4-2011

PeerCast: Improving link layer multicast through cooperative relaying

Jie XIONG
Singapore Management University, jxiong@smu.edu.sg

Romit Roy CHOUDHURY
Duke University

Citation

XIONG, Jie and CHOUDHURY, Romit Roy. PeerCast: Improving link layer multicast through cooperative relaying. (2011). *Proceedings IEEE INFOCOM 2011: Shanghai, China: 10-15 April 2011*. 2939-2947.
Available at: https://ink.library.smu.edu.sg/sis_research/2797

This paper was presented as part of the main technical program at IEEE INFOCOM 2011

PeerCast: Improving Link Layer Multicast through Cooperative Relaying

Jie Xiong
Department of Computer Science
University College London
j.xiong@cs.ucl.ac.uk

Romit Roy Choudhury
Department of Electrical and Computer Engineering
Duke University
romit.rc@duke.edu

***Abstract*—Wireless multicast applications, such as MobiTV, web telecast, and multimedia classrooms, are gaining rapid popularity. The broadcast nature of the wireless channel is amenable to such multicasts because a single packet transmission can be received by all clients. Unfortunately, the rate of this transmission is bottlenecked by data rate of the weakest client, degrading system performance. Attempts to increase the data rate results in lower reliability and higher unfairness. This paper presents *PeerCast*, a wireless multicast protocol that engages clients in cooperative relaying. The main idea is simple. Instead of multicasting at the bottleneck rate, the access point transmits at a high rate and suitably chooses a few stronger clients to relay the packet to the weaker ones. Multiple transmissions of the same packet, each at higher rate, can achieve better throughput than one transmission at the low, bottleneck rate. We propose a new simultaneous reply-back scheme for clients and detect the power level to estimate the AP's transmission strategy. PeerCast translates these ideas into a functional system using off-the-shelf hardware. Performance evaluation on a 9 node testbed demonstrates consistent throughput and reliability improvements over 802.11. Simulations in QualNet indicate similar trends in large-scale networks.**

I. Introduction

Content streaming applications are gaining popularity on the WiFi platform. Examples include WiFiTV [1], multimedia classrooms, live webcasts in offices, airports, and smart homes [2]. In all these applications, an access point (AP) is expected to disseminate the same content to a group of interested clients. The natural approach is to broadcast the packets on the wireless channel in a manner that all clients receive them. This forces the broadcast rate to be limited by the channel quality of the weakest client. Error recovery schemes must also be designed conservatively, imposing the need for more feedback and retransmissions. The multicast performance becomes inefficient [3]–[6]. The inefficiency is pronounced when only a few clients cause the bottleneck. Attempts to address this problem lead to sub-problems. In face of time-varying channel conditions, the weakest client can change over time. The bottleneck rate changes as well, making it necessary to continuously re-assess the appropriate broadcast rate. Even if the broadcast rate is efficiently assessed, the probabilistic nature of packet failures makes it hard to tell which clients experienced losses. Since per-client acknowledgment is an expensive proposition for multicast, meeting a desired level of reliability (across all multicast clients) is again a non-trivial problem. The above problems are not new – they have been identified and partly addressed in prior work [3], [4], [7]–[13]. While several existing ideas are indeed interesting, to the best of our knowledge, there still exists no WiFi-compatible system solution that accounts for throughput as well as reliability. Further, most studies are primarily theoretical, evaluated through simulations. We believe that validation of multicast performance is necessary on testbeds as well.

This paper designs, implements, and evaluates *PeerCast*, a multicast solution for 802.11-based WLANs. The main idea is simple (illustrated in Figure 1). Instead of broadcasting to all the clients at the bottleneck rate, the AP selects a higher rate to deliver a batch of packets to the *majority* of clients. Then, the AP chooses a suitable subset of these clients to relay the packets to the weaker ones. Since the channel quality between a strong and weak client can be significantly better than that between the AP and the weak client, the relayed transmissions can also occur at higher rates. Multiple transmissions at higher data rates can finish quicker than a single transmission at the low (bottleneck) rate. The multicast throughput and reliability can both improve.

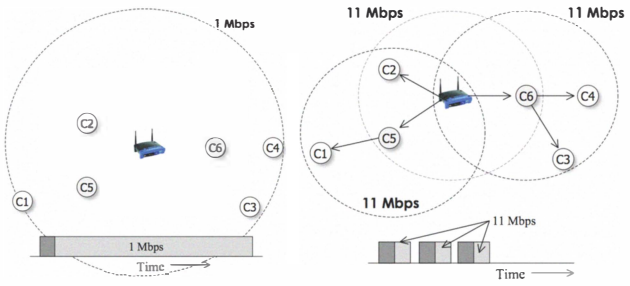

Fig. 1. Multiple high-rate packets (right) finish quicker than one low-rate packet (left). PeerCast exploits this opportunity.

Of course, a variety of challenges arise in the process of translating these ideas into a practical system. For instance, without explicit feedback from clients, how does an AP select the suitable rate to cover the "majority" of clients? How should relays be selected to ensure

that all weaker clients are serviced? How can a desired application-specified reliability be achieved? How can all the operations cope with time-varying channel fluctuations? The rest of the paper designs the overall protocol and describes its implementation using Click Modular Routers on Linux based platforms (laptops and Soekris boxes). Evaluation results on the testbed, and through Qualnet simulations, show consistent improvements, except in some pathological conditions. We summarize our main contributions as follows:

(1) Developing a cooperative architecture to achieve wireless multicast at the link layer. Strong clients relay packets to weak clients on behalf of the AP. The relaying responsibilities are balanced across multiple clients that experience good multicast service.

(2) A multicast rate selection scheme without serialized client feedback. Clients are required to respond simultaneously to periodic probes, sent at varying rates r_i. Although these responses collide at the AP, the received power can be correlated to the number of responders. This number indicates the reachable fraction of nodes at rate r_i.

