Bounds for Blind Rate Adaptation*†

Seth Gilbert[1], Calvin Newport[2], and Tonghe Wang[3]

1 **Department of Computer Science, National University of Singapore, Singapore**
seth.gilbert@comp.nus.edu.sg

2 **Department of Computer Science, Georgetown University, Washington, USA**
cnewport@cs.georgetown.edu

3 **Department of Computer Science, Georgetown University, Washington, USA**
tw473@georgetown.edu

Abstract

A core challenge in wireless communication is choosing appropriate transmission rates for packets. This rate selection problem is well understood in the context of unicast communication from a sender to a known receiver that can reply with acknowledgments. The problem is more difficult, however, in the multicast scenario where a sender must communicate with a potentially large and changing group of receivers with varied link qualities. In such settings, it is inefficient to gather feedback, and achieving good performance for every receiver is complicated by the potential diversity of their link conditions. This paper tackles this problem from an algorithmic perspective: identifying near optimal strategies for selecting rates that guarantee every receiver achieves throughput within reasonable factors of the optimal capacity of its link to the sender. Our algorithms have the added benefit that they are *blind*: they assume the sender has no information about the network and receives no feedback on its transmissions. We then prove new lower bounds on the fundamental difficulty of achieving good performance in the presence of fast fading (rapid and frequent changes to link quality), and conclude by studying strategies for achieving good throughput over multiple hops. We argue that the implementation of our algorithms should be easy because of the feature of being blind (it is independent to the network structure and the quality of links, so it's robust to changes). Our theoretical framework yields many new open problems within this important general topic of distributed transmission rate selection.

1998 ACM Subject Classification C.2.1 Network Architecture and Design, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases bitrate, multicast, packet transmission, latency, competitive ratio, lower bound, fading

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2015.8

1 Introduction

Consider the following scenario: a base station must wirelessly deliver a large file or stream a video to an unknown group of receivers in a conference center. It could send the data individually to each receiver using unicast communication, but this approach does not scale and requires knowledge of the group members. The standard alternative is for the sender

* Due to the matter of space, we defer details of the proof for some theorems or lemmas to the full version of this paper [8].

† The research in this paper has been supported by Singapore NUS FRC T1-251RES1404 and NSF grant number CCF-1320279.

to *multicast* messages to receivers, i.e., use one-to-many communication where the sender transmits packets addressed to the whole group.

A challenge faced by wireless multicast is that different receivers might have different quality links with the sender. Some receivers, for example, might have high quality links with the sender that can support high transmission rates, while others might have low quality links that can support only slow rates. *What rate(s) should the sender use?* The current answer – i.e., the solution implemented into 802.11 multicast [17] – is to transmit the multicast packets at the slowest rate. This solution is simple and reliable, but from a performance perspective it might be unacceptable for receivers capable of supporting much faster communication. Not surprisingly, practitioners consider the identification of better multicast rate selection strategies as an important open problem [10, 4, 16, 23, 3, 19, 6, 5, 22, 24].

In this paper, we tackle this open problem from an algorithmic perspective. We first formalize this rate selection problem in an abstract model of multi-rate wireless transmission. We then describe and analyze new rate selection strategies that guarantee *every* receiver in our above scenario achieves throughput within a reasonable factor of the optimal capacity of its sender link. We establish lower bounds indicating that these algorithms are near optimal. To the best of our knowledge, these are the first known rate selection strategies to offer competitive performance for every receiver in a wireless multicast scenario (see the related work below for more details). An added and perhaps surprising benefit of our algorithms is that they are also *blind*: the sender requires no knowledge of the network and receives no feedback on the fate of its transmission. This should make the implementation based on the algorithms robust to change.

1.1 Results

To formalize this multicast problem, assume a sender s and multiple receivers. For each packet to send, the sender must specify a transmission rate from a set of available rates. We normalize the transmission times associated with these rates such that the fastest rate delivers a packet in 1 time step, while the slowest delivers a packet in L time steps, for some system parameter $L > 1$. Notice, the range of possible rates with which a packet can be sent are fixed and provided by the radio hardware in most systems. Each receiver u is connected to s by a link labeled with a *fastest acceptable rate*. The sender succeeds in delivering a packet p to u iff it sends the packet at a rate no faster than the fastest acceptable rate for this link. Because we model a wireless network, the sender's packets are transmitted by broadcast. Therefore, each packet p, sent with some rate r, is received by *every* receiver with a link labeled by a fastest acceptable rate at least r. We measure the performance of the sender's rate selection strategy at receiver u by comparing the average latency between packets successfully delivered to u to the latency achieved if the sender had deployed the optimal strategy (for u) of transmitting every packet at the fastest acceptable rate for u's link. The sender's task is complicated by the fact that it must remain competitive for every receiver concurrently, even though their link qualities might vary widely and it receives no information on these qualities.

We begin by describing and analyzing a pair of blind rate selection algorithms (one randomized and one deterministic) that both guarantee that for each receiver v in the network, the throughput at v is within a $O(\log L)$-factor of optimal. Notice that the simple strategy of transmitting at the slowest available rate can be exponentially worse (i.e., achieve only an L factor of optimal). The core idea leveraged by both algorithms is to have the sender copy each packet into multiple queues, each associated with a different representative rate. The sender then dequeues and transmits messages from these queues at a frequency proportional to the corresponding rates (e.g., the fast rate queues are sampled more frequently than the slow rate queues). This means that each packet might end up being sent multiple times, but we show our proportional sampling prevents this from degrading throughput too much over time. We then establish this $O(\log L)$ competitive ratio near optimal by proving that no (randomized) blind rate selection algorithm can guarantee throughput better than a $\Omega(\frac{\log L}{\log\log L})$-factor of optimal (with constant probability).

We next turn our attention to the setting where the fastest acceptable rates on the links change rapidly and unpredictably, as might be described in a *fast fading* scenario. We prove that for every *deterministic* blind rate selection algorithm there exists a sequence of fades (i.e., link quality changes) that reduce its average latency guarantee to a trivial $\Omega(L)$-factor of optimal (which can always be matched within a constant factor by simply sending packets at the slowest rate). We then describe a type of fade for which no *randomized* algorithm can perform guarantee better than a $\Omega(\sqrt{L})$-factor of optimal (still exponentially worse than our results for the static setting). Interestingly, this latter bound holds even for non-blind unicast communication (i.e., where there is a single receiver and the sender learns the fate of each transmission), proving the difficulties caused by fading are not unique to blind algorithms.

