Mitigating Unfairness due to Physical Layer Capture in Practical 802.11 Mesh Networks

Wei Wang, Ben Leong, and Wei Tsang Ooi
Department of Computer Science, National University of Singapore

Abstract—In this paper, we describe *FairMesh*, which is the first attempt at mitigating the unfairness *arising from physical layer capture (PLC)* in 802.11 mesh networks. In the presence of PLC, which is surprisingly common in practical mesh networks, existing state-of-art solutions either fail to correctly identify the sender that needs to be throttled or are too aggressive in reducing the sending rate. FairMesh is able to accurately detect unfairness quickly and employs a simple CW_{min} adjustment algorithm to achieve approximate max-min fairness. Our key insight is that the nodes that cause an unfair situation to arise and can act to remedy it are often distinct from the ones that can accurately assess the degree of unfairness. To the best of our knowledge, we are the first to decouple the detection and assessment of unfairness from the remedial action. A key strength of our approach is its *simplicity*, which makes it amenable for deployment in practical 802.11 mesh networks to allow an arbitrary number of flows to operate concurrently without modifications to the 802.11 MAC. We show via simulation and with experiments on a 20-node outdoor 802.11 wireless mesh testbed that FairMesh has many desirable properties. First, it is fully distributed and has negligible control overhead. Second, it achieves approximate max-min fairness, and can be modified to support a different notion of fairness (e.g., proportional fairness). Third, it can handle multiple (more than two) competing links and can scale up to mesh networks with tens of nodes. Fourth, it remains efficient under high data rates and high loss rates. Finally, FairMesh interacts well with TCP and maintains good fairness when a multi-hop flow competes with a single-hop flow.

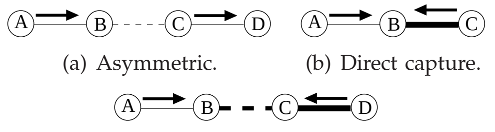
Index Terms—802.11 mesh network, Physical layer capture, Fairness.

1 Introduction

Ensuring fairness among competing flows in 802.11 mesh networks [17, 21, 22, 24] is a longstanding and difficult problem. The trouble with unfairness is that if left unchecked, it can often lead to the starvation of some flows, rendering wireless meshes practically unusable. Unfair behavior is known to arise under the asymmetric topology illustrated in Fig. 1(a). When node C has a backlog of data to send to node D, node A has little chance of successfully sending packets to node B as the data packets (or RTS) from node A would likely collide with the data packets from node C. Prior work has shown that the unfairness arising in this topology can be mitigated by adjusting the contention window CW_{min} [22, 24].

In our measurement study of a large number of flow pairs in a 20-node wireless mesh testbed, we discovered that in addition to the asymmetric topology, there are two other common mesh topologies that can cause significant MAC unfairness. The key difference between these topologies compared with the previously well-studied asymmetric topology [8, 16], is that the unfairness *arises because of physical layer capture (PLC)*, where packets can still be decoded correctly even in the event of packet collisions. While MAC unfairness arising from the capture effect was investigated in [15], the authors did not consider topologies with hidden-node collisions that are very common in mesh networks.

The two mesh topologies with capture-induced unfairness, which we refer to as *direct capture* and *indirect capture*, are illustrated in Fig. 1(b) and Fig. 1(c), respectively. PLC is surprisingly common. Lee et al. showed that a 1 dB difference in receive signal strength can result in link capture for Atheros adapters [28]. In our 20-node outdoor wireless mesh testbed with these adapters, 92.6% (87/94) and 18.6% (19/102) of the possible 3- and 4-node configurations exhibit direct capture and indirect capture, respectively.



(a) Asymmetric. (b) Direct capture. (c) Indirect capture.

Fig. 1. Topologies that can result in MAC unfairness. The arrows indicate the directions of the data flows, the bold lines indicate the captured links, and the dashed lines indicate overheard links.

In the direct capture scenario, node B is able to capture the packets from node C, and can successfully decode C's packets even if they collide with the packets from node A. This means that the link AB could be starved if node C has backlogged packets to node B. With RTS enabled, node A would have a lower throughput than C because C always wins in the RTS collision at B. In the indirect capture scenario, node C is able to capture node D's packets, and node B is able to capture node C's packets. As such, node C can receive D's RTS with high probability; also, the CTS from node C has a high probability of "overriding" any RTS from node A and preventing node B from sending CTS. This causes the throughput of the link AB to be significantly lower compared with that of link DC. Even if there is capture effect at node C but not at node B, link AB still has smaller throughput than link DC, albeit to a lesser degree.

For the rest of this paper, we will refer to link AB (in all the three topologies in Fig. 1) as the *victim* link and the link CD (in Fig. 1(a)), link CB (in Fig. 1(b)) and link DC (in Fig. 1(c)) as the *offending* links. The sender on the offending link is referred to as the *offender*.

Our extensive evaluation of two existing solutions [22, 24] reveals fundamental and practical shortcomings. As neither of them are explicitly designed to handle PLC, we found that they are able to achieve reasonable fairness in some, but not for all three, topologies in Fig. 1. Even in some scenarios where fairness is achieved, total throughput is reduced.

Motivated by our findings and the need for a simple, practical and distributed solution to mitigate unfairness in 802.11 mesh networks, we designed and implemented *FairMesh*, a new algorithm for adjusting the contention windows among competing flows to achieve approximate max-min fairness. Our key insight is that the nodes that cause an unfair situation to arise and can act to remedy it are often distinct from the ones that can promptly detect and accurately assess the unfairness. To the best of our knowledge, we are the first to decouple the detection and assessment of unfairness from the remedial action. A key strength of our approach is its *simplicity*, which makes it amenable for immediate deployment in practical 802.11 mesh networks using off-the-shelf commodity adapters.

FairMesh has many desirable properties: (i) it is fully distributed with negligible overhead; (ii) it works not only in all the three scenarios identified with two competing flows, with and without Binary Exponential Backoff (BEB), but is also scalable to more than two competing flows in a network with tens of nodes; (iii) it incorporates practical techniques to improve the performance in the presence of lossy links, high data rate, and TCP traffic, and (iv) it can be extended to other notions of fairness besides max-min fairness.

We show with a comprehensive set of experiments, both in simulation and on a 20-node outdoor 802.11 wireless mesh testbed, that FairMesh is not only more fair than 802.11 and prior work, but also achieves near-optimal max-min fairness allocation. The major contribution of this paper is thus a comprehensive, yet simple and practical solution to the longstanding problem of unfairness in 802.11 mesh networks.

The rest of this paper is organized as follows: in Section 2, we present our measurement study on the extent of unfairness in various scenarios. We describe how FairMesh works in Section 3, and evaluate FairMesh in Section 4. Finally, we present an overview of related work in Section 5 and conclude in Section 6.

2 Understanding MAC Unfairness

Unfairness among competing links arises only when the competing links have *backlogged traffic*, i.e., when there is an accumulation of packets at the transmission queue. Such a scenario is not uncommon, especially considering the prediction that the bandwidth-hungry video traffic is expected to make up 69% of the Internet traffic in 2017, more than half of which is carried by WiFi [2]. Thus, we are only concerned with backlogged-traffic scenarios in this paper.