(3) A relay selection algorithm to ensure a desired reliability across all the clients. We show that relay selection is equivalent to the Set Cover problem (hence NP-Complete). We propose practical heuristics.

(4) Implementation of PeerCast on Linux based laptops and Soekris boxes using modifications to the Click Router Modules. Testbed evaluation performed in realistic university environments yields throughput, reliability, and fairness gains. Qualnet simulations confirm scalability to larger systems.

II. Related work

There has been extensive research on network layer multicast for wireless ad hoc networks. Core graph theoretic ideas, including (connected) dominating sets [14], [15], spanning trees [14], Steiner trees [16], etc., have been applied towards optimizing a variety of performance metrics. In most of these network layer approaches, the wireless channel conditions have been abstracted with a cost metric. Some approaches have modeled channel variations only over long time scales, focusing on connectivity management, network stability, or multicast routing. This paper targets link layer multicast, a special case of the ad hoc network environment. However, we take advantage of the AP being in range of all the clients, hence, permitting a centralized multicast algorithm. In addition, we concentrate on prototyping the system on an off-the-shelf WiFi platform, thereby coping with the challenges from real channel conditions. Our system aims at improving both the throughput and reliability performances.

Several other researchers have also recognized the rich challenges inherent in link layer multicast [12], [17]. Authors in [4] have attempted variants of unicast schemes by requiring certain clients to acknowledge a packet on behalf of all nearby clients. The idea is that link qualities at spatially nearby clients may be correlated, hence, a client may send a proxy ACK on behalf of its neighbors. While this may be true outdoors, multipath and channel vagaries in indoor environments may violate these assumptions. To avoid the overhead of ACK storms [8], [9], alternate approaches have explored the possibility of choosing a conservative rate. Park *et. al* [10] propose a rate adaptation scheme that utilizes periodic (SNR) feedback from clients. The AP decides a transmission rate based on the lowest received SNR among the clients. While the protocol achieves good delivery ratio, its throughput is still bottlenecked by the weakest client.

To eliminate periodic probing of clients, [3] proposes a unary channel feedback (a type of tone), the length of which indicates the rate sustainable by a client. The AP receives all tones concurrently; the longest tone corresponds to the lowest-rate client in the network. Although useful, such a scheme may not be compatible with existing IEEE 802.11 standards. Saha et. al [18] and Kim et. al [11] have recently utilized OFDM sub-carriers to receive simultaneous feedback from clients. While this is an interesting approach, using few sub-carriers on an already weak channel can be highly susceptible to fading. Even if fading can be overcome, the transmission rate will still be bottlenecked by the weakest client, leading to low throughput. To bypass the "bottleneck rate" problem, our earlier work proposed a smart-antenna based solution in ICNP 2008 [19]. The idea is to transmit to strong clients using an omnidirectional antenna, and beamform to the weaker ones to improve reliability. Unfortunately, beamforming antennas may experience some problems in multipath environments (like directional client discovery, switching latency, etc.).

The ALLIANCES project by Athina et. al [20] proposed an optimal relay selection protocol in which the channel quality is obtained from the location information. This may not apply in real indoor environments as small distances between clients do not guarantee good channel conditions. MIP and BIP protocols proposed by Wieselthier et. al [21] focused on power consumption over throughput and reliability. Again, distance from location information is used to represent channel quality. This paper breaks away from smart antennas, and attempts to augment multicast performance with off-the-shelf 802.11 hardware and omnidirectional antennas.

III. PeerCast Overview

This section presents a high level overview of PeerCast (Figure 2), followed by metrics for performance evaluation. For ease of explanation, we assume that the AP is tasked with multicast transmissions only.

A. *Overview*

PeerCast consists of 3 main modules, namely Multicast Rate Selection (MRS), Relay Selection (RS), and Relay Transmissions (RT). In the steady state, the MRS module operates in batches of multicast packets. The first few packets of each batch act as probes to identify a suitable rate for the rest of the batch. The probes are transmitted at increasing data rates, and clients that receive the probe are expected to respond *in parallel* with a power-controlled ACK. Although the Parallel ACKs collide (Figure 2), the AP estimates the total received power and correlates it to the number of responders (detailed later). This allows the AP to understand the approximate fraction of clients reachable at different rates; the AP picks a rate, r_{AP}, that can cover more than a threshold fraction of clients. The remaining packets in the multicast batch are all transmitted using r_{AP}. Clients no longer respond with ACKs for these packets. At the end of a batch, clients *serially* reply with a Batch-ACK consisting of a bit vector that marks the missing packets at that client. Clients also piggyback the identifiers (and SINRs) of other clients, whose Batch-ACKs they have recently overheard. The AP consolidates all the ACK feedback into 2 tables – one table summarizes the link quality (SINRs) among client pairs, and the other reflects per-client relay needs for that batch. These tables are the inputs to the Relay Section and Relay Transmission modules.

For each packet, the Relay Selection module partitions the clients into two sets: those that have received that packet (S_{yes}), and those that have not (S_{no}). An optimal subset of S_{yes} should be selected such that together they can relay the packet to all members of S_{no}. Optimal relay selection is NP-complete, hence, PeerCast employs a greedy heuristic. The heuristic recruits relays that can cover a greater number of clients at a higher transmission rate. The AP shortlists these relays with rate information and broadcasts the entire schedule at a conservative rate. Since all clients are time-synchronized with the AP, the relays can conform tightly to the schedule and transmit one by one. As an optimization, the Relay Transmission module does not schedule relay of all packets; only an effective subset that can meet the reliability requirement at all clients. Once packet relaying is complete, the AP advances to the next batch of transmissions. The process repeats. PeerCast is a best effort service and cannot guarantee reliability. However, compared to pure AP-based approaches, PeerCast demonstrates consistent gains in throughput, reliability, and fairness.