To conclude, we consider multihop networks. We describe and analyze a generalization of the deterministic protocol that guarantees every receiver achieves throughput within a $O(\log L)$-factor of the optimal achievable through the best *single* multihop route (i.e., path) from the source. Notice, however, in a multihop network, the optimal throughput possible using *multiple* paths in the network might be better than the optimal throughput on a single path (e.g., perhaps multiple packets are routed concurrently on disjoint routes). We prove that no blind algorithm can guarantee a non-trivial approximation of this notion of optimal while also maintaining a fixed bound on its packet ordering.

1.2 Related Work

The multicast problem is well-studied in the wired network setting; c.f., [10]. In the wireless setting, the technology is still evolving. As mentioned, the default strategy implemented in 802.11 is to simply broadcast multicast packets at a slow but reliable bitrate. The research literature contains many proposals for adding more advanced functionality to wireless multicast, with a focus on detecting multicast packet loss. These strategies, however, depend on the sender interacting with at least some members of the multicast group (i.e., use feedback). Chandra et al. [4], for example, send the multicast data in unicast packets to a single member of the multicast group (leveraging link layer acknowledgments to detect packet loss), while the other members listen for these packets in promiscuous mode. Miroll et al. [16] refine this approach to select the group member with the worst channel as the unicast receiver to ensure more losses are detected. Sun et al. [23] organize nodes into clusters, and has the sender poll the leaders of each cluster to determine the fate of packets. (These are just a few examples among many: see [3] for a detailed survey.) The main goal of the above examples is to detect packet loss so the sender can schedule retransmissions. Most of these strategies, however, also implement some basic link adaptation. For example, when transmitting unicast packets to a leader, some of these strategies allow the default unicast rate adaptation strategy to adjust the rate used (e.g., [19, 6, 5, 22, 24]). Other wireless multicast strategies propose measuring loss rates for all receivers and using this information to choose the best *single* rate to use (as mentioned, a strategy that does not scale). This paper, by contrast, focuses on the problem of transmitting packets at multiple rates so as to ensure *every* receiver achieves a throughput close to its individual notion of optimal. It achieves this goal without the overhead and scaling issues of requiring feedback from group members.

It is also important to note that because we achieve competitive rates for every receiver, loss detection is less important with our scheme. For example, if we define the *fastest acceptable bitrate* for each receiver to be the fastest rate at which the forward error correcting coding parameters used for the packets are effective, then our blind rate selection algorithms will guarantee that every receiver has a sufficiently low loss rate to enable sufficient packet recovery.

Finally, wireless rate adaptation is well-studied in the context of unicast communication from a sender to a single known receiver capable of sending acknowledgments. Some strategies adapt the rate using frame loss information [12, 14, 11, 21, 15] and others attempt to directly measure the channel quality [18, 13, 25, 2]. Another approach is the use of rateless modulation schemes like Spinal codes [20] or Strider codes [9], in which the data transmitted in a fixed manner but the receiver is able to decode it at a rate close to the Shannon capacity for its channel. All these unicast adaptation strategies, however, depend on feedback from the single receiver to the sender. They cannot therefore be directly applied to the multicast scenario where such feedback is no longer efficient. Our blind protocols, by contrast, *can* be deployed in the unicast scenario. This might be desirable in low power scenarios where their simplicity provides an advantage, or scenarios such as satellite broadcast where feedback is prohibitively expensive (i.e., due to the much higher cost of uplink versus downlink transmission).

2 Model

We model a collection of wireless devices broadcasting in a synchronous radio network with variable link quality and transmission rates (typically called *bitrates* in the wireless literature as well as in the remainder of this paper). The network topology is represented as a connected directed graph $D = (V, E)$, where the nodes in V correspond to wireless devices and the edges in E represent links between nodes (e.g., an edge $(x, y) \in E$ means that x has a link to y with a quality above some minimum threshold). We use directed graphs for generality and to capture the well-known observation that link quality is not necessarily symmetric. The main topology we consider is a star with the node in the center playing the role of the sender with directed edges pointing toward the receivers. We call this configuration a *single hop* network. Later in the paper, we also examine the performance in general *multihop* networks of varying topologies connected with respect to the source (i.e., there is a path from the source to all nodes).

To capture link quality we assume that in each round, each *link* (i.e., edge in D) is assigned a minimum latency (i.e., fastest acceptable bitrate), which is specified by the weight function $C(r, e)$ (which we sometimes call a *channel*) for round r and edge e. We say that links are *static* if the weight of links does not change (i.e., $C(r, e) = C(r', e)$, for all r, r' and e); otherwise, we say that the links are *fading.* These weights capture the fastest transmission speed that can be supported by the current link quality. In this paper, we typically specify these weights in terms of the latency (i.e., rounds per packet), rather than in terms of the bitrate (which describes the inverse). Therefore, smaller weights represent higher quality links. We assume that all latencies are integers in $[L]$ (where we define $[k] = \{1, 2, ..., k\}$), with 1 and L rounds being the fastest and slowest transmission latencies, respectively. We also assume that L is a power of 2. To simplify our strategies, we will restrict the possible latencies considered for packet transmission to the set $\mathcal{L}^* = \{2, 4, 8, \ldots, L\}$. (Notice, there is a latency in $\mathcal{L}^*$ within a factor of 2 of each available latency.)

Nodes communicate with their neighbors in D using local broadcast. When a node s in D decides to broadcast a message in round r, it selects a latency $\ell \in \mathcal{L}^*$ (this is equivalent to selecting bitrate ℓ^{-1}). This transmission requires ℓ rounds to complete, and the transmitter must wait until the transmission completes before it can begin another transmission. Each neighbor w of s receives the message if the attempted broadcast latency remains as least as large as the minimal acceptable latency for the link throughout the full transmission. Formally, the transmission succeeds at w if and only if: $\forall i \in [0 \ldots \ell - 1] \ : \ C(r+i,(s,w)) \leq \ell$. If this condition does not hold, then node w does not receive the message. We assume no feedback mechanism (e.g., link layer ACKs) for the sender to learn the fate of its transmission, and assume nodes have no information about the network size or link weights.