2.1 Degree of Unfairness

To understand the degree of unfairness for the topologies cited in Fig. 1, we simulated the three topologies with the `ns-2` simulator with backlogged UDP flows using the default 802.11 parameters (CW_{min}= 15, CW_{max}= 1023, and 1,500-byte packets). We implemented the capture effect in our `ns-2` simulation model, by building on the `ns-2` enhancements by Chen et al. [12]. Since Atheros network adapters do not double the backoff window after failed transmissions [19], we also investigated the impact of not using BEB. We verified on our Atheros-based 802.11 testbed that the results without BEB are similar to that produced by the `ns-2` simulations, validating the simulation model. BEB cannot be enabled on our Atheros-based adapters, thus we can only use `ns-2` to evaluate scenarios with BEB enabled.

In Fig. 2, we compare the throughput of the victim link with the offending link on all three topologies, for data rates of 6 Mbps and 24 Mbps. With BEB enabled, the victim link in all three topologies is starved (i.e., has throughput values that are close to zero). With BEB disabled and RTS/CTS enabled, the victim link is not starved for the asymmetric and direct capture topologies (though unfairness still exists), but is still starved for the indirect capture topology. Without RTS/CTS, the victim links are starved for the first two topologies and the performance is poor for the indirect capture topology.

2.2 Design Decisions

A straightforward way to improve fairness is to adjust the waiting time between consecutive packets, so that the opportunity of the victim to access the channel can be increased. There are two important design decisions that we shall carefully address. The first design decision is whether to slow down the offender (i.e., increasing the offender's waiting time between its consecutive packets) or to speed up the victim (i.e., reducing the victim's waiting time). Through both testbed experiment and simulation, we found that speeding up the victim is not sufficient to ensure fairness and sometimes it could exacerbate the unfairness due to an increased likelihood of hidden-node collisions. We will show that slowing down the offender is more effective in achieving fairness. Note that the 802.11e standard employs a similar "reduced waiting time" mechanism to increase channel access priority, and is thus not too effective either.

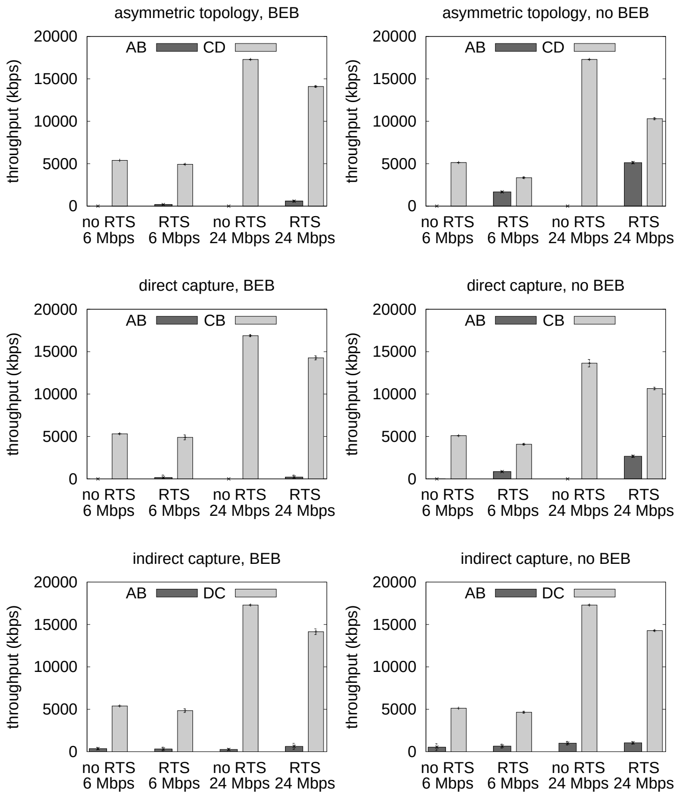


Fig. 2. MAC unfairness for topologies in Fig. 1.

The second design decision is which MAC parameter to adjust in order to slow down the offender. There are two possible options: Arbitrary Inter-Frame Space (AIFS) [9, 18] and CW_{min} [22]. The former specifies the minimum waiting time of a sender after the channel becomes idle, whereas the latter determines the random backoff time. Note that, unlike DIFS, AIFS is not fixed, e.g., from 2 to 7 in the 802.11e standard. We found that it is possible to mitigate unfairness by adjusting either AIFS or CW_{min} in the three topologies in Fig. 1. However, the drawback of adjusting AIFS is that it is possible for a sender with large AIFS value to be completely starved if a neighboring node starts transmitting with the default MAC parameters. This is because, when a sender is waiting for the AIFS period to expire and the channel becomes busy, the sender will start a fresh AIFS period later after the channel becomes idle. With a large AIFS, the AIFS period for the sender might not expire before it is repeatedly renewed, thereby causing starvation. On the other hand, a sender with large CW_{min} will not experience starvation because the backoff counter is frozen when the channel is busy and resumes decrementing after the channel becomes idle. Therefore, we choose to adjust CW_{min} instead of AIFS.

2.3 Impact of CW_{min}

To understand the effect of CW_{min} adjustment, we plot the average throughput of backlogged UDP traffic for the three different topologies with different combinations

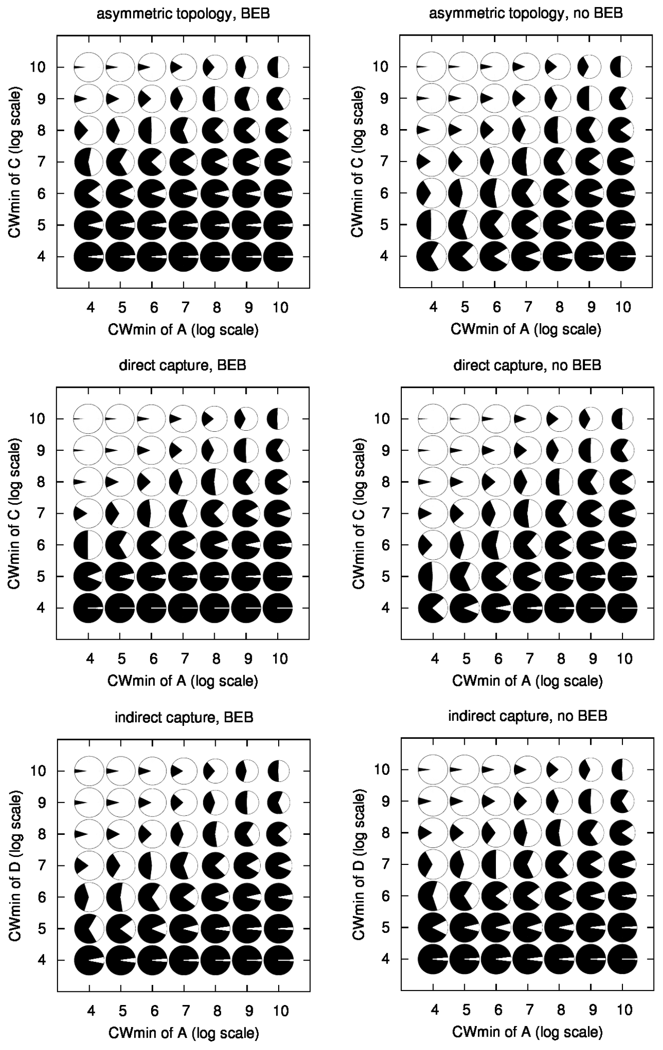


Fig. 3. Throughput under different combinations of CW_{min}, for the topologies in Fig. 1, with RTS/CTS. The values on the axes are the logarithms of CW_{min}. Larger pie size indicates better overall throughput. Identical size of black and white sectors means perfect fairness.

of CW_{min} obtained from `ns-2` simulations in Fig. 3. The CW_{min} ranges from 15 (default) to 1023 (default value of CW_{max}). The area of each pie is proportional to the total throughput. The white and black sectors represent the throughput of the victim and offending links, respectively.