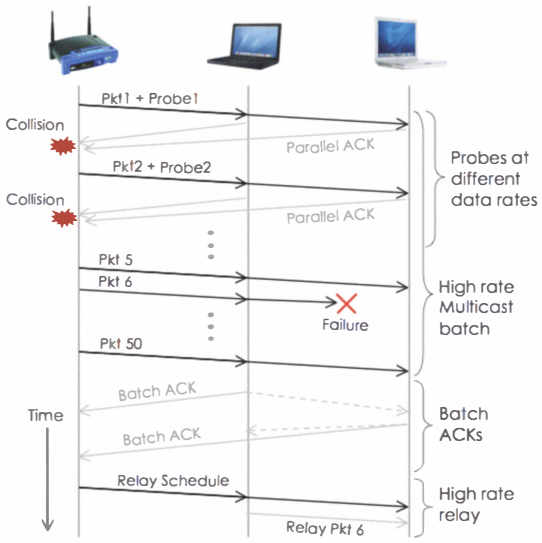

Fig. 2. Timeline for PeerCast

B. *Performance Metrics*

PeerCast is designed with the following 3 metrics in mind.

(1) Multicast Throughput is defined as the average number of packets received by the multicast clients per unit time. More formally, let us assume that an AP multicasts M packets over a T_m time window. Let N denote the number of multicast clients, and let m_i denote the number of packets received by i^{th} user. Multicast throughput, MT, is then defined by

$$MT = \frac{\sum_{i=1}^{N} m_i}{NT_m} \quad (1)$$

(2) Reliability. The reliability for client i is defined as

$$Rel_i = \frac{m_i}{M} \quad (2)$$

Minimum reliability of a network is the minimum Rel_i over all clients. We assume that the multicast application will expect each client to achieve a minimum reliability.

(3) Jain's Fairness Index, denoted by $f(.) \in [0, 1]$, is used to characterize the network's fairness. If x_i is an individual node's throughput ($\frac{m_i}{T_m}$), and N, the total number of clients, then Jain's fairness index is:

$$f(x_1, x_2, ..., x_N) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N * \sum_{i=1}^{N} {x_i}^2} \quad (3)$$

Fairness Index is used to compare how evenly the packets are received at clients. If all clients receive the same number of packets, fairness index is 1.

IV. Protocol Description

PeerCast exploits the intuition that intermediate clients between the AP and the weak clients can potentially act as relays. Figure 3(a) validates this intuition – the RSSI

of all client-AP pairs and client-client pairs were measured in our Engineering building classrooms across 20 topologies. Link SNR between clients is certainly greater than that between AP-to-clients. Towards leveraging this opportunity, two key questions need to be addressed. (i) At what rate should the AP multicast? This rate will influence which clients receive the packet successfully, and thereby, become candidate relays. (ii) Among the candidates, which subset must be designated as relays and at what rate should they transmit? We design PeerCast around these two questions.

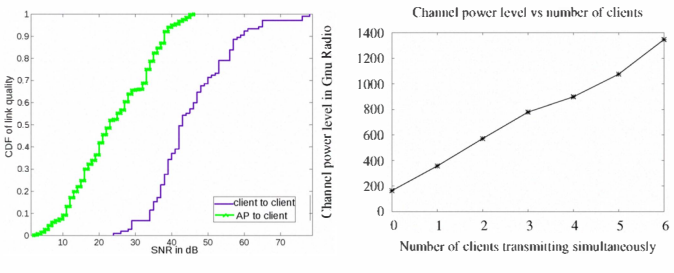

Fig. 3. (a) CDF of AP-to-Client SNR and Client-to-Client SNR. (b) USRP/GNU Radio based experiments demonstrating the impact of multiple colliding transmissions on the received power of AP.

(1) Multicast Rate Selection (MRS)

One design choice is to select a relatively high multicast transmission rate, estimate which clients are likely to get packets at this rate, and statically assign subsets of these clients as relays. Unfortunately, channel fluctuations may cause failures at a pre-chosen relay, affecting a larger group of clients that depend on it. Reducing the multicast rate may reduce this possibility, but the throughput will degrade as well. Moreover, some weak clients may receive a packet at this reduced rate, and a relayed packet may then be redundant. Choosing rates and delays without the awareness of packet losses is a risky proposition. Instead of such a *blind* approach, we introduce some degree of client feedback into the design, and amortize the overhead across a batch of packets.

Consider a batch of multicast packets of size B; PeerCast needs to estimate the suitable transmission rate for this batch. To this end, it uses the first few packets of this batch as probes. The first packet is transmitted at the lowest rate, and all clients that receive the packet respond with an ACK immediately. Carrier sensing is turned off to enable simultaneous transmission. The transmit power of the ACK, P^t_{ACK}, is also modified such that, irrespective of the position of the client, all ACKs arrive at the AP with roughly the same incident power, P^*. This can be achieved if client c_i records the received RSSI of the AP's packet, say P^r_i, and computes its path loss L_i from the AP. Since the AP transmits at a globally known power P^t_{AP}, we get $L_i = P^t_{AP} - P^r_i$. Assuming channel reciprocity [22], the ACK's transmit power at client c_i is then chosen based on the relation, $P^t_{ACKi} - L_i = P^*$. Reorganizing the terms, we have $P^t_{ACKi} = P^* + P^t_{AP} - P^r_i$

Although the simultaneous ACKs collide at the AP, it may be possible to correlate the total incident power on the AP's interface to the approximate fraction of clients that responded with the ACK. For the first probe at the base rate, r_0, the total received power, U_{r_0} indicates a fraction of 100%. The AP then transmits the next probe (the second packet in the batch) at a higher rate, r_1. The responding fraction is computed as a ratio of U_{r_1}/U_{r_0}. At higher rate probes, the ratio becomes a smaller value since fewer clients are able to overhear and respond. The highest rate that achieves a threshold ratio is selected as the multicast rate, r_{AP}.

