Efficient Wireless Multicast Protocol with Orthogonal CTS Modulation Supporting Video Conferencing

Ju Wang, Hongsik Choi, Esther A. Hughes
Virginia Commonwealth University
Email: jwang3,eahughes, hchoi@vcu.edu

Yong Tang
University of Florida
Email: yt1@cise.ufl.edu

***Abstract*— Multicast based video conferencing in wireless networks is recently receiving more and more attention. In this paper, we propose a novel linker-level protocol, Multicast with Orthorgonal CTS (MOCTS) protocol, to provide efficient wireless multicast. Our approach uses orthogonal Walsh code to modulate CTS packets at different receiving nodes. The modulated CTSs are simultaneously transmitted and decoded at the source node, thus considerably reducing multicast CTS collisions. Simulation results show that the MOCTS protocol can reduce more than 20% in total transmission efforts.**

I. Introduction

Multicast is a promising technique to support video conferencing in wireless networks. However, compared to multicast in wireline networks [6]–[8], multicast in wireless networks must address new challenges due to the wireless advantage, the movement of wireless terminals (nodes) [9] and the power saving requirement [5]. Due to the shared media nature, wireless transmissions in open air will inevitably interfere to each other.Efficient Resolving of multicast-CTS collisions is critical [3] [1] [10].

In this paper, we propose a novel wireless multicast scheme based on an efficient, but completely different, method that can quickly resolve multicast CTS collisions. Our proposed protocol is motivated by the success of Code Division Multiple Access (CDMA) technology in the cellular network where nodes transmit in parallel. The basic idea is to allow the multicast source to process multiple CTS packets concurrently to reduce the signalling time. To achieve this, destination nodes will modulate their CTS packets with different Walsh codes and transmit these CTSs simultaneously at a synchronized slot. Since Walsh codes are (ideally) orthogonal to each other, the superimposed signal picked up by the source node can be successfully demodulated for each individual nodes. We show that a distributed implementation of such an idea requires some signalling procedure to resolve the Walsh code collisions among nodes. The expected average collision resolution time is analyzed and augmented by simulation results.

We then propose a distributed-local-gain-maximizing (DLGM) algorithm to control the propagation of multicast packets and to take advantage of the network level wireless advantage. The algorithm will determine the next node to transmit based on local topology. Each node keeps track the receiving status for all nodes within its 2-hop neighborhood. Each node will then calculate a multicast-gain for all direct neighbors that are eligible for relay. If a node finds its own multicast-gain to be not the largest among its neighbors, it will yield and postpone local relay. Our simulation shows that such a greedy one-hop hueristic can reduce the total multicast energy expenditure up to 20%.

The rest of this paper is organized as follows: Section 2 shows preliminary analysis about the importance of reducing the multicast signalling process. In Section 3, we present the MOCTS scheme based on a one-cell scenario and effect of collision salvage; Section 4 discusses the extension to multi-hop network and the DLGM algorithm, with simulation results. Section 5 concludes this work.

II. One-hop MOCTS Multicast Protocol

We first present the signalling protocol assuming all nodes can send/receive from each other. A multicast session starts with the S-node (source node) sending out an MRTS (multicast request-to-send) packet. The S-node then wait to collect the MCTS (multicast confirm to send) packets from all destination nodes. When receive an MRTS and ready for receiving data, a D-node (destination node) will randomly select a Walsh code to modulate its MCTS packet and send it to the S-node.

A key part of our protocol is that all ready D-nodes will transmit synchronized MCTS packets simultaneously. Due to the orthogonality of Walsh codes, the MCTS packets from different D-nodes can be decoded if their modulating Walsh codes are different. Thus, ideally, if all D-nodes choose their Walsh code differently, there will be no collision and the S-node will receive all required MCTS within one slot time.

However, there is always a possibility that some D-nodes will use the same Walsh code, which will cause a code collision and packet detect errors at the S-node (e.g., CRC errors) corresponding to the colliding code. To resolve code collision, the S-node might have to retransmit MRTS several times to collect replies from all D-nodes. The protocol for the S-node is shown below. Due to the space limitation, the D-node protocol is not presented in detail.