We make several observations. First, as expected, increasing the CW_{min} of the two flows makes their throughput more equal but hurts the total throughput, i.e., the size of the pie reduces. To allow us to trade off the total throughput against the fairness between the two flows, some notion of fairness is required. In this paper, we focus on max-min fairness [7]. In Section 4.6, we show how we can extend our work to proportional fairness.

Second, the pie graph in Fig. 3 makes it easy to find the optimal CW_{min} combination for max-min fairness, but it is generally difficult to obtain similar figures for arbitrary topologies. Since 802.11 adapters often allow CW_{min} to be set only to a value of $2^k - 1$, where k is an integer, it is in principle possible to generate Fig. 3 via offline simulation/measurement. However, each figure is specific to the interaction between a given pair of flows. In practice, the number of possible interacting flows/links could be huge and it is not feasible to probe all possible CW_{min} values to determine the optimal values for each and every topology. In addition, although there are existing proposals [16, 42] to model the asymmetric topology in Fig. 1(a), it is not trivial to extend these models to arbitrary topologies.

Third, we note that the pies can be loosely divided into two contiguous regions: a black-dominant region and a white-dominant region. The optimal CW_{min} values to achieve max-min fairness must lie on the boundary between the two regions. This observation suggests a simple approach for approximating the optimal CW_{min} values: *increase the CW_{min} of the offender until we reach the boundary between the two regions and once we get there, we move along the boundary until we find the allocation that achieves max-min fairness.* While this idea is simple, implementing it in a distributed way for arbitrary pairs of links is not entirely straightforward and it involves many details, which we describe in the next section.

3 FAIRMESH

Unfairness can fundamentally only be mitigated by slowing down the offender(s) that are sending too fast, in order to provide the victim node(s) with a fair chance at accessing the wireless media. Since unfairness does not occur all the time and depends on both the network topology and traffic patterns, we should only intervene when unfairness arises, to avoid interfering with the normal operation of the 802.11 MAC.

In this light, the general approach to solve this problem would involve (i) detecting the existence of unfairness in a timely manner, (ii) identifying the offending link(s) accurately; and (iii) reducing the transmission rate of the offending link(s) appropriately. In a wireless mesh network, it is also preferable to achieve the above in a distributed manner without a central coordinator.

FairMesh detects unfairness and identifies the offending link(s) by continuously tracking and overhearing the transmissions to and from the nodes within its one-hop neighborhood. It turns out that simply overhearing the transmissions of one-hop neighbors is generally not sufficient to provide a node with an accurate view of the transmissions in its locality. As explained in Section 3.1, we insert an additional short header (between MAC and IP headers) and piggyback additional information in order to allow nodes to infer the transmissions of packets that are not successfully overheard.

A problem, not commonly considered in previous work, is the coordination among multiple parties that could detect unfairness concurrently in a practical wireless mesh network with multiple flows. To avoid multiple nodes from acting on the same problem and causing the throughput to be reduced excessively, FairMesh uses a distributed algorithm (described in Section 3.2) to elect a unique coordinating node. Also, we have found situations where the nodes that can detect an unfair situation are distinct from the ones that can remedy the problem.

Once an offending link is identified, it remains for FairMesh to throttle the offender to improve fairness. There are several possible notions of fairness. FairMesh adopts max-min fairness [7] because it is simple to implement and practical. We show in Section 4.6 that FairMesh can, in principle, be modified to support other notions of fairness, such as proportional fairness.

The performance of FairMesh depends critically on the amount of throttling, as it affects both the eventual efficiency and the fairness of the system. Our investigations into BEB and CW_{min} tuning, briefly described in Section 2, suggested that the adjustment of CW_{min}, according to the general principle of moving to the boundary of two unfair regions and then searching along the boundary, can achieve a good solution. In this light, we propose the following algorithm to throttle the offender: the coordinator sends a control message to the offender to double its CW_{min}. Once a change in CW_{min} is detected, the coordinator checks if the change improves the situation. If so, it informs the offender to further double the CW_{min}; otherwise, it informs the offender to roll back the earlier instruction. As we will explain in Section 3.3, it is challenging to get this algorithm to work in a general multiple-flow scenario, because after an offender is slowed down, another node may become the new offender.

We have considered different solutions based on these ideas and the final algorithm that we describe in this section represents a solution that achieves approximate max-min fairness in a distributed way, guided by the principles of simplicity and practicality. Our primary motivation is to develop a reliable algorithm that can allow an arbitrary number of flows to operate concurrently in a fair and consistent way, without reducing overall efficiency.

3.1 Estimating Throughput Accurately

The degree of unfairness observed at a node is measured using the number of *successfully* transmitted packets per second, which reflects the channel access opportunities of different senders. If the packet size is constant, the packet sending rate is directly proportional to throughput. We track the unfairness on a *per-link* basis rather than a *per-node* basis, as the latter would tend to penalize the nodes with many neighbors. Alternatively, we can enforce fairness on a per-node basis by aggregating per-link information, if desired. Correspondingly, each outgoing link from a node to a different neighbor also maintains an independent value of CW_{min}, which would be adjusted according to the algorithm to be described in Section 3.3.

Each node has knowledge about all the nodes within two hops, which can be maintained at negligible cost, since mesh networks are stationary and relatively stable. Each time a node transmits, receives, or overhears a data packet, the node updates the throughput estimates of all its observed traffic. A *sliding-window*-based approach is used to estimate the throughput: for a link AB, the number of packets successfully transmitted from node A to node B during a given time window is used as an estimate of the throughput for link AB. We investigated different time window sizes in our mesh testbed, ranging from several hundred milliseconds to a few seconds, and found that 0.5 to 1 s works well in practice, so we set the window size to 1 s in our algorithm.

While a node can accurately track the number of packets it has successfully transmitted to or received from a neighbor, the same cannot be said for overheard packets. Overheard packets are often corrupted or missed, since we are operating in a regime where collisions are common. This problem causes the throughput of an overheard link to be under-estimated. To address the issue, we introduce a new per-neighbor sequence number and piggyback this information in the packets by adding it into the FairMesh header, which is between the IP and MAC headers. This sequence number is incremented after every successful packet transmission and a node can estimate the throughput of an overheard link from the sequence number instead of by counting packets. If the packets are not of uniform size, it is straightforward to modify the implementation to piggyback the number of successfully transmitted bytes instead. Our approach works even if *rate adaptation* [39] is enabled because fairness is measured by the amount of data transmitted, not by the underlying transmission rate.

3.2 Detecting Unfairness

All nodes constantly monitor the traffic to and from its one-hop neighbors. It remains for us to use this information to (i) detect unfairness and situations where we can potentially improve fairness; and (ii) elect a coordinator to coordinate the CW_{min} adjustment.

Since unfairness only arises when competing links are backlogged, FairMesh needs only to consider backlogged links. To identify such links, we piggyback an additional bit in the FairMesh header to indicate if the transmission queue for a link is backlogged.

Identifying the Victim. The backlogged links with the lowest throughput values among all the flows observed within each node's neighborhood are the potential victim links. From the perspective of an observing node, a *potential victim* is defined as the link with the smallest throughput among all the observed links (i.e., among all the outgoing, incoming, and overheard links); and the offending link is a link that has a higher throughput than the victim and also has a *conflict* with the victim. By *conflict*, we mean that their senders cannot successfully transmit packets at the same time, i.e., a link has at least one node within the transmission range of either the sender or receiver of the other link [32]. Note that, if two links share the same sender, they are not considered conflicting links, since their packets share the same transmission queue and do not affect each other's probability of channel access.