A subtlety of our model motivation is that our abstract notion of "receiving" a message does not correspond to the concrete notion of a packet being successfully delivered. For u to "receive" a message from s in our model simply means that s sent this packet at a rate that was acceptable for its current link to u. What it means for a rate to be acceptable are details we abstract away: we aim only for every packet to be sent at acceptable rates for each receiver, for whatever definition of acceptable is relevant to a given a scenario.[1]

3 Problem

In this paper, we study the problem of a single distinguished *source* node s attempting to transmit an infinite stream of packets to the other nodes (called *receivers*) in the network. We measure the performance of an algorithm in a given execution by comparing the average packet receive latency at each receiver u with an offline optimal algorithm that services only u. We consider a restricted type of solution called a *blind rate selection algorithm*. An algorithm of this type running on the source node in the network is provided access to a packet queue called the *source queue*. To simplify definitions we assume the queue is infinite and the packets unique. The source node can only dequeue and transmit packets from the source queue (i.e., it cannot send arbitrary packets). In the multihop setting, where non-source nodes can forward packets, we assume each arriving packet is queued in a FIFO queue (ignoring duplicates), and then restrict nodes to dequeueing and transmitting packets from their local queue. In the single hop setting, receivers are passive (i.e., they cannot send packets.)

Recall, as described in the previous section, to "receive" a message in our abstract model simply means it was sent at an acceptable rate for the relevant link. How this translates to low level packet loss behavior is abstracted away. In this paper, we study the performance of correct blind rate selection algorithms defined with respect to the average time between packet arrivals at a given destination. We call this metric *average latency* (which is a mild abuse of terminology as "latency" often refers to end-to-end delivery, not inter-packet delay at a receiver). More precisely: Fix an execution of a blind rate selection algorithm in a network $D = (V, E)$ with weight function C. Fix some receiver $v \in V$ (i.e., non-source node) and integer duration $T \geq 1$. Let N_v^T be the number of unique packets received by v in the first T rounds of the execution. We define the *average latency* of v through the first T rounds of this execution to be T/N_v^T. By contrast, let OPT_v^T be the *optimal* average latency v could have achieved in these T rounds given an offline optimal schedule for transmissions and rate selections (when clear, we will use simply OPT). We now pull together these pieces to obtain the main performance definition:

[1] For example, an acceptable rate for a link in a given scenario might be defined as a rate for which the packet loss rate is sufficiently low. To send a packet at an acceptable rate in this example, therefore, does not guarantee that it was delivered, but merely that it was given a reasonable chance of delivery.

▶ **Definition 1.** Fix some function $f : \mathbb{N}^* \to \mathbb{R}$. We say a deterministic (randomized) blind rate selection algorithm $\mathcal{A}$ is $f(L)$-competitive with respect to a family $\mathcal{D}$ of networks and static/fading channels, if the algorithm is correct and there exists some integer duration $T_0 \geq 1$ such that for every $D \in \mathcal{D}$ and static/fading link weight function C, for every receiver v in D, and for every duration $T \geq T_0$: the (expected) average latency of v through T rounds in an execution of $\mathcal{A}$ in D with C, is no more than $f(L) \cdot OPT_v^T$.

4 Static Links

In this section, we study blind rate selection algorithms in the context of single hop networks with *static* links (i.e., the link qualities do not change). We begin by describing a simple randomized blind rate adaption algorithm that is $O(\log L)$-competitive in single hop networks. We then describe a more complex deterministic algorithm that matches this same $O(\log L)$-competitive ratio. There are two motivations for deterministic solutions. The first is that rate selection algorithms are often implemented at low layers of the network stack where efficient access to randomness is difficult. Second, the multihop algorithm studied later in the paper uses the deterministic algorithm as a key building block (analyzing the randomize strategy over multiple hops is difficult). We conclude this section by proving our algorithms near optimal with an $\Omega(\log L/\log\log L)$ lower bound on competitive ratio for blind rate adaption algorithms.

4.1 The RandSelect Algorithm

We model the infinite packet queue at the source s with the notation $Q_0 = p_1p_2p_3\ldots$. Recall, as defined in Section 2, $\mathcal{L}^* = \{2, 4, \ldots, L\}$.

A simple random strategy for s would be to dequeue packets from Q_0 one by one, sending each at a latency chosen uniformly from $\mathcal{L}^*$. An issue with this approach is reliability: if s chooses some latency ℓ for a given packet p_i, and there is some receiver u such that $C(s, u) > \ell$, then u will fail to receive p_i. Another issue with this strategy is that the slowest latency, L, will be chosen approximately once every $\log L$ rounds – requiring L full rounds for a single transmission every time it is chosen. This will yield non-competitive performance for receivers connected to the source with low latency links. Our proposed algorithm improves this simple scheme with two modifications to circumvent these two issues. First, the source maintains $\log L$ copies of its source queue, associating one copy with each latency in $\mathcal{L}^*$. We logically organize these queue copies into a *packet table* with one row for each latency. In more detail, each row $j \in [\log L]$ is associated with latency 2^j, and contains its own copy of the source queue, denoted Q_j, as well as a $nextpacket_j$ field which indicates the packet currently at the head Q_j. The second modification is to replace the uniform distribution over rates in $\mathcal{L}^*$ with the following distribution $\pi(x)$ over the latency indices $\{1, 2, ..., \log L\}$[2]:

$$\pi(x) : Pr\{j = x\} \begin{cases} 2^{-x} & x \in [1, \log L - 1] \\ 2/L & x = \log L \end{cases} .$$

Combining these modifications, our algorithm, which we call RandSelect, works as follows: At the beginning of the execution, the sender initializes its packet table by setting Q_j to Q_0 for each $j \in [\log L]$. It then proceeds by repeating the following steps: draw a *latency index* x from π, transmit $nextpacket_x$ at latency 2^x, and then update $nextpacket_x$

[2] Notice, it is easy to show that distribution π is normalized, i.e. $\sum_{x=1}^{\log L} Pr\{j = x\} = 1$.

```
RandSelect (for sender $s$)
Initialization:
  for $j \leftarrow 1$ to $\log L$
    $Q_j \leftarrow Q_0$
    $nextpacket_j \leftarrow p_1$

Transmission:
  do
    Select $j$ according to distribution $\pi$
    Send $nextpacket_j$ with latency $\ell \leftarrow 2^j$
    $pop(Q_j)$
    $nextpacket_j \leftarrow peek(Q_j)$
  while TRUE
```

Figure 1 The RandSelect Algorithm.

by dequeueing the packet at the head of Q_x. (See Figure 1 for the algorithm pseudocode.) Notice that this strategy overcomes both the issues described above for the simple random strategy: no packet is ever lost, as every packet is eventually sent at the slowest latency, and the algorithm now samples the slow latencies less frequently than the fast latencies, preventing them from dominating the link bandwidths.

The following theorem establishes that the actual competitive ratio guaranteed by this strategy is bounded by $O(\log L)$. We defer the proof of this theorem to the full version [8].