Notice that only the colliding D-nodes will be required to re-select a new Walsh code. When re-selecting Walsh code, a D-node (say n_1) might select a code w which is also picked by another D-node (say n_2)in previous round. This will not be treated as a collision since n_2 will not reply to the retransmitted MRTS. An important parameter is WSIZE— the number of Walsh codes to be selected at the D-nodes. When WSIZE is greater than the number of D-nodes, the system will converge to a non-collision status after some MRTS/MCTS exchange. When in non-collision status, all D-nodes have a unique Walsh code.

```
S-Node Protocol
  $st \in \{READY, MRTS, rMRTS, DATA,$
  $T_0 :$  $timer;\ recv_count : integer;\ WSIZE : integer$

        (st==READY) and (have M-packet in queue)
           send MRTS
           set $T_0 = 1\ MCTS\ slot$ and start $T_0$
           sense media for activity
           $\dashrightarrow st := MRTS$
        (st == MRTS) and ($T_0$ not expired) and (received MCTS packets)
           decode all MCTS packets
           if (all expected MCTS arrived with positive reply)
done:            $\dashrightarrow st = DATA$
              send data packet
           if (there is code collision in some MCTS packets)
              $\dashrightarrow st = rMRTS$
              recv_count = number of MCTS that is
                    successfully decoded.
retry:        determine the new WSIZE
              append the code id of the colliding Walsh code in
              the MRTS packet.
              reset $T_0$ and re-send MRTS packet
        (st == rMRTS) and ($T_0$ not expired) and (received MCTS packets)
           num= successfully decode MCTS packets
           recv_count = recv_count + num
           if (recv_count == number of receiving nodes)
               goto done
           else goto retry
```

A. *Synchronization of D-nodes*

Synchronization among a group of mobile terminal in general is a difficult task(this is one of the reason to use PN code in the reverse link at CDMA cellular system). Fortunately, it is relatively easy to achieve when mobile nodes are displaced in relatively small areas. We need to synchronize N senders (D-nodes) and one receiver (S-node). Obviously, only one synchronization signal is allowed among those $N+1$ nodes to avoid ambiguity. Two choices are possible: (1) let the multicast sender send out the synchronization signal (pilot signal), or (2) one of the receiving nodes provide a synchronization signal. In both methods, by the end of synchronization, all receiving nodes will directly transmit their modulated MCTS. The former method is simple since the synchronization signal can readily be provided by the S-node. For the second method, the S-node must pick one D-node which will provide the synchronization tone. The S-node can send the node identification in its MRTS packet.

B. *Collision Salvaging*

In the protocol discussed above, when two or more nodes choose the same Walsh code w, the source node will simply decide there is a collision on w and ask the involved nodes to select a new Walsh code. This strategy might lead to many retransmissions of MRTS/MCTS when the number of node is large and the number of Walsh codes is relative small. To improve system performance under such circumstances, useful information can be extracted from collision cases to reduce the number of MCTS retransmission, which we call this **collision salvaging**.

The basic idea of **collision salvaging** is to distinguish different degrees of Walsh-code-collisions. For example, assume two nodes i and j choose Walsh code w, and both nodes send back positive MCTS. After spread spectrum demodulation, the signal strength of the collided MCTS will be twice as high as that of the normal non-collision case. The S-node thus can conclude that two nodes sent back a positive reply. If all other nodes choose a different Walsh code for their MCTS, the S-node can further conclude that all required MCTSs have arrived and proceed to data transmission. In this example, the signalling part only take one MRTS/MCTS handshake despite that fact that node i and j have a code collision.

Collision salvaging can effectively reduce the number of MCTS retransmissions. In fact, it is not difficult to see that most collisions involve only a few nodes. The probability of having l nodes colliding on one particular Walsh code, given a total of L Walsh codes, is $p_{L,l} = (\frac{1}{L})^l$. The probability of 4-node collision is, thus, only $(\frac{1}{L})^2$ of the 2-node collision. We thus expect that this improvement can eliminate most of MCTS retransmissions. This claim is confirmed by our simulation results in later sections.

III. Performance Analysis and Simulation Results

Let m be the number of receiving nodes and k be the number of Walsh codes used. We are interested in evaluating $T(k,m)$ – the average number of CTS collisions per multicast packet. $T(k,m)$ is equivalent to the code dissemination time to assign a unique code to each member of the multicast group.

$T(k,m)$ can be represented as the following recursive form

$$T(k,m) = \sum_{i=0}^{m} P_m^k(i)(1 + T(k-i, m-i)) \qquad (1)$$

where $T(j,0) = 0$ for all integer j, and $P_m^k(i)$ is the probability of the event where $i(out\ of\ m)$ nodes successfully select distinguishing codes, where the size of code pool is k. The term $P_m^k(i)(1+T(k,m-i))$ on the right hand represents the events that i nodes pick different Walsh codes and $(m-i)$ nodes have code collisions. The $(m-i)$ colliding nodes must re-select Walsh codes from a pool of k codes. The slot number for the $(m-i)$ nodes is $T(k,m-i)$.