Whether a potential victim link is really a victim depends on the notion of fairness. In the case of max-min fairness, which is what FairMesh implements by default, we consider the link with the largest throughput (denoted with l_O) that has conflict with the backlogged link with the lowest throughput (denoted with l_V). If the throughput of l_O exceeds that of l_V by more than a threshold value δ, we consider l_V to be a victim and execute the CW_{min} adjustment algorithm to reduce the rate of l_O. The threshold δ is determined by the estimated error in the throughput estimation, which depends on the size of the sliding window. If a victim link is not found, then a node does nothing and continues to monitor the flows within its neighborhood.

Coordinator Election. It is likely that several nodes simultaneously detect the same unfair situation (i.e., same l_O and l_V). In FairMesh, one of these nodes will become the coordinator and initiate the CW_{min} adjustment algorithm described in Section 3.3. To ensure that there is a unique coordinator for each offender-victim pair, a node determines whether it should become the coordinator according to whether the victim and offending links are incoming, outgoing or overheard links as well as rules $R1$ and $R2$, shown in the following table:

		l_V		
		Incoming	Outgoing	Overheard
l_O	Incoming	YES	$R1$	$R2$
	Outgoing	YES	Impossible	NO
	Overheard	YES	$R1$	NO

A cell with "YES" indicates that the node will become a coordinator; a cell marked $R1$ and $R2$ means that the node will check the following two additional rules to decide if it should be a coordinator. According to Rule $R1$, a node will become the coordinator if the receiver of the victim link cannot hear the offender. According to Rule $R2$, a node will become the coordinator if both the sender and receiver of the victim link cannot hear the offender. An example is shown in Fig. 4, where it is assumed that AB and CD are l_V and l_O, respectively. Nodes A, B, and C all detect the same unfair situation between AB and CD. From A's perspective, l_V (AB) is outgoing and l_O (CD) is overheard, and thus Rule $R1$ is checked. A is not elected since the receiver of l_V (B) can hear the offender (C). C is not elected either, and only B will be elected. The proof of uniqueness for the coordinator in other scenarios is relatively straightforward by enumeration. Our approach works even when the network topology changes, because the neighborhood information required for $R1$ and $R2$ is obtained from and kept up-to-date by the periodic Hello beacons.

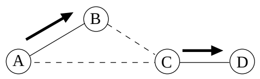


Fig. 4. Example of multiple nodes detecting the same unfair situation between l_V (AB) and l_O (CD).

Algorithm 1 Water-discharging CW_{min} Adjustment

```
1: $prev\_MIN \leftarrow 0$
2: while true do
3:     $MIN \leftarrow$ the throughput of $l_V$
4:     if $MIN > prev\_MIN$ then
5:         $l_O \leftarrow$ offender
6:         $prev\_MIN \leftarrow MIN$
7:         Double the $CW_{min}$ of $l_O$, using a small control message
8:         Wait for one time window
9:     else
10:        Undo the previous doubling, using a new control message
11:        Break
12:    end if
13: end while
```

3.3 CWmin Adjustment Algorithm

FairMesh's CW_{min} adjustment algorithm aims to achieve the max-min fairness between the identified victim and offending links. The coordinator sends a control message to the offender in order to adjust its CW_{min}. In addition to the per-neighbor sequence number (for throughput estimation) and backlog bit (to identify backlogged flows), each packet also contains the CW_{min}. This information allows the coordinator to determine whether its control message has been executed at the offender.

The classic solution to max-min fairness is the *water-filling* algorithm [7], originally proposed for wired networks. It is not feasible to apply this algorithm in our context, because we cannot increase the throughput of all conflicting links at the same rate. Instead, FairMesh takes a simpler and more practical approach inspired by our observations in Fig. 3: we gradually slow down the offender by doubling its CW_{min} to increase the victim's effective transmission opportunity, until the minimum throughput among all the links (denoted by MIN) cannot be improved. We call this the *water-discharging* algorithm, and the pseudocode for the case where there are two competing links is shown in Algorithm 1.

We illustrate the execution of the water-discharging algorithm for a sample asymmetric topology (see Fig. 1(a)) found in our 20-node wireless testbed in Fig. 5. The algorithm is manually activated after 7 s, and the coordinator B starts slowing down offending link CD. One time window (1 s) later, node B observes a better MIN and starts slowing down link AB, which then becomes the offending link. After another time window, a lower MIN is observed, so node B rolls back the previous decision and the algorithm terminates. The resulting CW_{min}'s turn out to be the optimal ones according to Fig. 3, and it took only three messages and about 3 s to find the CW_{min}.

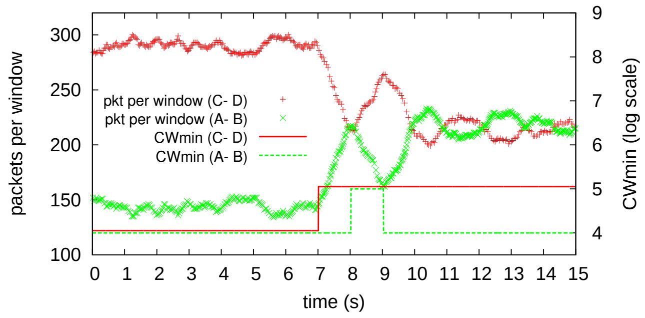


Fig. 5. The evolution of CW_{min} and the corresponding packet count per window, for a window size of 1 s. The CW_{min} values are in logarithm.

Multiple Offending Links. If multiple offending links exist, the coordinator will slow down the *fastest* offender. As the fastest offender is slowed down, another offender might end up taking its place as the new fastest offender, while the victim remains the same without any throughput improvement. In this case, the coordinator does not roll back the CW_{min} of the previous offender, but proceeds to slow down the new fastest offender. This process repeats until either (i) the MIN is improved, and then the coordinator moves to next iteration; or (ii) all offenders have been slowed down without any improvement to the MIN, and the coordinator rolls back all their CW_{min} in a batch before terminating.

Possible Abort. Note that if a coordinator detects an unexpected change in the CW_{min} for any of its monitored links when running the water-discharging algorithm, the algorithm will abort and the coordinator will roll back to the previous state with the best MIN. Again, this step avoids several coordinators from modifying the network state simultaneously.

Hysteresis. We found that the natural variation in the throughput estimates for one link is approximately 10% with our chosen window size of 1 s. In other words, the actual throughput for a link can be expected to vary as much as 10% from the estimated value, so we set δ, the threshold for l_O and l_V, at 10%. We also introduce a 10% hysteresis in the water-discharging algorithm and only consider a step to have improved the MIN when the new minimum value is more than 10% larger than the previous MIN. After each CW_{min} doubling, the coordinator will monitor the offending link to check that its control message is executed. Once executed, the coordinator will wait for one window of time to get a updated view of the throughput of the various links that it is monitoring before it executes the next iteration of the water-discharging algorithm.

Periodic Updates. After increasing CW_{min}, a link does not stay at this new CW_{min} value indefinitely, otherwise in the long term, it would itself suffer from unfairness. Rather, the link will halve its CW_{min} every τ time windows if the CW_{min} is larger than the default value of 15. This automatic decay of CW_{min} allows links to return to the default state over time after the original backlogged victim link stops transmissions. If an offender halves its CW_{min} prematurely and causes unfairness, the *water-discharging* algorithm would kick in and cause the CW_{min} to be inflated again. In our implementation, τ is set at 10.