▶ **Theorem 2.** *The* RandSelect *blind rate selection algorithm is* $O(\log L)$*-competitive with respect to single hop networks and static links.*

A straightforward practical optimization would be to remove a packet from fast latency queues in the case that it is sent first by a slower latency. This optimization does not effect the asymptotic analysis.

4.2 The BCSSelect Algorithm

We now describe a deterministic blind rate selection algorithm we call BCSSelect (see Figure 2), which we will prove to have the same competitive ratio as RandSelect in single hop networks. The only difference between these two algorithm is how indices are selected. Our main strategy for derandomizing RandSelect is to leverage a useful object from number theory called the *binary carry seqeuence* (BCS) [1]. This BCS is defined such that its k^{th} term is the lowest position of a 1 bit in the binary representation of k. To use this sequence for our algorithms, we use the deterministic *schedule* function defined as follows: for $k \in \mathbb{N}^* : schedule(k) = \max\{\alpha \in \mathbb{N} : 2^{\alpha-1} | k\}$. The output of *schedule*, for example, produces the sequence: 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2, 1, 3, 1... Before proceeding to our algorithm description (which uses the *schedule* output to sample queues), we first state some useful facts about *schedule* (first identified in [7] and reworded here):

▶ **Lemma 3.** *Each value* $j \in \mathbb{N}$ *is selected every* 2^j *iterations.*

▶ **Lemma 4.** *If* $s = schedule(k)$*, then:*
(i) $\forall t < s$, $\exists r \in [k - 2^{s-2}, k)$ *such that* $t = schedule(r)$*;*
(ii) $\forall t < s$, $\exists r \in (k, k + 2^{s-2}]$ *such that* $t = schedule(r)$*.*

```
BCSSelect (for source $s$)
Initialization:
  $k \leftarrow 1$
  for $j \leftarrow 1$ to $\log L$
    $Q_j \leftarrow Q_0$
    $nextpacket_j \leftarrow p_1$

Transmission:
  do
    $j \leftarrow schedule(k)$
    Send $(nextpacket_j, k)$ with latency $\ell = 2^j$
    $pop(Q_j)$
    $nextpacket_j \leftarrow peek(Q_j)$
    $k \leftarrow$ Update$(k)$
  while TRUE

Update$(k)$
  if $k = L/2$ then
    return 1
  else
    return $k + 1$
```

Figure 2 The BCSSelect Algorithm.

The BCSSelect algorithm (described in Figure 2), applies the *schedule* function as a subroutine to sample latencies from a (bounded) binary carry sequence – which ensures, as with the random distribution from before, that all latencies are sampled, but small latencies are sampled more often than their slower counterparts. In more detail, the algorithm uses *schedule* to sample the BCS until the first time latency L is sampled, at which point it restarts the sequence. As a result, the sequence of latencies sampled by *schedule* can be seen as repeating the same bounded BCS block with $L/2$ terms.[3] Here the source sends current BCS index k along with the packet, as this will prove useful in the later multihop version of the algorithm we study later.

We now argue that BCSSelect is $O(\log L)$-competitive, the same competitive ratio as RandSelect.

▶ **Theorem 5.** *The* BCSSelect *blind rate selection algorithm is $O(\log L)$-competitive with respect to single hop networks and static links.*

Proof. Fix receiver v with $C(s,v) = c$, where $\log c \in J = [1, \log L]$. Fix $T_0 = L \log L$ as well and suppose all the execution runs for $T \geq T_0$ rounds. By the same token, we have $OPT = c$ because the optimal solution runs with the sender v knowing $C(s,v) = c$ *a priori* and applying latency c throughout the execution.

Now the task is to find the average latency of BCSSelect. Consider the number of rounds required by the update of $nextpacket_{\log c}$, upper bounding the average latency we are looking for. One BCS block has $L/2$ terms because the greatest BCS index $j = \log L$ is

[3] Notice that the probability of selecting latency ℓ given by distribution π from RandSelect is equal to the proportion of latency ℓ in one such bounded BCS block.

selected for the first time when $k = L/2$. According to Lemma 3 and 4, the total time cost to generate one block is exactly the time needed to get another $j = \log L$ after the previous $j = \log L$, given by: $\sum_{j=1}^{\log L-1}(L/2) \cdot (1/2^j) \cdot 2^j + 2^{\log L} = L(\log L + 1)/2$. Lemma 3 tells us that latency c (BCS index $\log c$) appears every $2^{\log c} = c$ iterations of *schedule*. So $\log c$ appears $L/2c$ times in one block, and the average time needed for $nextpacket_{\log c}$ to update will be $O\left(\frac{L(\log L+1)/2}{L/2c}\right) = O(c \log L)$.

In conclusion, the competitive ratio of BCSSelect during one block is $O(\log L)$. Because of the fact that the infinite latency sequence is actually the repeating of the same block, the competitive ratio during the whole execution process is still $O(\log L)$. ◀

4.3 Lower Bound

We now prove our algorithms near optimal by showing that every blind rate selection algorithm is at best $\Omega(\log L/\log\log L)$-competitive. Our argument is combinatorial in nature – it demonstrates that no sequence of rate selections can be sufficiently competitive for every receiver in a particular lower bound network – and therefore it applies to randomized solutions as well as deterministic.

▶ **Theorem 6.** *For a randomized blind rate selection algorithm $\mathcal{A}$, if $\mathcal{A}$ is $f(L)$-competitive with respect to any single hop network and static links, then $f(L) \in \Omega(\log L/\log\log L)$.*

Assume for contradiction that we have some algorithm $\mathcal{A}$ that is $o(\log L/\log\log L)$-competitive for all networks and weight functions. For our lower bound, we define the following single hop *lower bound network*: let D be a directed graph that consists of $n = \log L/\log\log L$ receivers[4] denoted $r_1, r_2, ..., r_n$. Next, we define the weight function C such that for each $i \in [n]$, $C(s, r_i) = \log^i L$. That is, the weights in this network are: $W = \{\log L, \log^2 L, \log^3 L, ..., \log^{\log L/\log\log L} L = L\}$. We proceed with a series of proof step that build toward the conclusion that executing $\mathcal{A}$ in the lower bound network cannot yield the assumed small latency at every receiver. Our first step is to transform the algorithm $\mathcal{A}$ into an algorithm $\mathcal{B}$ that uses only the latencies in W. The following lemma states that there exists transformation of this type that do not affect performance. The proof is deferred to the full version of this paper [8].