We can rewrite $T(k,m)$ as

$$T(k,m) = \frac{\sum_{i=0}^{m-1} P_m^k(i)(1 + T(k,m-i) + P_m^k(0)}{1 - P_m^k(0)} \qquad (2)$$

$P_n^k(i)$ is calculated in [2] using random assignment model.

$$P_n^k(i) = \frac{(-1)^i k! m!}{k^m i!} \cdot \frac{\sum_{j=i}^{m} (-1)^j (k-j)^{m-j}}{(j-i)!(k-j)!(m-j)!} \qquad (3)$$

The performance of the protocol is largely decided by the number of Walsh codes and the nodes population. In general, the average collision resolution time increases rapidly for small $WSIZE$. For $WSIZE = 2$, the average MCTS packet to resolve collision already exceeds 119 when m increase to 10. Given 10 D-nodes, it requires 8.29 MCTS packets for $k = 4$, and 3.6 for $k = 8$. The $k = 4$ case reaches the saturation point at 43 nodes, and the $k = 8$ case saturates at 89 D-nodes.

Figure 1(a) shows the results for cases with WSIZE= 16, 32 and 64. Figure 1.(b) shows the average collision times when collision salvaging is deployed in the protocol. In this simulation, all 2-nodes collisions are treated as a non-collision. The reduction in the average collision times is significant. For

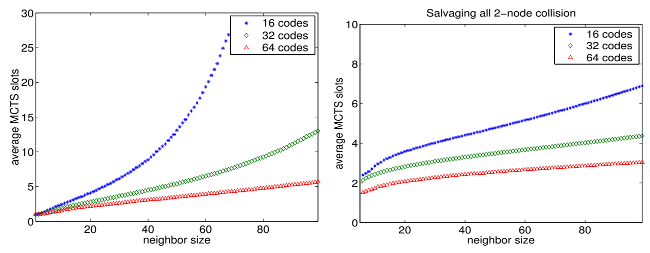


Fig. 1: Average collision slots(a) Simulation results with 16, 32 and 64 codes, (b)with collision salvaging: all 2-nodes collision are salvaged

$k = 16$, the average collision time is 2.4 at 5 nodes, and 7.2 with 99 nodes. For $k = 64$, the number of average collision time is even smaller and increases very slowly as the node size increases. Evidently, the collision salvaging technique is quite effective even when only 2-node collision is considered.

IV. Extending the Protocol to Multi-hop Network

A. *Multicast Deadlock (M-deadlock)*

Intuitively, the MOCTS protocol can be readily extended to multi-hop networks with the following modifications: (1) Execute the MOCTS protocol at all nodes independently, with only the multicast source node take the S-node role initially, and (2) When a D-node successfully received a multicast packet, it will switch to the S-node mode and relay the multicast packet. However, one must be careful to avoid over-flooding, and prevent deadlock as well. As shown by the example below, deadlock might happen among relaying nodes.

The following deadlock example assumes that each D-node only reply to the first MRTS, a rather conservative one. The network topology and MRTS/MCTS exchanging are shown in Figure 2(a). The multicast is initiated at the source node a, and successfully received by nodes b1, b2, and b3. Notice that nodes b1, b2, and b3 are not connected, thus they all will independently decide to further broadcast the packet out. Without loss of generality, let b1, b2 and b3 send out their first MRTS at time 0, 0.5 and 1 slot time respectively. As a result, b1's MRTS collide with b3's MRTS at c3, and b2's MRTS collide to b3's MRTS at c2. Only c1 successfully receive b1's MRTS and send back a MCTS. After a random wait, b2 will re-transmit MRTS and receive a MCTS from c2, and finally b3 will re-transmit MRTS and receive a MCTS from c3. Thus, b1, b2 and b3 each has one MCTS from c1, c2 and c3 respectively. Yet each of the three b nodes needs one more MCTS to transmit the data, constituting a deadlock situation.