3.4 Handling Indirectly Overheard Links

The basic solution presented above is based on the implicit assumption that, between two conflicting links, at least one node (of these two links) can hear both the senders. In other words, at least one node is aware of the actual degree of unfairness between the two links, using the throughput estimation technique described in Section 3.1. It turns out that for the indirect capture topology shown in Fig. 1(c), this assumption does not hold. Under this unfair situation, none of the nodes overhear both senders simultaneously and hence cannot detect the unfairness.

To detect the unfairness due to indirect capture, we enhance the throughput estimation algorithm to estimate the throughput of an *indirectly* overheard link by counting the ACK from the receiver of the indirect link. In particular, for the topology in Fig. 1(c), node B estimates the throughput of DC by counting the ACK from node C. A minor technical challenge is that an ACK message only contains the receiver's MAC address and not the sender's MAC address. To circumvent this problem, we use the RSSI reading of a node as a *signature* to identify the ACK sender. The RSSI reading is obtained from either the periodic Hello beacon or a recently transmitted data packet, and the RSSI changes on a time scale that is slow enough for this signature method to be relatively accurate. Note that this unknown-sender problem may not exist for the 802.11n standard, in which the Block ACK frame includes both the sender's and receiver's MAC addresses.

A drawback of this ACK counting method is that it tends to under-estimate the throughput, which makes it difficult to achieve the required max-min fairness. In addition, to address the indirect capture scenario, the coordinator cannot send control messages to the offender directly, but instead will need to have a common neighbor forward the control messages. For example in Fig. 1(c), when node B finds that link AB is the victim link and link DC is the offending link, it would slow down link DC by sending the control message to node D through node C.

To improve the accuracy of ACK counting, we also implemented a per-neighbor sequence number in the ACKs, like in the data packets. Our current implementation is naive and only works for a fixed packet size, but it is straightforward to support different packet sizes by incrementing the sequence number in byte count. Unfortunately, we are not able to modify the ACK in our 802.11 testbed directly, and are only able to implement this feature in our `ns-2` simulations.

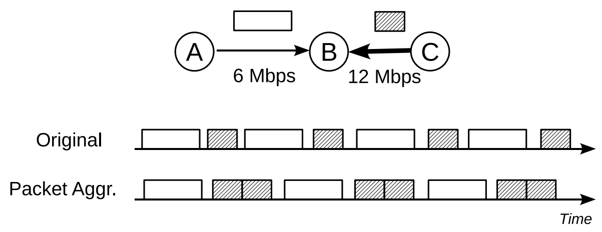


Fig. 6. Illustration of packet aggregation.

3.5 Optimizations

In this section, we describe the optimizations to handle high data rates and also to improve TCP performance.

Handling High Data Rates. FairMesh enables RTS/CTS to mitigate hidden node collisions, which can lead to significant overhead at high data rates. Because we are working with backlogged transmissions, we implemented *packet aggregation* in order to improve efficiency at the higher data rates. This idea is not new and a similar technique has been incorporated in the 802.11n standard: instead of sending one packet per RTS/CTS, we send several packets per RTS/CTS for the higher rates to improve overall throughput.

In addition to reduced overhead, packet aggregation also allows FairMesh to achieve approximate max-min fairness in terms of *transmission air time* instead of throughput. As illustrated in Fig. 6, link CB transmits multiple packets during one RTS/CTS, so that the packets from AB and CB have comparable transmission times in the air even though they are sending at different data rates. Although with packet aggregation, the lower rate link AB sends fewer packets, the time it gives up is more efficiently utilized by the higher rate link CB, thereby improving the overall throughput (in $kbps$). In other words, with packet aggregation, FairMesh is able to achieve approximate max-min fairness in terms of transmission air time and does not unduly penalize the link with a higher data rate.

Preventing TCP Starvation. Significant contention in a wireless mesh often leads to high packet loss, causing TCP to go into exponential backoff [31] and to starve. We found that we can prevent TCP starvation if we retransmit failed packets up to three more times using a second hardware queue, even after 802.11 gives up. This method keeps the packet loss rate below 5%. We are mindful that additional retransmissions will increase latency and may cause TCP timeouts, but we found that three retransmissions achieves a good trade off between latency and reliability in practice.

4 Evaluation

We now present our evaluation of FairMesh through an extensive set of experiments, both on an 802.11 wireless mesh testbed and in `ns-2` simulation. We begin with evaluating FairMesh using the three basic problematic topologies in Section 4.2. We then evaluate FairMesh with more complex topologies, consisting of multiple competing links, in Section 4.3. In Section 4.4, we compare FairMesh with two existing proposals to mitigate unfairness. Section 4.5 evaluates the interaction between links with different data rates under FairMesh. The performance of FairMesh in the presence of high packet loss rates is evaluated in Section 4.6. We then extend the evaluation to large topologies (on our 20-node testbed and a 50-node Berlin topology [30] in simulation) in Section 4.7. Finally, the impact of FairMesh on multi-hop TCP is evaluated in Section 4.8.

4.1 802.11 Wireless Mesh Testbed

Our testbed is deployed in a college dormitory at the National University of Singapore [4] over an area that is approximately 350 m $\times$ 200 m. The dormitory consists of tall and dense buildings, and thus 802.11 radio signals experience significant attenuation. The network has an average node degree of 4.5 and the network diameter is 6 hops. The deployment map of our testbed can be found in [41].

Hardware. The testbed consists of 20 ALIX boards [1]. We use Compex IEEE 802.11abg adapters [3], with the Atheros AR5414 chipset. All the experiments are conducted using the 802.11a mode. The Atheros adapters have several characteristics that are worth highlighting. First, the BEB mechanism in our adapters cannot be enabled [19]. As such, we have to resort to ns−2 simulations to investigate the behavior of FairMesh with BEB. Second, the capture effect in our Atheros adapters is similar to that of the adapters in [28]. Third, like the TFA testbed [10], our Atheros adapters do not employ energy detection in the Clear Channel Assessment (CCA) mechanism. Instead, the channel is considered busy if the hardware is able to successfully decode the preamble and the PLCP header. In other words, the carrier sense range is the same as the transmission range at 6 Mbps.

Software. The board runs OpenWRT Kamikaze 7.09 with kernel version 2.6.25. The driver for the wireless adapter is MadWifi (version 0.9.4) with a default MAC retry limit of 10 and runs in monitor mode. We modified the MadWifi driver to insert a small 4-byte FairMesh header between MAC and IP headers (as shown in Fig. 7), and also to support per-packet CW_{min} adjustment. To avoid the performance degradation described in [35], both the Transmit Antenna Diversity and the Ambient Noise Immunity features were disabled. FairMesh was implemented using the Click modular router [13] (version 1.6.0) in user space. Iperf was used to generate traffic and the default packet size is 1,500 bytes. The transmission queue size in our FairMesh implementation is 50 packets.