To verify the feasibility of this idea, we used a USRP/GNU Radio platform [23]. Since the USRP exports the sampled signals to the GNU Radio, we were able to compute the total incident power at the receiver irrespective of collisions. As a starting point, we placed multiple transmitters (laptops and Soekris boxes running 802.11) around the USRP receiver. We manually regulated the transmit powers such that the individual received powers are approximately equal at the receiver. Then, we increased the number of simultaneous transmitters and recorded the corresponding received powers at the USRP. Figure 3(b) shows the variation, and offers reason to believe that the technique may be viable.

Recall that the above operation helps in identifying the suitable rate, r_{AP}, for reaching a threshold fraction of clients. Remaining packets in the multicast batch are transmitted at this rate, and are not acknowledged individually. In the last packet of the batch, the AP piggybacks an ordered schedule of Batch-ACK (BACK) transmissions. Clients reply with BACKs in this specified order, and remain in the promiscuous mode to overhear BACKs from nearby clients. The BACK includes (1) a bit vector indicating missing packets from the batch, and (2) the source address and SNR of all overheard BACKs. The AP consolidates the BACKs into an $N \times B$ table, where N is the number of multicast clients and B, the batch size. For any given packet j, the table shows which clients have received this packet. A second $N \times N$ table shows the pair-wise channel quality between clients. Relays can be scheduled based on these tables, hence, the tables are forwarded to the Relay Selection (RS) module.

(2) Relay Selection (RS)

Clients that receive packet j qualify for relaying j. Let us denote this set as S^j_{yes}. Obviously, not all these clients need to relay – a subset may be sufficient to cover all weak clients that have not received j, denoted S^j_{no}. However, each relay can support different transmission rates to different members of S^j_{no}, hence, the total relaying time varies across relay sets. The RS module's

task is to select the optimal relay set from S^j_{yes}, that minimizes the total relaying time for packet j. We show that Optimal Relay Selection is NP-complete. We present a heuristic within a $log(n)$ approximation of the optimal.

Theorem 1: Optimal Relay Selection is NP-complete

Proof: We prove this through a reduction from the Set Cover problem, known to be NP-Complete.
The Set Cover problem: Given a universe of n members, $U = \{1, 2, ...n\}$, and $S = \{s_1, s_2, ...s_m\}$ with $s_i \subseteq U$, find the fewest subset of S, denoted C, such that any member of U belongs to at least one member of C.

We present a reduction from an arbitrary instance of the set cover problem. Corresponding to the universe of n elements in Set Cover, we construct n clients that have not received the packet, i.e., $S_{no} = \{c_1, c_2, ...c_n\}$. For the m subsets $s_1, s_2, ...s_m$, we construct m corresponding client groups $g_1, g_2, ...g_m$, and m relays $R_1, R_2, ...R_m$, $R_i \in S_{yes}$. We connect a relay R_i to all members of g_i, indicating that R_i can deliver packets to the group g_i. The above operations require polynomial time. We assume all transmissions at a fixed rate, a special case of the Optimal Relay Selection problem. Thus, to efficiently solve the Set Cover problem, it is enough to solve the Optimal Relay Selection problem in polynomial time. This proves that Optimal Relay Selection is NP-complete.

We present a greedy heuristic to PeerCast's Optimal Relay Selection problem. The main idea is to favor relays that can cover a larger group of weak clients in a shorter duration of time. Thus, for any given relay, R_i, the number of clients it covers in S_{no} at each rate r_j is computed as N_{ij}. Denoting t_{ij} to be the time of packet transmission at rate r_j for relay R_i, we define R_i's contribution as $\frac{t_{ij}}{N_{ij}}$. These fractions are computed per-relay per-rate. The least fraction is chosen, and the corresponding $< relay, rate >$ tuple forms the first transmission. The client group covered by this relay are removed from S_{no}, and the metric re-executed on the updated set of uncovered clients. The new least fraction leads to the second choice of relay. This continues until all weak clients have been covered. A relay could be recruited multiple times (i.e., for different rates). In such a case, the relay performs only one transmission at the lowest of selected rates.

(3) Relay Transmissions (RT)

The above heuristic selects the relays and corresponding rates *for each packet* in the multicast batch. The AP composes a relaying schedule for each of the packets and broadcasts the entire schedule at a conservative rate (we discuss failure possibilities later). Clients overhear the schedule, and since they are tightly time-synchronized to the AP (802.11 TSF method is accurate to around 5μs), they transmit at the specified times. The schedule is serial, meaning no two relays transmit at the same time. Once the relaying schedule is complete, the AP advances to the next batch of transmissions.

Optimizing for Reliability: The above scheme relays a packet as long as there exists at least one client that has not received it. Various streaming applications may be able to tolerate some losses, hence, a client may not need to receive all packets. PeerCast exploits this opportunity to reduce relay transmissions. The key idea is to identify packets that have been lost at many clients, and to retransmit them to "plug as many holes". Relay transmissions need to continue until all clients have received at least a threshold number of packets, say 90%. Unfortunately, because the optimal choice of packets is also guided by their transmit durations (and hence, their relays and transmission rates), the problem is again NP-Complete. In the interest of space we omit the proof. We present a heuristic for practical implementation.