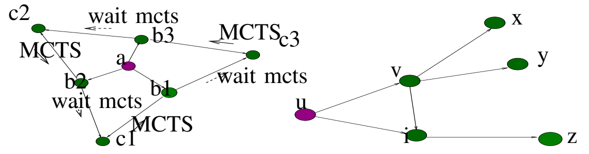


Fig. 2: (a)Multicast deadlock in a multi-hop wireless network,(b)maximizing local multicast gain example

Our revised relaying procedure consists of two algorithms to achieve deadlock avoidance:

On Receiving an MRTS

1) If this is the first MRTS that v received, mark the sender u as Associated-S-Node and send a MCTS to u.
2) If the MRTS received is a re-transmission MRTS from A-S-Node, and v has a code collision in the previous MCTS it sent, resend MCTS with a different Walsh code.
3) If the MRTS received is not from A-S-Node u, no action is taken.
4) If the MRTS received is a re-transmission MRTS, but not from A-S-Node u, resend the MCTS.

The second algorithm deals with an incoming MCTS. The following will be executed on a node v when it hear an MCTS packet from node x.

Receiving an MCTS

1) Extract the destination node y of the received MCTS,
2) if $v == y$, follow the protocol in the original MOCTS,
3) If $v != y$, remove x from its waiting list,
4) If the waiting list is empty, and the MCTS list is not empty, proceed to transmit the data packet.
5) if both the waiting list and MCTS list are empty, aboard transmission. This corresponds to the case where all it neighbor have received the multicast packet from other nodes.

It is not difficult to verify that deadlock will be always resolved with above protocol. In fact, the 4th rule in the MRTS protocol and the 3rd rule in the MCTS together will break the waiting condition if deadlock happens.

B. *Maximizing the Multicast Gain*

Another issue in extending MOCTS to multi-hop networks is the distributed control of packet propagation for better efficiency. We introduce new data structures so each node can dynamically track their local multicast status. Let graph $G(V, E)$ be the multi-hop wireless network in consideration. For each node $v \in V$, we denote its direct neighbor set by $N(v)$. At each node v, we maintain (1) $N(v)$, and (2) $N(u)$ for each $u \in N(v)$. This is, each node knows the network topology of its 2-hop neighborhood. The 2-hop neighbor set surrounding v is denoted by $N^2(v) = N(v) \bigcup_{i \in N(v)} N(i)$.

With these new data structures available at each node, the following locally executed algorithm is used to decide whether the received packet will be relayed, delayed or discarded (the protocol behavior is based on an arbitrary node v)

Distributed Local Gain Maximizing (DLGM)

- For each node $i \in N(v)$, define a multicast gain function $g(i)$ as the number of nodes in $N(i)$ that is not marked. Initially all nodes are not marked, thus $g(i) = |N(i)|$.
- When receiving an MRTS packet from a node u, mark all nodes in $N(u)$ as received.
- For each node $i \in (N(v) \bigcap N(u))/v$, update their $g(.)$ function accordingly. Particularly, $g(u)$ will become zero.
- If there exists a node $i \in (N(v) \bigcap N(u))/v$ such that $g(i) >= g(v)$, node v will mark a delay flag to itself.

- wait for the data transmission from node u to complete,
- When media become clear, if the delay flag is set, wait a random number of time slots, else send an MRTS to relay the multicast packet.

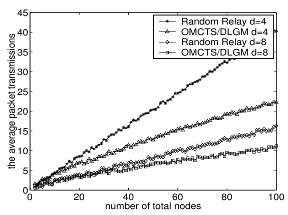
The above protocol will make sure that a consensus, about the node with the highest multicast gain, is formed in the $N(u)$ neighborhood of any S-node u. That node will transmit in the next round of multicast before any of its neighbors. Figure 2(b) shows the protocol execution for a small network. Multicast starts from node u, which has two direct neighbor v and i. Before node u send out MRTS, we have the following $g(.)$ value at v:$g_v(v) = 4, g_v(u) = 2, g_v(i) = 3, g_v(x) = 1, g_v(y) = 1$. When receiving MRTS from u, the $g(.)$ function at v is updated: $g_v(v) = 2, g_v(u) = 0, g_v(i) = 1, g_v(x) = 0, g_v(y) = 0$. Therefore node v will decide to relay the packet immediately. Similarly, at node i, we have $g_i(v) = 2, g_i(u) = 0, g_i(i) = 1, g_v(z) = 0$. Thus node i will decide to delay its relay request.

C. Simulation Results

The performance of the revised MOCTS protocol is evaluated by simulation. Our simulator randomly places mobile stations in a 1000*1000 square meters rectangle area. The number of nodes in the simulation increases from 2 to 100. For a given node number, we generate 50 different topologies to calculate the average measurement value. All topologies generated here are connected graph. Our main interest is to see how our protocol might affect the total number of packets relayed during multicast.