4.2 Basic Scenarios

We first run FairMesh on the three problematic topologies in Fig. 1 to understand its basic behavior, and compare it with 802.11 with and without RTS/CTS, and with and without BEB. Note that since the testbed does not support BEB, the evaluation without BEB is done using the testbed, while the evaluation with BEB is done in ns−2 simulation using a configuration similar to that on the testbed. In all our experiments in this paper (except in Section 4.8), all source nodes send saturating UDP traffic at the maximum data rate because our goal is to evaluate unfairness under worst case conditions. The average throughput values are summarized in Fig. 8. The error bars indicate the magnitude of the standard deviation in the data points, each of which is the average throughput over a 1-s interval. Each experiment lasts for 300 s and produces a total of 300 data points.

FairMesh vs. 802.11. As shown in Fig. 8, there is significant unfairness in all three topologies without RTS/CTS. The victim link AB is starved in most cases. As before, we see that BEB will also exacerbate the unfairness. With BEB disabled, RTS/CTS can significantly improve the throughput for the asymmetric topology and somewhat improve that for the two capture topologies. On the other hand, FairMesh is able to achieve significantly better fairness in all scenarios, without affecting the total throughput. In view that the performance of 802.11 without RTS/CTS is so bad, 802.11 will be used with

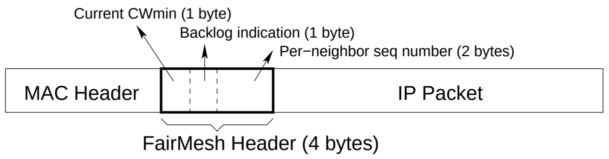

Current CWmin (1 byte)
Backlog indication (1 byte)
Per−neighbor seq number (2 bytes)
MAC Header
IP Packet
FairMesh Header (4 bytes)

Fig. 7. Format of FairMesh header in our implementation.

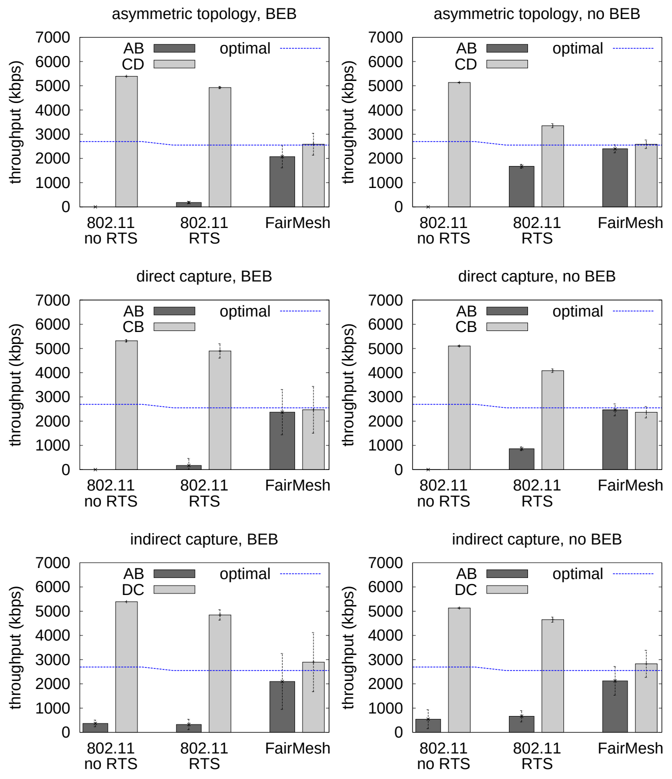


Fig. 8. Comparison between 802.11 and FairMesh: BEB enabled in simulation, and BEB disabled in testbed.

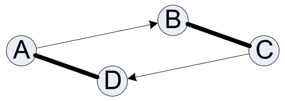


Fig. 9. Scenario where disabling BEB results in catastrophic failure. The packets from A and C are captured by D and B, respectively.

RTS/CTS enabled by default for the remainder of this paper.

To BEB or not to BEB? FairMesh seems to be able to achieve similar or better fairness without BEB in all cases. For 802.11 with RTS/CTS, the improvement in fairness is especially significant when BEB is disabled. This makes it tempting to completely disable BEB.

It turns out, however, that if we disable BEB, there are scenarios which can result in catastrophic failure and complete starvation. One such scenario is illustrated in Fig. 9. In this example, A is trying to send packets to B, and C is trying to send packets to D. The packets from A and C are captured by D and B, respectively.

When RTS/CTS is enabled and BEB is disabled, B and D will keep overhearing RTS packets from C and A respectively. This will cause the NAV at both B and D to increase progressively and prevent them from sending CTS packets to A and C. Consequently, both AB and CD will get starved.

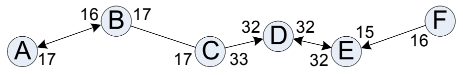
When RTS/CTS and BEB are both disabled, each sender will continuously send data packets that collide and corrupt the data packet of the other. Since the default inter-packet gap is small (because BEB is disabled) compared to the data packet, the packets will always collide, which again results in starvation.

We confirmed that this scenario indeed results in catastrophic failure when BEB is disabled on both our testbed and in simulation. Unfortunately, FairMesh is not able to improve the situation for this scenario because we cannot even detect the traffic to begin with. Overall, our view is that although it is possible to mitigate unfairness to some extent by disabling BEB, caution has to be exercised when doing so in practice.

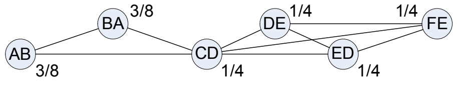
4.3 Optimal Capacity & Multiple Links

Having shown that FairMesh performs well in the three basic topologies with two competing links, we evaluate FairMesh in a more complex scenario with multiple competing links. We chose a 6-node topology from our testbed (see Fig. 10(a)) and configured each node to transmit to a neighbor. We replicated this topology in the `ns-2` simulator to evaluate the performance of the same topology with BEB enabled.

To compute the optimal max-min allocation of throughput for this topology, we used a classic *conflict-graph*-based algorithm [6, 22]. The vertices in a conflict graph are the links with backlogged traffic, and there exists an edge between two vertices if the associated



(a) Network topology.



(b) Conflict graph.

Fig. 10. Network topology and its conflict graph: the numbers in (a) are RSSI values, and the numbers in (b) are the nominal capacity for the links.

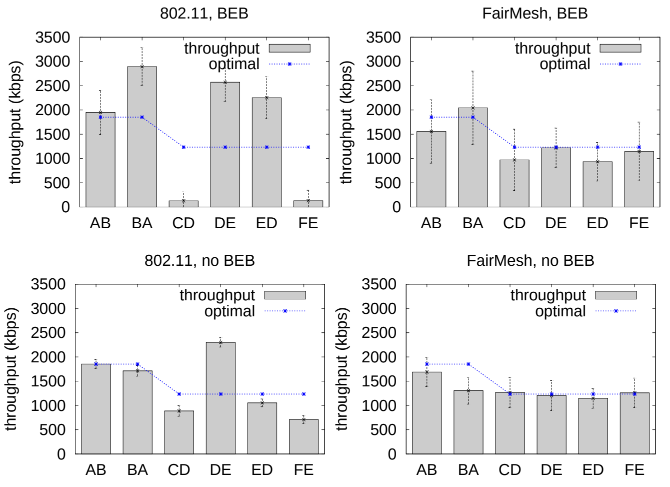


Fig. 11. Actual throughput and the optimal allocation: with BEB in simulation, and without BEB in testbed.

links are in conflict with each other[1]. Each *maximal clique* in the conflict graph initially has a nominal capacity of 1. The algorithm iteratively computes the nominal capacity assigned to each node in the conflict graph (or each link in the actual graph). To convert to the absolute capacity, the nominal value is multiplied by the maximum link throughput of 5,100 kbps. The corresponding conflict graph of the 6-node topology and the computed optimal nominal capacity for each link are shown in Fig. 10(b).