Heuristic: In our prior heuristic, we greedily selected the $< relay,\ rate >$ tuples for each packet. The target was to deliver this packet to all clients. We relax the target now by allowing a packet to be delivered only to a subset of clients, as long as each client receives a minimum number of packets at the end of a batch. Thus, instead of choosing the $< relay,\ rate >$ tuples per-packet, we consider all feasible $< packet,\ relay,\ rate >$ tuples. Now, for all feasible tuples $< P_i,\ R_j,\ r_k >$, let g_{ijk} denote the group of clients that require packet P_i and also can be served by relay R_j at rate r_k. Say $N_{ijk} = |g_{ijk}|$. The proposed heuristic computes the ratio $\psi_{ijk} = \frac{1}{r_k N_{ijk}}$ for all tuples. The minimum ψ_{ijk} value is isolated, say ψ^*_{ijk}. Thus, relay R^*_j now transmits packet P^*_j to all members of group g^*_{ijk}. PeerCast assumes that these packets will be successful, and thereby removes all clients whose reliability requirement will be satisfied by this transmission. The remaining clients still need more packets to achieve desired reliability. The heuristic updates the feasible tuples, as well as the g_{ijk} groups (note that removal of some satisfied clients can cause membership changes in g_{ijk}). The ψ_{ijk} values are computed again and the minimum value selected. This continues until all clients have satisfied the minimum reliability constraint. At this point, the AP forms the schedule based on the selected $< P^*, R^*, r^* >$ tuples, and broadcasts it at the base rate. The relays follow the schedule and perform timely relay transmissions. Algorithm 1 presents the pseudo code for the above heuristic.

Some Points of Discussion

(a) Do clients have an incentive to relay packets on the AP's behalf? Since relaying accelerates multicasting, a client may find it beneficial to participate so that its own unicast/multicast throughput increases [12]. A scheduling algorithm at the AP could even reward relays

Algorithm 1 Reliability-Based Packet Selection

Input: W: Set of clients below reliability threshold
$N = 0$
while $W \neq$ NULL **do**
  for all Packets P_i **do**
    for all Relays R_j **do**
      for all Transmission Rates r_k **do**
        if PacketReceived$(P_i, R_j) ==$ FALSE **then**
          Break
        g_{ijk} = ComputeRelayGroup(P_i, R_j, r_k)
        $N_{ijk} = |g_{ijk}|$
        $\psi_{ijk} = \frac{1}{r_k N_{ijk}}$
  $< \psi^*_{ijk}, g^*_{ijk} >$ = min (ψ_{ijk})
  $Schedule[N++]$ = UpdateRelaySchedule(i, j, k)
  for all client $c_i \in g^*_{ijk}$ **do**
    UpdateMissingPackets(c_i)
    if IsSatisfiedReliability$(c_i) ==$ TRUE **then**
      $W \leftarrow W - c_i$
Return $Schedule[.]$

by scheduling their (unicast) packets earlier. If energy is not a concern (as is the case with devices connected to power outlets), such a reward can be attractive. For devices running on batteries, the relaying energy may be more important than increased throughput. PeerCast tries to address this through load balancing. Clients that have relayed packets in the past accumulate credits. Based on the network designer's preferences, throughput and load balancing can be appropriately traded off. In our current implementation, we have chosen the highest-throughput relay, and broken ties using accumulated credits.

(b) Channel quality feedback from clients may be stale – does PeerCast address that problem? Recall that BACKs indicate missing packets as well as *channel conditions overheard recently* from nearby clients. Assuming N clients transmitting BACKs, the j^{th} client can piggyback SNRs overheard from the previous $(j-1)$ clients. For the remaining clients, $(j+1)$ to N, it piggybacks the SNRs overheard from BACKs sent after the previous multicast batch. This information can become stale, affecting relay selection. We propose three simple ideas to alleviate this problem. (1) The clients can transmit the BACKs in random orders specified by the AP, so that no fixed client subset is always uploading stale information. (2) The AP can deduce which part of the piggybacked information is stale, and can only use it if the relatively fresh information is inadequate for selecting relays. (3) In reality, unicast and multicast packets will be interspersed. Clients can overhear recent ACKs for unicast packets and piggyback these SNRs onto their BACKs. The current evaluation of PeerCast does not include these optimizations.

(c) How can PeerCast handle losses of BACKs and relay schedules? PeerCast is a best effort protocol – its performance may fail to uphold minimum reliability under adverse channel conditions. The damage is worst when the *Relay Schedule* from the AP fails to reach a relay R, hence, client groups depending on R suffer low delivery ratios. We argue that PeerCast can degrade gracefully. First, since relays have a strong channel quality to the AP (by design), and because the schedule is transmitted at base rate, the failure probability is proportionally lower. Second, because the AP load-balances across multiple relays, failure at any single (or few) relays may not be drastic. Packets overheard from other relays will still "plug some holes". Finally, by observing that a client did not relay packets in its specified slot, the AP can retransmit the schedule to that relay, and trigger its transmission. Traces from our evaluation show that PeerCast is not heavily affected by losses of BACKs and relay-schedules.

(d) How is batch size and the majority threshold chosen? We argue that AP transmission rate r_{AP} is not sensitive to batch size, B. Even though the entire batch is sent using a rate estimated at the beginning of the batch, independent channel fluctuations does not significantly affect the group's channel statistics (i.e., it is unlikely that all clients become stronger or weaker at the same time). PeerCast performance is also not sensitive to "majority" threshold because the AP receives client feedback, and can suitably retransmit to the failed clients. If many packets were lost due to an incorrect rate selection, the AP can adaptively decrease r_{AP} in the next batch.

V. Performance evaluation

Prototype Implementation

We have implemented PeerCast on a testbed consisting of laptops (running Linux kernel 2.6.27 and equipped with Atheros interfaces) and Soekris embedded PCs (running Metrix Pyramid Linux with Atheros Mini PCI interfaces). A laptop served as an AP, while Soekris boxes and additional laptops served as clients. PeerCast's functional logic is implemented through element extensions to the *Click Modular Router*. The AP is backlogged with broadcast packets for the entire duration of the experiment. Packets received by clients are sent to a network file system for offline processing. The AP is stationed in one classroom in our university building, and clients randomly scattered in the same and neighboring rooms as shown in Figure 4. Our testbed has interference from co-existing networks deployed by IT department. Since the university is densely populated with APs, clients are never too weak. We mimicked this in our experiment topologies (even though this is not favorable to PeerCast). The underlying MAC protocol is 802.11b. We evaluated scenarios with 2 to 8 multicast clients,

with multicast batch sizes of 100 packets. We empirically chose the majority threshold as 0.6. Each result is an average of 20 topologies with 95% confidence interval.