▶ **Lemma 7.** *Let $\mathcal{A}$ be a rate adaptation algorithm that is $f(L)$-competitive in the lower bound network. There exists an algorithm $\mathcal{B}$ that only selects latency in W but is still $f(L)$-competitive in the lower bound network D.*

Fix some blind rate selection algorithm $\mathcal{B}$ that only uses latencies in W. We now prove that a constant fraction of the packets received by a given receiver must be at a “good” rate (i.e., the link weight) for that receiver.

▶ **Lemma 8.** *Let $\mathcal{B}$ be a rate adaptation algorithm that only selects latencies in W and is $f(L)$-competitive in the lower bound network D. Fix some duration $T_0 = L$. Let k_i be the number of messages received by receiver r_i in a T round execution of $\mathcal{B}$, where $T \geq T_0$. It follows that at least $\lceil k_i/2 \rceil$ of these messages are sent at latency $\ell_i = \log^i L$.*

Proof. Assume for contradiction that half or more of these k_i message were sent at a latency greater than ℓ_i. The minimum latency for a packet sent to receiver r_i is ℓ_i. Therefore, if we

[4] For notational simplicity we assume $\log L$ and $\log L/\log\log L$ are positive, integral values.

assume that at least half the messages arriving at r_i are sent at a latency longer than ℓ_i, the next best case average latency for r_i would be at least: $((k/2)\ell_{i+1} + (k/2)\ell_i)/k$.

In the above, we assume the minimum number of packets (in this case, $k/2$) were sent at a slower latency, and we made this the next slowest latency after ℓ_i (i.e., $\ell_{i+1} = \log^{i+1} L$). The rate above also assumes the source was only servicing r_i and sent packets continuously with no gaps throughout the T rounds. It is, in other words, a quite optimistic bound. We now simplify:

$$\frac{(k/2)\ell_{i+1} + (k/2)\ell_i}{k} = \frac{(k/2)\ell_i \log L + (k/2)\ell_i}{k} = \frac{(1/2)\cdot k \cdot \ell_i(\log L + 1)}{k} > \frac{1}{2}\cdot \ell_i \cdot \log L\,.$$

The optimal average latency for r_i is clearly ℓ_i. Therefore, $\mathcal{B}$ is *at best* $(\log L/2)$-competitive. We assumed earlier, however, that $\mathcal{B}$ is $f(L)$-competitive for a function that is no larger than $c \cdot \log L/\log\log L < \log L/2$ (for sufficiently small constant $c > 0$). This contradicts our assumption that at least half of r_i's packets were sent slowly. ◀

We have just established that to achieve a reasonable competitive ratio for a given receiver in our network, at least half of the packets sent to the the receiver must use a rate well-suited to the receiver's link. We next establish formally another important observation for our overall lower bound: to achieve a good rate in an execution of length T, the source must successfully deliver many packets.

▶ **Lemma 9.** *Let $\mathcal{B}$ be an algorithm that is $f(L)$-competitive in the lower bound network D. Fix some receiver r_i and duration $T \geq T_0 = L$. It follows that r_i receives at least $T/(\ell_i \cdot f(L))$ packets during these T slots, where $\ell_i = \log^i L$.*

Proof. Fix some $f(L)$-competitive algorithm $\mathcal{B}$, as well as some r_i and $T \geq T_0 = L$. Consider a T-round execution of $\mathcal{B}$ in the lower bound network. Let $\ell_1^*, \ell_2^*, ..., \ell_j^*$ be the latency for each receive event at r_i in our T-round interval. Let α be the average latency at r_i in this interval. Notice, by definition: $\alpha = (1/j)\cdot\sum_{h=1}^{j}\ell_h^*$. Also note, however, that by definition: $\sum_{h=1}^{j}\ell_h^* = T$. It follows that $\alpha = T/j$, and therefore $j = T/\alpha$. By assumption, $\mathcal{B}$ is $f(L)$-competitive. This means that when considering r_i in particular, its average latency is no greater than $\ell_i \cdot f(L)$ and $j \geq T/(\ell_i \cdot f(L))$, as required. ◀

Proof (of Theorem 6). Let $\mathcal{A}$ be the rate adaptation algorithm that we assumed to be $f(L)$-competitive for some $f(L) < c \cdot \log L/\log\log L$. Let $\mathcal{B}$ be the constrained rate adaptation algorithm provided by Lemma 7. By the guarantees of this lemma, $\mathcal{B}$ is $f(L)$-competitive in the lower bound network D. We will now show that this leads to a contradiction.

In particular, we will show that for any sufficiently large duration $T \geq T_0 = L$, $\mathcal{B}$ is at best $\Omega(\log L)$ competitive which is $\omega(f(L))$ which contradicts our assumption that it is $f(L)$-competitive.

To get this result, let k_i be number of packets that the source sends at latency $\ell_i \in W$ in a T-round execution of $\mathcal{B}$ in the lower bound network D. By Lemma 9, we know that receiver r_i receives at least $T/(\ell_i \cdot f(L))$ packets. By Lemma 8, we know at least half these packets are sent at latency ℓ_i. It follows that $k_i \geq T/(2\cdot\ell_i\cdot f(L))$. We can now evaluate how many rounds are required for the source to send the needed number of packets at each rate, and derive the following answer:

$$\sum_{i=1}^{n} k_i \cdot \ell_i = \sum_{i=1}^{\log L/\log\log L} \frac{T}{2\cdot \ell_i f(L)}\cdot \ell_i = \frac{\log L}{\log\log L}\cdot\frac{T}{2f(L)}\,.$$

By assumption, however, $f(L) < c \cdot \log L / \log\log L$. If we set c to be sufficiently small (e.g., $c < 1/2$), it simplifies to something strictly larger than T. There are only T rounds available, however, to complete all broadcasts. This yields a contradiction to our assumption about the bound on $f(L)$, and therefore $f(L)$ is in $\Omega(\log L / \log\log L)$ ◀

5 Fast Fading Links

In this section, we consider the setting where link weights can change from round to round. It is straightforward to identify a weight function C that causes our static BCSSelect algorithm to perform poorly in the face of some dynamism (i.e., achieve only an $O(L)$ competitive ratio with respect to optimal).

Here we generalize this observation by proving this weakness is true of *all* deterministic blind rate selection algorithms. We prove that there is a link weight definition for which *no* randomized algorithm can guarantee better than a $\sqrt{L}$-competitive ratio (which is still exponentially worse than the $\log L$ ratio we achieve for static links). Perhaps surprisingly, this latter bound even holds for the powerful model of unicast communication with a single receiver and packet feedback. These bounds indicate that it is quixotic to seek an algorithm that can always adapt competitively to fast fades.