Fig. 11 compares the performance of FairMesh with 802.11 in the above topology both on the testbed (with BEB disabled) and `ns-2` simulator (with BEB enabled). The computed optimal throughput is plotted as well. The results show that the throughput of FairMesh is much closer to the optimal capacity compared with 802.11. This is especially true for the case with BEB, where CD and FE are starved for 802.11. We verified that `ns-2` yielded similar results as our testbed for the case with BEB disabled.

Table 1 summarizes the median CW_{min} values of the six links and the total number of control messages sent by each node during the experiment. The CW_{min} values with BEB enabled are generally larger than those with BEB disabled, as the victim links contend much less ag-

1. On our testbed, we consider two links to be in conflict if one link has a node (either sender or receiver) that has a packet delivery ratio of at least 50% to either the sender or receiver of the other link at 6 Mbps.

TABLE 1
Summary of median CW_{min} values for each link & total number of control messages from each node.

CW_{min}	links	AB	BA	CD	DE	ED	FE
	BEB	15	63	63	255	511	255
	no BEB	31	31	63	31	31	15
Overhead	nodes	A	B	C	D	E	F
	BEB	28	36	26	32	21	22
	no BEB	10	77	42	23	37	45

gressively with BEB enabled. We make two observations about the number of control messages initiated by each node. First, every node was elected at some point as a coordinator, since we can see that they each sent some control messages. Second, compared with the number of data packets (approximately 33,000 per node for the whole experiment), the number of control messages is insignificant, demonstrating that FairMesh incurs very little overhead.

4.4 Comparison with Prior Work

Next, we compare FairMesh with two previous proposals that also address MAC unfairness, one by Huang and Bensaou [22] (denoted with HB), and another by Jian and Chen [24] (called PISD), to demonstrate that FairMesh performs better than existing solutions for mitigating MAC unfairness. The HB method uses naive packet counting to estimate throughput, and does not consider throughput at the granularity of individual links like FairMesh. Besides, for the HB method, a node only adjusts its own CW_{min}, and thus is unable to handle the indirect capture scenario. PISD tries to do away with overhearing by emulating the AIMD mechanism in 802.11 networks. Links with queue lengths larger than a threshold would use a small CW_{min} to jam the channel, thereby creating a congestion signal. We found, however, that such jamming signals are not always propagated to the appropriate nodes, especially when BEB is enabled.

Both HB and PISD were implemented for the `ns-2` simulator. Our implementation of HB is an approximate one—the original proposed algorithm has a component that estimates the max-min fair throughput; in our simulation, we implemented an oracle that feeds the exact values for the max-min fair throughput into the main HB algorithm. Our HB implementation is thus equally or more accurate than what the proposed HB can achieve. Both HB and PISD operate with RTS/CTS enabled.

Fig. 12 shows the comparison of FairMesh to both HB and PISD, using the three topologies in Fig. 1. HB performs well for the asymmetric topology, but is less efficient for the direct capture topology. It is also unable to react correctly for the indirect capture topology. PISD is unable to achieve fairness in the asymmetric topology with BEB enabled. This is because the jamming signal from node A does not propagate promptly to node C, and link AB reduces its rate more frequently than link CD. We discovered that most of the evaluations in [24]

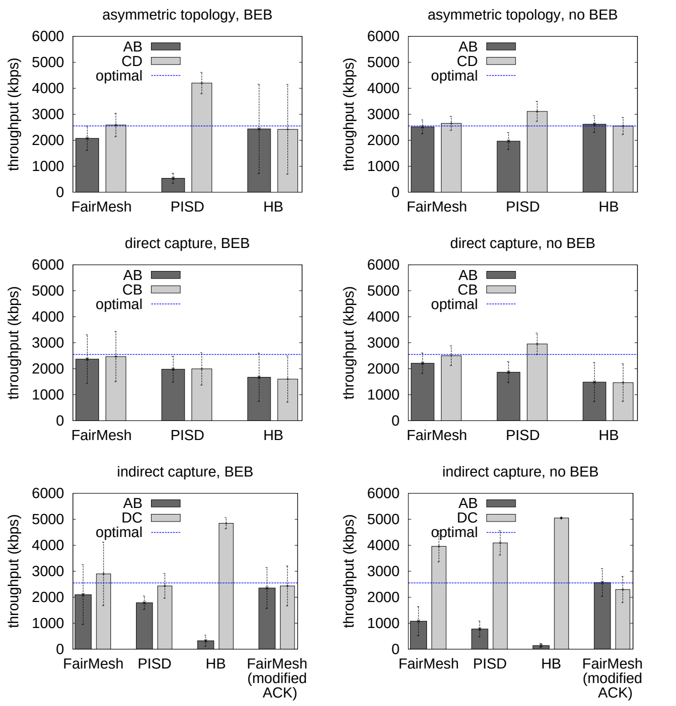
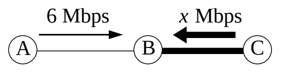


Fig. 12. Comparison of FairMesh to HB and PISD for all three problematic topologies in simulation.

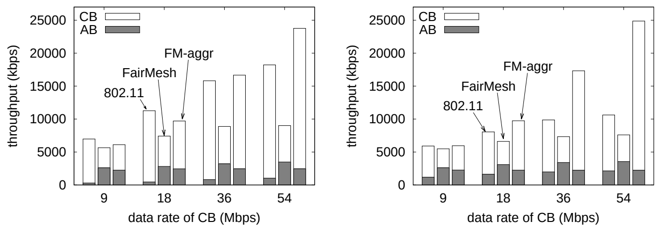
were conducted with the senders all *within* the carrier sense range of each other, which is why jamming was reported to be effective. PISD performs better in the direct capture scenario because the CTS from node B to node C can be overheard by node A, which prevents node A from "blindly" accessing the channel as in the asymmetric topology case.

If we compare the results of FairMesh for the indirect capture scenario without BEB in the testbed (shown in Fig. 8) with that in the `ns-2` simulation (shown in Fig. 12), we see that the improvement in the fairness for FairMesh is better for the testbed. We found that this is because node B overheard fewer of node C's ACKs in the simulation, while node B was able to successfully overhear more of these ACKs in the testbed. Nevertheless, in both cases, FairMesh is still slightly more fair than both HB and PISD.

In Section 3.4, we explained that we can improve the accuracy of ACK counting by introducing a per-neighbor sequence number in ACKs, but because we cannot modify hardware ACKs, we can only implement this ACK modification in simulation. In Fig. 12, we also include the results of FairMesh with the modified ACK for the indirect capture topology. Because the sequence number in ACK enables node B to accurately assess the throughput of DC, FairMesh is now able to achieve almost equal share between AB and DC. Note that the modified ACK only provides additional information to assist in the throughput assessment of indirectly overheard links, and it does not interfere with the performance of FairMesh in other topologies. For the rest of the paper, we will not consider the modified ACK



(a) Higher data rates scenario.



(b) BEB (c) no BEB

Fig. 13. Evaluation of 802.11, FairMesh, and FairMesh with packet aggregation (FM-aggr) under selected higher data rates via simulation.

optimization to keep the FairMesh implementation for the testbed consistent with that for the `ns-2` simulation.