As a comparison point, we implemented *'802.11b with feedback'* with our simultaneous feedback approach via channel power detection. This scheme receives feedbacks quicker than SARM proposed in [10] and is equivalent to the best known proposed in [11] and [18]. The AP conservatively chooses the rate to approximately cover at least 90% of the clients. No batch feedbacks and replay transmissions are associated with '802.11b with feedback'.

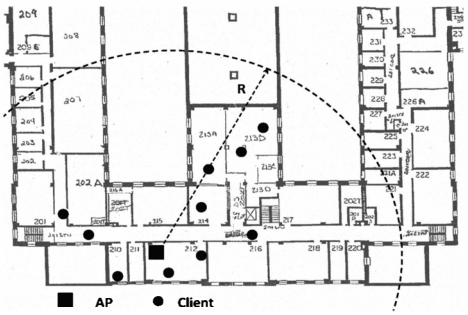

Fig. 4. Building floor plan with client locations marked.

Results: (1) *Throughput:* Figure 5(a) shows the throughput gain from PeerCast with increasing client density. The improvement is considerable for clients varying between 2 to 8, although the margin reduces at higher client size. This is because in these experiments the number of weak clients is very few, and hence, the overhead due to per-client BACKs dominates. Put differently, the throughput gain from covering the few weak clients does not compensate the higher BACK overhead. Since 802.11 operates at the bottleneck rate, its performance is poor whenever there is a weak client. When the weakest client is reasonably close to the AP, the performance improves. Not all environments may be as densely packed as university science/engineering buildings. We observed that in the university cafes and quads, weak clients were prevalent in greater numbers (especially due to shadow regions). Thus, we mimicked such client scatter in another set of throughput experiments. Figure 5(b) shows the outcome. In presence of more weaker clients, the absolute throughput decreased for both PeerCast and 802.11. However, the throughput gap increased because 802.11 was forced to pick low data rates, while PeerCast could benefit from high-rate relaying. In that sense, PeerCast is geared to cope well with more weak clients.

(2) *Reliability:* For the same cafe scenarios, Figure 5(c) shows the minimum delivery ratio (MDR) and the minimum average delivery ratios (MADR). The MDR is the minimum across all clients and topologies, while MADR is the average of the minimum in each topology. Surprisingly, the reliability difference is quite significant. Examination of the traces showed that even though 802.11 was transmitting at low data rates, packets were often lost due to channel fading and background interference. Further, SNR-based rate estimation is known to be inaccurate due to multipath signal environments [12]. PeerCast accounted for these losses and "plugged" them through high-rate relaying. This suggests that conservative rate selection may not be sufficient for multicasting – attending to individual clients' retransmission needs is crucial. Of course, even with relay transmissions, PeerCast achieved low reliability with 2-client topologies. This was because in some instances, the relay and the weak client were not in range of each other.

(3) *Fairness:* Table I shows PeerCast's fairness improvements over 802.11.

TABLE I
JAIN'S FAIRNESS INDEX COMPARISON

Number of clients	2	4	6	8
PeerCast	0.9919	0.9967	0.9942	0.9982
802.11b with feedback	0.9723	0.9743	0.9652	0.9769

(4) *Load Balancing:* PeerCast distributes the relaying responsibility among multiple clients. Table II shows the results from 3 random topologies selected from the 8-client experiments. Several clients shared the relaying load indicating reasonable load balancing properties. It's also shown that the total number of relay transmissions is much smaller than the packet batch size.

TABLE II
NUMBER OF RELAYS CARRIED OUT BY EACH CLIENT

Client ID	C1	C2	C3	C4	C5	C6	C7	C8
Topology 1	5	7	5	6	0	0	0	0
Topology 2	3	0	3	3	3	3	0	5
Topology 3	0	14	16	0	15	15	0	0

Qualnet Simulation

To understand PeerCast's behavior in larger networks, we simulated the protocol in QualNet [24]. The AP was placed in the center, while clients were scattered randomly around it. Experiments were performed with increasing number of clients, under various fading models. We used 20 topologies for each experiment. Table III summarizes the key parameters used in simulation.

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Physical layer	802.11b
Path Loss Model	Two-ray
Fading Model	Rayleigh, Rician (K=2)
Antenna Model	Omnidirectional
Number of clients	20,40,60
Minimum Reliability	90% unless mentioned otherwise
Space dimension	350m x 350m

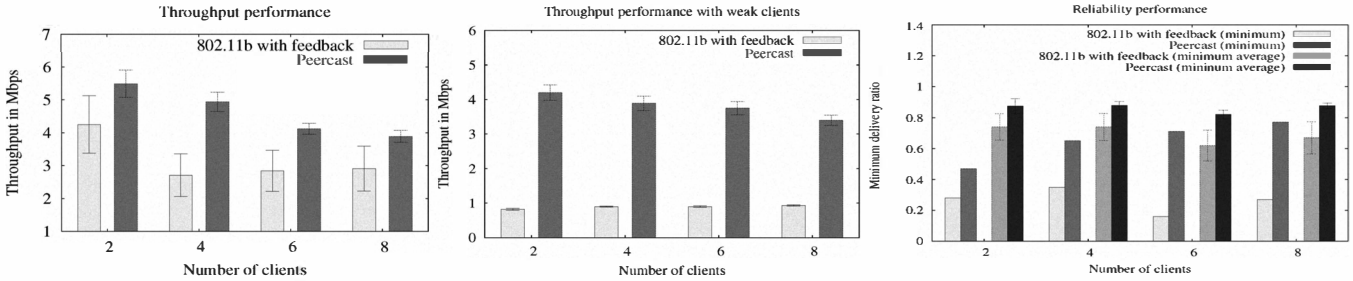

Fig. 5. (a) Average throughput performance. (b) Throughput with weak clients. (c) MDR and MADR against varying client density.

