign slots to cliques that have an isolated node on the edges of the network, then to assign slots to the most richly connected cliques in the interior, and then to assign slots to cliques that bridge these. Thus, a clique is assigned a slot under the following conditions and in the following order:

1. The clique has a node that is only a member of one clique.
2. The clique has more or an equal amount of neighboring cliques than any neighboring clique.
3. The clique has 2 or more neighbors assigned slots in steps 1 and 2.
4. The clique has 2 or more neighbors assigned slots in step 1.
5. The clique has a node that has not been covered by a clique from the previous steps.
6. The clique has not yet been assigned a slot in the previous steps.

Notice that the application of these rules requires no more knowledge than is already available in the NMOP, namely the sets of this node's cliques and this node's neighbors' cliques. Applying just steps 1 to 3 to the example network results in all of the nodes being covered as shown below.

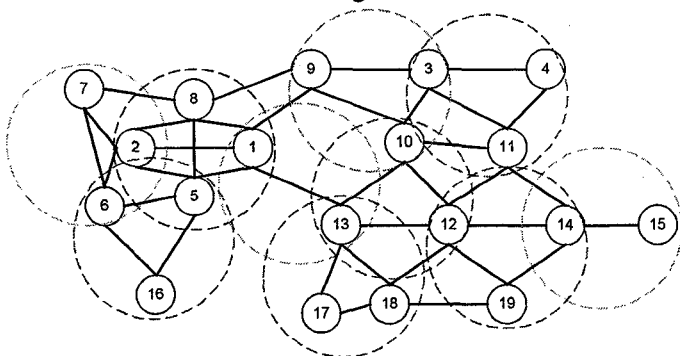


Figure 7. Optimized Clique Cover

This is reflected in the clique graph below where the colors are followed by the step applied in parentheses.

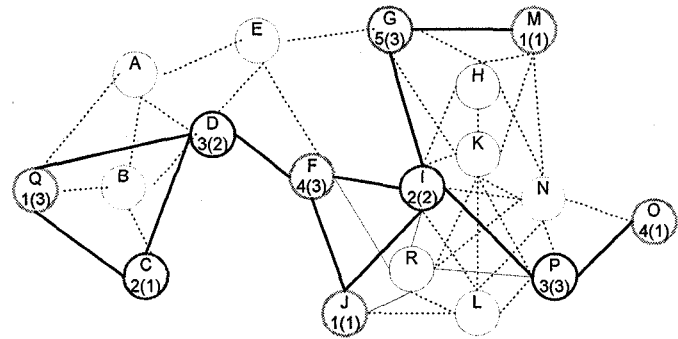


Figure 8. Optimized Clique View

Notice that only 5 colors are required compared to the 7 colors of the original node allocation heuristic. If a total of 10 slots were available, the remaining 5 slots assigned during the application of the subsequent steps could serve to enrich the connectivity of the mesh for the sake of robustness. Another option is to leave the remaining slots unassigned to make link allocations on the other channels possible.

### CONCLUSION

This paper proposes a new type of channel access, CAMA, which allows neighboring nodes to form cliques for the purpose of supporting a broadcast channel. CAMA is attractive because it operates using only local knowledge, meaning it can scale to large networks. It also has the potential of requiring less channel resources than node allocation. In addition, it would lend itself well to an application like push-to-talk voice where only one transmitter per neighborhood is active (except for relay nodes). The downside is that its implementation is quite complex, it has higher protocol overhead than a USAP based heuristic, and it requires a considerable amount of optimization to meet or exceed the performance of simple node allocation.

### REFERENCES

- [1] C. D. Young, "USAP: A Unifying Dynamic Distributed Multichannel TDMA Slot Assignment Protocol", Proc. IEEE MILCOM'96, Vol 1, October 1996
- [2] R. C. Sunlin, "A Hybrid Distributed Slot Assignment TDMA Channel Access Protocol", Proc. IEEE MILCOM'90