Lower Bound for Deterministic Algorithms. A deterministic blind rate selection algorithm can be described as a fixed sequence of latency choices. Here we prove that for any such sequence we can define a link weight function for a two-node network (the simplest possible network for rate selection) that guarantees a poor competitive ratio.

▶ **Theorem 10.** *For a deterministic blind rate selection algorithm $\mathcal{A}$, if $\mathcal{A}$ is $f(L)$-competitive with respect to two-node networks and fading links, then it follows that $f(L) \in \Omega(L)$.*

The proof of this theorem is deferred to the full version [8]. At a high-level, this argument defines a weight function that keeps the link weight large when the algorithm attempts fast transmissions, and reduces the weight to something small when the algorithm attempts slow transmissions.

A Lower Bound for Randomized Algorithms. Here we show that randomization cannot guarantee much advantage over determinism given fading links. The following theorem holds even for non-blind rate selection algorithms in which the sender learns the fate of each packet.

▶ **Theorem 11.** *For a randomized blind rate selection algorithm $\mathcal{A}$, if $\mathcal{A}$ is $f(L)$-competitive with respect to two-node networks and fading links, then it follows that $f(L) \in \Omega(\sqrt{L})$. This bound holds even with packet delivery acknowledgements.*

The proof of Theorem 11 depends on the following lemma.

▶ **Lemma 12.** *For a blind rate selection algorithm $\mathcal{A}$, if $\mathcal{A}$ is $f(L)$-competitive with respect to two-node networks and fading links, then for every function $g : \mathbb{N}^* \to \mathbb{R}$ such that $g(L) < L/2$,*

$$f(L) \in \Omega\left(\min\left\{(L/g(L)), g(L)\right\}\right).$$

This bound holds even with packet delivery acknowledgements.

To come close to the optimal solution, a randomized algorithm must effectively guess correctly the beginning of the fast interval in each block. Receiving feedback after the fact regarding whether it guessed correctly does not help its future guesses. We formalize this argument in the full version of this paper [8]. Armed with this lemma, we can prove our theorem.

Proof (of Theorem 11). According to Lemma 12, for any appropriately chosen $g(L)$, the lower bound for any randomized blind rate adaption algorithm $\mathcal{A}$ is $\Omega\left(\min\left\{\left(L/g(L)\right), g(L)\right\}\right)$. In particular, the strongest lower bound $\Omega(\sqrt{L})$ is achieved when $g(L) = \sqrt{L}$, and the duration $T \geq T_0 = 2g(L) = 2\sqrt{L}$. ◀

6 Multihop Networks

In this section, we turn our attention to routing information through multihop networks. In particular, consider a multihop network in which the source s may not have a direct link to some designated receiver t. In this setting, s will have to forward messages through intermediate nodes to get to t, with each such node needing to make its own rate selection decisions.

We consider two natural methods to measure optimality with respect to t. The first method is to consider any single path from s to t in the network, and compare t's throughput when the algorithm is run on this path to the throughput obtained by the optimal algorithm for the path. We call this *single path* optimality. We describe a deterministic algorithm called MultiBCSSelect that generalizes the single-hop BCSSelect algorithm to obtain throughput within a $O(\log L)$-factor of the single path optimal solution. The second method is to compare the throughput at t when the algorithm is executed in the entire network as compared to the optimal algorithm executed in the entire network. Notice, once you make use of the entire network, it might be possible to obtain more performance (e.g, by routing multiple packets to the destination concurrently over disjoint paths). Our MultiBCSSelect algorithm cannot guarantee this *multiple path* optimality. We prove, however, that in some sense *no* blind algorithm can. In more detail, we prove that it is impossible for a blind rate selection algorithm to guarantee a non-trivial approximation of the multiple path optimal solution *and* to be δ-order preserving (i.e., sequence numbers of received packets do not get more than δ values out of order), for any fixed δ. We note that this latter property is necessary for many network applications, and our MultiBCSSelect algorithm is 0-order preserving.

6.1 The MultiBCSSelect Algorithm

Here we describe a blind rate adaptation algorithm for multihop packet transmission based on BCSSelect. In particular, we have the source node s run BCSSelect, as in the single hop setting. The non-source nodes, by contrast, run the MultiBCSSelect algorithm described in Figure 3. This algorithm initializes each Q_j at an intermediate node as an empty queue. As an intermediate node receives a packet for the first time, it pushes it onto the back of each of its queues. This algorithm has nodes sample queues as in the single hop algorithm. In the case that it samples an empty queue, the node will simply transmit an "empty packet" (technically, we can interpret this as not sending any packet). We synchronize the indexes nodes use to sample the BCS by propagating the current index in the transmitted packets.

It is straightforward to show that MultiBCSSelect is correct in a multihop setting. More interesting is analyzing its performance. Because we consider single path optimality, we restrict our attention to a subgraph P consisting of a path from s to some fixed destination t, i.e., $V_P = \{v_0 = s, v_1, v_2, \ldots, v_n, v_{n+1} = t\}, E_P = \{(v_i, v_{i+1}) : i = 0, 1, \ldots, n\}$. We show that

```
MultiBCSSelect(for intermediate nodes)
Initialization:
  for $j \leftarrow 1$ to $\log L$
    $Q_j$ is initialized by an empty packet queue

On receiving $(p_0, k_0)$:
  $k \leftarrow$ Update$(k_0)$
  for $j \leftarrow 1$ to $\log L$
    $Q_j \leftarrow push(Q_j, p_0)$
    $nextpacket_j \leftarrow peek(Q_j)$

Transmission:
  do
    $j \leftarrow schedule(k)$
    Send $(nextpacket_j, k)$ with latency $\ell = 2^j$
    $pop(Q_j)$
    $nextpacket_j \leftarrow peek(Q_j)$
    $k \leftarrow$ Update$(k)$
  while TRUE
```

Figure 3 Algorithm of MultiBCSSelect.

the performance of MultiBCSSelect is competitive with that achieved by the best path P, i.e., the best multihop route to t. (The best multihop route is the path which gives the highest throughput.)