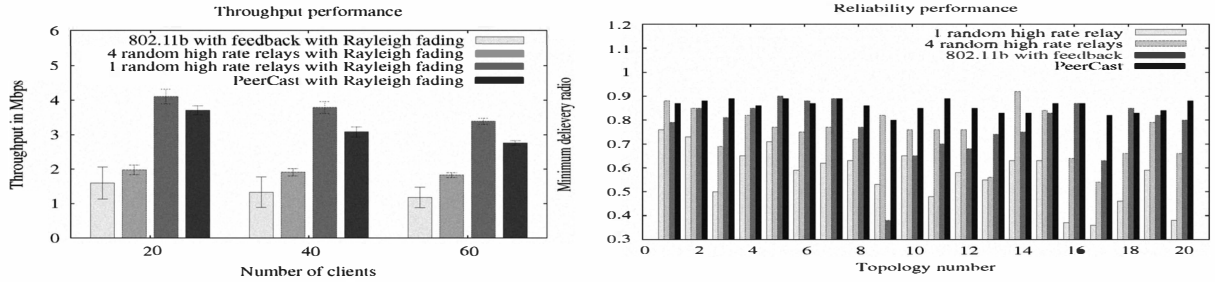

Fig. 6. (a) Average throughput with increasing multicast groups, under Rayleigh fading models. (b) Minimum delivery ratio at weakest client. Specified reliability threshold is 90%. The multicast group size is 20.

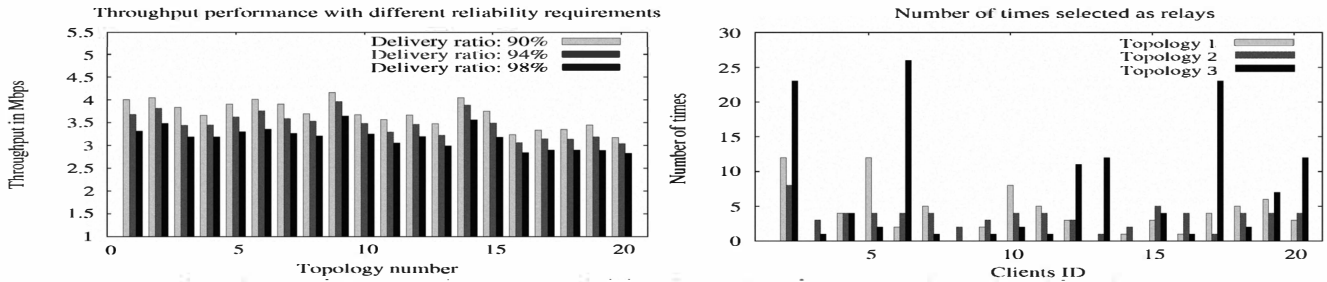

Fig. 7. (a) Throughput performance with increasing reliability. (b) Number of times each client selected for relay transmissions.

Results: (1) *Throughput:* Figure 6(a) shows PeerCasts performance improvement over 802.11b, because the latter chooses a conservative rate to ensure reliability. We also simulate a third scheme where K randomly selected relays are made to retransmit each packet. In this scheme, the AP transmits at the same rate as PeerCast while the relay transmits at the maximum allowable rate (11 Mbps). We show the performance for K=1 and 4. With K=1, the throughput is slightly higher than PeerCast, however, as we shall see soon, the reliability degrades severely. Also, results with varying K and data rates showed a strong trade-off between throughput and reliability. With K=4, the throughput reduces in exchange for higher reliability. However, both their reliabilities are lower than PeerCast. We observed that Rayleigh fading leads to worse performance than Ricean (not reported in the interest of space). This happens because Rayleigh fading is more severe, requiring more rounds of relaying with PeerCast. The impact is pronounced with 802.11b because under such strict fading environments, the bottleneck rate is limited by the weakest client, and hence, throughput degrades. Random relay selection already experiences a poor packet delivery ratio, so its relative degradation is slightly less with Rayleigh fading. Results from Rician fading show similar trends.

(2) *Reliability*: Figure 6(b) shows minimum delivery ratio across 20 topologies with a specified reliability threshold of 90%. Even the weakest client with PeerCast can achieve reliability reasonably close to the desired threshold. As admitted earlier, PeerCast is a best effort service and is unable to offer deterministic guarantees. Yet, we believe that none of the clients suffer significantly even under severe fading conditions. We observed that the median reliability with PeerCast was 96% across all clients and topologies. IEEE 802.11b observes reasonable reliability, except in some occasions when the degradation is severe. The 9^{th} topology is an example, and corresponds to the higher rate selected for 802.11b. The weakest client receives a 38% reliability, far below the target. Random relay selection performs poorly because the relays may be weak and may cover clients that have already received the packet correctly. As more relays are selected for each packet, the reliability of random relay selection approaches 802.11b with feedback but is still worse than PeerCast. This shows that although random relay selection can eliminate serial Batch-ACKs (hence, gain in throughput), the severe lack of reliability

makes it unacceptable. It's also noted that PeerCast copes well with the throughput-reliability tradeoff. Figure 7(a) shows throughput variation with increasing reliability thresholds. Observe that the degradation is quite graceful even the specified reliability is 98%.

(3) *Fairness:* Table IV shows PeerCast's fairness using Jain's fairness index. PeerCast consistently outperforms all the other protocols across all scenarios.