Figure 3.(a) shows the average number of packet transmissions during a multicast session, including all relayed packets. The result is classified for two comparing protocols: one with the proposed DLGM algorithm, and another assuming random selection for the relaying multicast nodes. We use two set of network topologies in simulation. The first set has an average node-degree of 4, and the second set has an average node-degree of 8.

We observed that the average packet number increases almost linearly as the number of nodes increases. The number of relays is significantly reduced when DLGM is employed. For the 100-node configuration, only 24 packets are required in average, indicating an 37% reduction compared to that of random selection. With $d = 8$, we observed similar performance trend. Also notice that, due to the higher node-degree, the number of packets required is significantly reduced. A close-to-20% reduction is observed for $d = 8$ case.



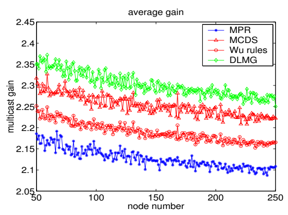
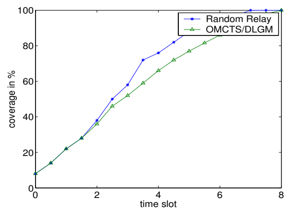


Fig. 3: (a) Number of relay vs network size, (b) multicast gain

Figure 4 shows the spread of multicast packet at different time instances for networks with 50 nodes and 100 nodes respectively. For networks with 50 nodes, multicast is completed within 8 packet slots in average. We compiled the packet propagation statistics for such cases, and obtain an time-coverage map for both cases. At any given point, the coverage of the random selection algorithm is larger than that of the DLGM. For random selection, 50% coverage is reached at 3.6 time slot. With DLGM, 50% coverage occurs at 4.0 time slot. The relative faster packet propagation is also observed for the 100-node configurations. Nevertheless, in term of overall multicast time, the difference of the two compared algorithm is not significant.



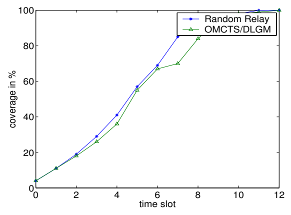


Fig. 4: Coverage vs Time (a) N=50, (b) N=100

V. Conclusion

We proposed a reliable MAC-layer multicast protocol, MOCTS, for wireless networks. Our approach is distinguished from other methods by the use of spread spectrum modulation in the CTS packets and allowing parallel transmission of CTS from all nodes. We discussed a distributed code selection mechanism to reduce code collision and demonstrate the protocol designs to deal with deadlock issue. Our simulation shows that MOCTS scheme significantly improve multicast efficiency in wireless network.

References

[1] ANSI/IEEE Standard 802.11, In 1999 Edition.

[2] W. Szapnkowski ”Analysis and Stability Consideration in a Reservation Multiaccess System”, *IEEE Trans. on Comm*, Vol. 31, No. 5, 1983 .

[3] V. Bharghavan, A. Demers, S. Shenker and L. Zhang, ”MACAW: A media access protocol for wireless LANs”, *In Proc. of ACM SIGCOMM (August 1994)*.

[4] H. Chhaya and S. Gupta, ”Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol”, *Wireless Networks*, vol.3, no.3, Aug 1997.

[5] B. Wang and S. K. S. Gupta, “S-REMiT: A Distributed Algorithm for Source-based Energy Efficient Multicasting in Wireless Ad Hoc Networks,” *In IEEE GLOBECOM'03*, pp.3519-3524.

[6] V. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs, *Int. J. Math.*, vol. 14, no. 1, pp. 1523, 1983.

[7] K. Chandy and J. Misra, “Distributed computation on graphs: Shortest path algorithms,” *Communications of the ACM*, vol. 25, no. 11, pp. 833 837, Nov. 1982.

[8] M. Doar and I. Leslie, How bad is naive multicast routing?, *in Proc. IEEE INFOCOM*, 1993, pp. 8289.

[9] K. Chen, N. Huang, and B. Li, ”CTMS: A Novel Constrained Tree Migration Scheme for Multicast Services in Generic Wireless Systems”, *IEEE J. ON Selected Areas in Comm*, vol.19, no.10, OCTOBER 2001, pp1998-2005.

[10] J. Kuri and S.K. Kasera, “Reliable Multicast in Multi access, Wireless LANs,” *In IEEE INFOCOM 99*, 1999.