▶ **Theorem 13.** *The* MultiBCSSelect *blind rate selection algorithm is $O(\log L)$-competitive with respect to the single path optimal solution.*

Before completing the proof of Theorem 13, we will bound the performance of the optimal algorithm on P (the proof of this lemma is in [8]):

▶ **Lemma 14.** *Fix a multihop route consisting of the path P with $n+2$ nodes $v_0, \ldots, v_{n+1}$, where $s = v_0$ and $t = v_{n+1}$. Suppose $c^* = \max_{0 \le i \le n}\{C(v_i, v_{i+1})\}$. For all multihop routes with n intermediate nodes, if the links are static, the optimal average latency $OPT_t = \Omega(c^*)$.*

Proof (of Theorem 13). Now we need to capture the average latency of MultiBCSSelect. Consider some link $(v_\beta, v_{\beta+1})$ with β being the greatest index such that $C(v_\beta, v_{\beta+1}) = \max_{0 \le \alpha \le n}\{C(v_\alpha, v_{\alpha+1})\} = c^*$. In other words, $(v_\beta, v_{\beta+1})$ is the last bottleneck, or the last slowest link. Since MultiBCSSelect applies synchronous binary carry sequence, a packet will be in transmission successfully before any packet arrives, indicating that v_β is the last place where packets get queued.

When running algorithm MultiBCSSelect, according to the analysis from Theorem 5, the link with weight c sends a new packet within $c \log L$ rounds during one block of binary carry sequence. We will see that the worst case for transmission through path P derives from the case where $C(v_i, v_{i+1}) = c^*$ for all $i > \beta$. Then the average time for v_n to update $nextpacket_{\log c^*}$ is no more than $O(c^* \log L)$, and the corresponding competitive ratio on this path is therefore $O(\log L)$.

Since this is true for all paths P, including the best such path, we have proved our claim. ◀

6.2 Lower Bound for Multiple Path Optimality

Here we show it is impossible to be multiple path optimal and still maintain a natural packet ordering property. This latter property is captured by the following two definitions.

▶ **Definition 15.** We define the sequence number of a packet p, denoted by $seq(p)$, to be the order of packet p in the source's packet queue at the beginning of the execution (where the packet at the head of the queue occupies position 1, and so on). Fix some non-source node t. Similarly, we define the transmission number of p with respect to t, denoted by $tn(t,p)$, to be the order in which t first received p, ignoring duplicate receives of packets.

▶ **Definition 16.** Fix some integer $\delta \geq 0$. A rate adaptation algorithm is δ-order preserving if for every packet p in the source queue, and every non-source node t, we have $|seq(p) - tn(t,p)| \leq \delta$.

This notion of δ-order preserving is important for many applications in which received packets need to be reordered for processing. If I need, for example, k out of an original group of t packets to recover some coded information, and some of these packets can get arbitrarily out of order, I might have to wait an arbitrarily long time to complete the decoding.

We continue by noting that our multihop algorithm is perfectly order preserving:

▶ **Theorem 17.** MultiBCSSelect *is* 0*-order preserving.*

Proof. The source node copies its source queue into $\log L$ transmission queues, each one associated with a different latency. Packets are removed and transmitted from each queue in FIFO order. A straightforward consequence is that for any two packets p and p', such that $seq(p) < seq(p')$, the source cannot send p' for the first time before it sends p (consider the queue from which the source samples p' for the first time: in order to reach p' in that queue, the source must have previously sampled and transmitted p).

It then follows that all neighbors of the source will receive messages for the first time in the same order as they appear in the source queue. They will subsequently add them to their transmission queues the same way. We can, therefore, apply the same argument as before to show this order is preserved to their neighbors, and so on, until we have considered every node in the network. It follows that for any non-source node t and any two packets p and p, if $seq(p) < seq(p')$, then $tn(t,p) < tn(t,p')$. ◀

With these definitions established, we can now state our main theorem, which claims that a blind algorithm cannot be both non-trivially competitive with respect to the multiple path optimal results, and be order preserving for some fixed δ.

▶ **Theorem 18.** *There exists a constant $c' > 1$, such that for every integer $\delta \geq 0$ and competitive factor $c < L/c'$, there does not exist a blind rate adaptation algorithm that is c-competitive with respect to the multiple path optimal path solution* and *δ-order preserving.*

To prove this lower bound, we will make use of a graph $D_r = (V, E)$, where V consists of a source, s, a destination, t, and L relay nodes, $r_1, r_2, \ldots, r_L$. Let $E = \{(s, r_i) : 1 \leq i \leq L\} \cup \{(r_i, t) : 1 \leq i \leq L\}$. Fix $C(s, r_i) = 1$ and $C(r_i, t) = L$ for all $i = 1, 2, \ldots, L$.

Fix some rate adaptation algorithm $\mathcal{A}$ that is c-competitive for some constant $c > 0$. To prove Theorem 18, we will show that there exists a network such that for all $x \geq 1$, there exists a packet p such that $|seq(p) - tn(t,p)| = \Omega(x \cdot L)$. For any fixed δ, therefore, we can find a sufficiently large x for which the algorithm is not δ-order preserving.

Let us study the constant competitive algorithm $\mathcal{A}$. Since all links coming out of the source have the same capacity, relay nodes will receive the same packet in the same transmission round. In order to achieve good competitiveness, relay node may not send packets in FIFO order. Otherwise, there will be a great amount of repetition of packets at the destination t, since relay nodes receives the same packet in each round. Actually we can claim without proof that the constant competitive solution for one single transmission round is to have L different relay nodes send $\Theta(L)$ different packets in the queue.

We will prove that after x transmission rounds (xL communication rounds), the maximum sequence number of the packets that have already been sent will be $\Omega(x) \cdot \Theta(L) = \Omega(xL)$ for all $x \geq 1$. We will forget about the first $L+1$ communication rounds and regard the start of the $(L+2)$th rounds as the start of $x = 1$.

▶ **Lemma 19.** *When executing $\mathcal{A}$ on graph D_r, there exists some relay node r, such that for every integer $x \geq 1$, there exists an integer $x' \geq x$, such that $seq(p_{x'}^{(r)}) \geq (1/c)x'L$, where $p_{x'}^{(r)}$ is the x'th unique packet that r sends.*

We will put the proof of this lemma in the full version [8]. A simple counting argument yields the next lemma:

▶ **Lemma 20.** *When executing $\mathcal{A}$ in any network, for every node u, after u transmits x unique packets, the smallest sequence number among packets u has not yet sent is no more than $x+1$.*

We can now pull together the pieces to prove our main theorem.

Proof (of Theorem 18). The key observation used by this proof is that a relay node r_i cannot distinguish an execution in D_r from an execution in the graph $D_r^{(i)}$ which consists only of: the source s, with a directed edge to r_i, with a directed edge to t. Now consider an execution of $\mathcal{A}$ in L copies of $D_r^{(i)}$, one for each r_i. At the same time, run this algorithm with the same random bits in D_r. We will look at the behavior of $\mathcal{A}$ in D_r to point to an i for which $D_r^{(i)}$ behaves poorly.