TABLE IV
JAIN'S FAIRNESS INDEX COMPARISON

Number of clients	20	40	60
802.11b with feedback	0.9942	0.9949	0.9946
1 random relay	0.9839	0.9861	0.9850
4 random relays	0.9947	0.9955	0.9960
PeerCast	0.9981	0.9982	0.9982

(4) *Relay load distribution:* Figure 7(b) presents the number of times each client was burdened with the responsibility of relaying. We present results from 3 arbitrarily picked topologies. The results are from one batch of transmissions (batch size is 100 packets). Observe that more than 80% of the clients were selected at least once even in one single batch. We believe this load balancing is a desirable property of a collaborative system.

VI. LIMITATIONS AND DISCUSSION

Misbehavior: Clients may misbehave to avoid relaying. A misbehaving client can report that it has not overheard any Batch-ACKs from nearby clients. Consequently, the AP will not schedule this client for relaying. Aggregating information from all BACKs, the AP can deduce that several clients have overheard a client c, but c has not overheard the others. If the suspicion is reinforced over multiple rounds, the client can be partially reprimanded.

Simultaneous relay transmissions: With interference map centralized at AP generated from the feedback of each client, several clients can retransmit simultaneously without interfering each other. This will increase spatial reuse and further improve throughput performance.

Mobility: We have not considered client mobility while evaluating our proposal. Even though PeerCast is reasonably robust to channel fading, the impact of mobility needs to be studied carefully. We intend to perform this study as a part of our future work.

Unicast: Real AP traffic will be composed of unicast and multicast sessions [7], [25]. Multiplexing between them intelligently may offer benefits, e.g., by overhearing unicast transmissions, PeerCast may be able to gather valuable information about the channel. On the other hand, introduction of unicast packets may also increase the duration of a batch, and affect the schedule of BACKs. We plan to address these issues in future.

VII. CONCLUSION

Wireless link layer multicast is bottlenecked by the channel quality of the weakest client. This paper addresses this problem through PeerCast, a multicast protocol that exploits client collaboration. The key intuition is that the channel quality between peer clients can be better than that between the AP and the weak client. PeerCast translates this simple idea into a functional system. Testbed evaluation and simulation results show promising throughput, fairness, and reliability results. Network coding, channel aware video coding, and other sophisticated schemes can be applied alongside PeerCast to further improve performance.

REFERENCES

[1] "MobiTV," Mobile television and radio service provider.
[2] "Smarthome," http://www.smarthome.com/index.aspx.
[3] Ai Chen, Gayathri Chandrasekaran, Dongwook Lee, and Prasun Sinha, "HIMAC: High throughput MAC layer multicasting in wireless networks," in *IEEE MASS*, 2006.
[4] J. Kuri and S.K. Kasera, "Reliable multicast in multi-access wireless lans," in *INFOCOM*, 1999.
[5] De-Nian Yang and Wanjiun Liao, "Protocol design for scalable and adaptive multicast for group communications," in *ICNP*, 2008.
[6] W. Tu, C.J. Sreenan, C. T. Chou, A. Misra, and S. Jha, "Resource-aware video multicasting via access gateways in wireless mesh networks," 2008.
[7] P. Chaporkar, A. Bhat, and S. Sarkar, "An adaptive strategy for maximizing throughput in MAC layer wireless multicast," in *MobiHoc*, 2004.
[8] K. Tang and M. Gerla, "Mac reliable broadcast in ad hoc networks," in *Proceedings of MILCOM*, 2001.
[9] M. T. Sun, L. Huang, A. Arora, and T. H. Lai, "Reliable MAC layer multicast in ieee 802.11 wireless networks," in *IEEE ICPP*, 2002.
[10] Y. Park, Y. Seok, N. Choi, Y. Choi, and J. Bonnin, "Rate-adaptive media multicasting over 802.11 wireless lans," in *CCNC*, 2006.
[11] BS. Kim, SW. Kim, and Ishmanov. F, "Reliable wireless multicasting with minimum overheads in OFDM-based wlans," in *ICC*, 2008.
[12] V. Bahl, R. Chandra, Patric P. C. Lee, V. Misra, J.Padhye, D. Rubenstein, and Y. Yu, "Opportunistic use of client repeaters to improve performance of wlans," in *CoNEXT*, 2008.
[13] K. Sinkar, A. Jagirdar, T. Korakis, S. Panwar, H. Liu, and S. Mathur, "Cooperative recovery in heterogeneous mobile networks," in *SECON*, 2008.
[14] XY. Li and I. Stojmenovic, "Broadcasting and topology control in wireless ad hoc networks," 2003.
[15] B. Mans and N. Shrestha, "Performance evaluation of approx algorithms for multipoint relay selection," in *MedHocNet*, 2004.
[16] W. Liang, "Approximate minimum-energy multicasting in wireless ad hoc networks," in *IEEE Transaction on mobile computing*, 2006.
[17] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu, "UCAN: A unified cellular and ad-hoc network architecture," in *Mobicom*, 2003.
[18] D. Saha, A. Dutta, D. Grunwald, and D. Sicker, "PAMAC: A PHY aided mac for wireless networks," in *INFOCOM*, 2009.
[19] S. Sen, J. Xiong, R. Ghosh, and R. R. Choudhury, "Link layer multicasting with smart antennas: No client left behind," in *ICNP*, 2008.
[20] H.Yang and Athina P.Petropulu, "Alliances with optimal relay selection," in *IEEE ICASSP*, 2007.
[21] J.E. Wieselthier, Gam D. Nguyen, and A. Ephremides, "Energy-aware broadcasting and multicasting in wireless ad hoc networks: A cross-layering approach," in *Mobile Networks and App*, 2004.
[22] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *MobiSys*, 2008.
[23] "GNU radio," http://www.gnu.org/software/gnuradio/.
[24] "QualNet," http://www.scalable-networks.com/.
[25] H. Won, H. Cai, D. Eun, K. Guo, A. Netrevali, I. Rhee, and K. Sabnani, "Multicast scheduling in celluar data networks," in *IEEE Infocom*, 2007.