In more detail, we apply Lemma 19, which identifies some r_i in D_r for which the lemma statement holds. Let x be the value identified by the statement for r_i. Let $x' = \max\{x, (\delta+1)/(\frac{L}{c}-1)\}$, where δ is the order-preserving bound from the theorem statement. Consider $p_{x'}^{(i)}$, the x'th packet sent by r_i. By the statement, $seq(p) \geq (1/c)x'L$. By Lemma 20, however, there is some sequence number $q \leq x'+1$, such that r_i has not yet sent the packet with that number.

Now consider r_i in $D_r^{(i)}$. It too will send a packet with sequence number at least $(1/c)x'L$ before it sends a packet with number $x'+1$. Because r_i must eventually send every packet in this graph (as it is the only relay node), when it does eventually get to the packet with sequence number $x'+1$, it will be out of order. In particular, the gap between this packet's number and the x'th packet's number is at least: $x'(L/c) - q \geq x'(L/c) - (x'+1) > \delta$. (Notice, it is here that we require that c is sufficiently small compared to L.) We have just identified, however, an execution of $\mathcal{A}$ in a graph that is non-order preserving. A contradiction. ◀

7 Conclusion

In this paper, we study *blind* multicast rate selection algorithms which do not require feedback from receivers, and yet allows each receiver to achieve throughput within a reasonable constant factor of its link's optimal capacity. We prove these algorithms near optimal and then explore the fundamental impossibilities of coping with fast fading, even for non-blind algorithms. We conclude by showing how our deterministic strategy can be effectively adapted to multihop scenarios.

We argue that our algorithms are easy to implement in practice, because they are blind, or they does not require any information on the network structure or the quality of links. Our formal model of multi-rate transmission, however, is a standalone contribution as it helps bring together the practical concerns of rate selection with the theoretical toolkit of algorithmic analysis. This framework yields many interesting open questions. For example, this paper only scratches the surface of understanding optimal rate selection in general network topologies. Even identifying an efficient centralized solution for approximating an optimal selection sequence is an open problem. Another natural approach would be to consider unicast rate selection where the sender receives feedback on each transmitted packet's fate. The existing solutions for this problem rely on heuristics. It would be useful to study this problem from an algorithmic perspective, seeking formal bounds.

References

1 K. Atanassov. On the 37th and the 38th Smarandache problems. *Notes on Number Theory and Discrete Mathematics*, pages 83–85, 1999.

2 Saâd Biaz and Shaoen Wu. Loss differentiated rate adaptation in wireless networks. In *IEEE WCNC 2008*, 2008.

3 Saâd Biaz and Shaoen Wu. Rate adaptation algorithms for IEEE 802.11 networks: A survey and comparison. In *Proceedings of IEEE Symposium on Computers and Communications*, 2008.

4 R. Chandra, S. Karanth, T. Moscibroda, V. Navda, J. Padhye, R. Ramjee, and L. Ravindranath. Dircast: A practical and efficient Wi-Fi multicast system. In *Proceedings of the 17th IEEE International Conference on Network Protocols*, 2009.

5 N. Choi, Y. Seok, T. Kwon, and Y. Choi. Leader-based multicast service in IEEE 802.11v networks. In *Proceedings of the 7th IEEE Consumer Communications and Networking Conference*, 2010.

6 S. Choi, N. Choi, Y. Seok, and T. Kwon. Leader-based rate adaptive multicasting for wireless LANs. In *Proceedings of IEEE Global Telecommunications Conference*, 2007.

7 Alejandro Cornejo and Calvin Newport. Prioritized gossip in vehicular networks. In *DIALM-POMC'10*, 2010.

8 Seth Gilbert, Calvin Newport, and Tonghe Wang. Bounds for blind rate adaptation. Available at: `http://people.cs.georgetown.edu/~cnewport/publications.html`.

9 Aditya Gudipati and Sachin Katti. Stanford networked systems group. `http://snsg.stanford.edu/projects/strider/`.

10 Lawrence Harte. *Introduction to Data Multicasting.* Althos Publishing, 2008.

11 G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM MOBICOM'01*, 2001.

12 A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 1997.

13 J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. In *IEEE INFOCOM'06*, 2006.

14 M. Lacage, M. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: A practical approach. In *MSWiM04*, 2004.

15 Z. Li, A. Das, A. K. Gupta, and S. Nandi. Full ato rate MAC protocol for wireless ad hoc networks. In *IEEE Proceedings on Communication*, 2005.

16 J. Miroll and Z. Li. Aggregate block-ACK definition. *Tech. Rep. IEEE*, 2010.

17 Sai Shankar N., Debashis Dash, Hassan El Madi, and Guru Gopalakrishnan. WiGig and IEEE 802.11ad for Multi-Gigabyte-Per-Second WPAN and WLAN. *arXiv:1211.7356*, 2012.

18 Qixiang Pang, Victor Leung, and Soung C. Liew. A rate adaptation algorithm for IEEE 802.11 WLANs based on MAC-layer loss differentiation. In *Proceedings of IEEE Broadband Wireless networking symposium*, 2005.

19 Y. Park, Y. Seok, N. Choi, Y. Choi, and J.-M. Bonnin. Rate-adaptive multimedia multicasting over IEEE 802.11 wireless LANs. In *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, 2006.

20 Janathan Perry, Hari Baladrishnan, and Devavrat Shah. Rateless spinal codes. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011.

21 B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *MOBICOM02*, 2002.

22 Y. Seok and T. Turletti. Practical rate-adaptive multicast schemes for multimedia over IEEE 802.11 WLANs, 2006. https://hal.inria.fr/inria-00104699.

23 M.-T. Sun, L. Huang, A. Arora, and T.-H. Lai. Reliable MAC layer multicast in IEEE 802.11 wireless networks. In *Proceedings of International Conference on Parallel Processing*, 2002.

24 J. Villalón, P. Cuenca, L. Orozoco-Barbosa, Y. Seok, and T. Turletti. Cross-layer architecture for adaptive video multicast streaming over multirate wireless LANs. *IEEE J. Sel. Areas Commun.*, 25(4):699–711, 2007.

25 S. Wong, H. Yang, S. Lu, and B. Bharghavan. Robust rate adaption for 802.11 wireless networks. In *MOBICOM'06*, 2006.