4.5 Higher Data Rates

To evaluate FairMesh with higher data rates, we consider the direct capture scenario shown in Fig. 13(a), where link AB uses 6 Mbps and the stronger link CB uses higher data rate of x Mbps, where $9 \leq x \leq 54$. To implement the packet aggregation technique in Section 3.5, we aggregate multiple packets of link CB into one so that links AB and CB have the same air time per packet. In this way, FairMesh ensures fairness in terms of *transmission air time* of each link, instead of the original *throughput*-based fairness.

Fig. 13(b) and Fig. 13(c) show the results of 802.11, FairMesh, and FairMesh with packet aggregation (labelled as "FM-aggr"), via simulation (we are unable to do packet aggregation on the testbed due to the limited MTU size of our hardware). As compared with 802.11, the original FairMesh achieves almost the same throughput between AB and CB (i.e., better fairness). However, the total throughput of FairMesh is smaller than that of 802.11 (i.e., reduces overall efficiency). This is because FairMesh re-allocates certain amount of transmission air time from the fast link CB to the slow link AB. With packet aggregation, FairMesh significantly increases the total throughput, with only a slight drop in the throughput for AB.

4.6 Lossy Links & Proportional Fairness

Next, we demonstrate that FairMesh works well even with lossy links, and that FairMesh can be easily modified to work with a different notion of fairness besides max-min fairness.

Take the sample testbed topology in Fig. 14(a) for example, where node B can capture C's packets and link AB has a loss rate of 42% due to poor RSSI. FairMesh is still able to produce comparable throughput for the two links. However, the transmission opportunity given up by link CB does not translate into the same number of effective transmissions for link AB. With 42% losses, for every 5 transmissions given up by link CB, link AB would only gain approximately 3 successful transmissions. This leads to a significant loss in overall efficiency and this is one situation where proportional fairness [26] would allow us to trade off fairness for better efficiency.

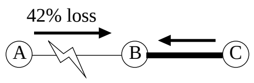
FairMesh can be easily modified to support proportional fairness, by checking whether a *utility function* $\sum ln(r_i)$ improves in the water-discharging algorithm, rather than the MIN, where r_i is the throughput of an observed link i. Fig. 14(b) and Fig. 14(c) show the results of FairMesh (max-min), FairMesh (proportional fairness) and 802.11 for the scenario in Fig. 14(a). Compared with FairMesh (max-min), FairMesh (proportional fairness) achieves a higher total throughput at the expense of slightly less equal allocations between AB and CB.

4.7 Large-Scale Experiments

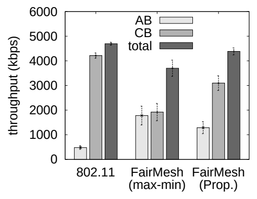
Having shown that FairMesh performs well in the three basic topologies with two competing links, we evaluated FairMesh in a more complex scenario with *large-scale arbitrary competing links*. We used our 20-node testbed to study the case without BEB, and used `ns-2` simulation to study the case with BEB. In these experiments, each node sends saturating UDP traffic to one randomly chosen neighboring node. The point is not to attempt to simulate real traffic, but to investigate how FairMesh and existing algorithms hold up under conditions of severe contention, where unfairness is most likely to manifest. As a benchmark for comparison, we used the classic *conflict-graph*-based algorithm [22] to compute the optimal max-min allocation of throughput.

Testbed. We randomly selected 20 links in the 20-node testbed to evaluate FairMesh with multiple concurrent flows. Fig. 15 shows the performance of 802.11 and FairMesh on the testbed. We plot the optimal max-min throughput allocation with a blue dashed line. The links are sorted in descending order of throughput. We observed that the distribution of the link throughput for FairMesh is significantly closer to the optimal max-min fair allocation. Also, the overall throughput of 802.11 is slightly higher than that of FairMesh, as the throughput reduction of one link can cause a throughput increase in multiple links, i.e., 802.11 exploits spatial diversity more efficiently at the expense of fairness. The total rate of the control messages for the entire network is about 1.48 packets per second, which is negligible compared to the data throughput.

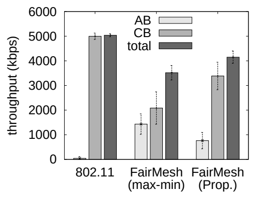
Simulation. We next performed a large-scale simulation study of FairMesh on a 50-node topology with an average degree of 3.6, generated using the Berlin network methodology [30]. This simulation allows us to study the behavior of FairMesh with BEB in a large network, as well as to compare FairMesh with HB and PISD. Like before, each node randomly selects a neighbor as receiver and sends saturating UDP traffic. We



(a) Lossy link scenario.

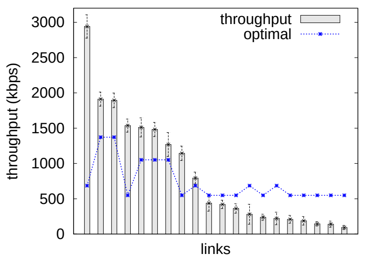


(b) no BEB (testbed).

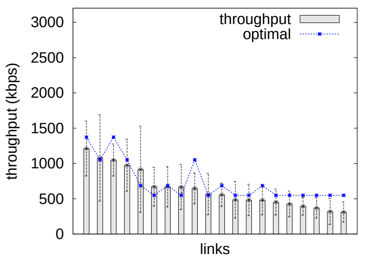


(c) BEB (simulation).

Fig. 14. Scenario with lossy link: max-min and proportional fairness have different impacts.



(a) 802.11



(b) FairMesh

Fig. 15. Comparing FairMesh to 802.11 in the real 20-node testbed. The links are sorted according to their throughput results in descending order.

plot the resulting throughput in Fig. 16. Again, FairMesh achieves a throughput allocation that is much closer to the optimal max-min fair allocation than 802.11, which has a large number of nearly starved links. To help us visualize the differences in fairness among FairMesh, HB and PISD, we also plot the CDF of the throughput ratio between the achieved throughput and the optimal max-min throughput, for the 50 links in Fig. 17. FairMesh achieves a throughput allocation that is closer to the optimal max-min allocation than both PISD and HB. In particular, PISD does not perform well in terms of fairness improvement. On the other hand, while HB achieves better fairness, it has poorer overall efficiency.

We investigated the convergence performance of FairMesh. As a link in FairMesh periodically halves its CW_{min} to increase throughput, its CW_{min} does not converge to a fixed value but varies within certain range of the possible seven CW_{min} levels (i.e., from 4 to 10 in logarithmic value). To evaluate the variation of CW_{min} in FairMesh, we recorded the interdecile range (i.e., from 10% to 90%) of each link's CW_{min} logarithmic values during the 300-s experiment above. We found that the majority of the links (88%) have their interdecile range of CW_{min} equal to or less than 3. This implies that FairMesh does not cause significant fluctuations in the CW_{min}.

4.8 TCP & Multi-Hop Flows

Finally, we show that FairMesh can improve the fairness perceived by TCP. To evaluate the efficacy of the additional retransmissions in FairMesh (see Section 3.5), we compared FairMesh with 802.11 and the *counter-starvation policy* proposed by Shi et al. [34] (denoted as "fixed CW_{min}"). To implement Shi et al.'s counter-starvation policy, we set the CW_{min} for the nodes near the gateway to 255. For our evaluation, we randomly selected various "1-hop vs. N-hop" topologies ($1 \leq N \leq 4$) from our testbed (see Fig. 18(a)), and ran two concurrent TCP flows on them. Typically, the N-hop flow not only experiences contention within itself but also gets overwhelmed by the 1-hop flow, because of the capture effect at the destination (which acts like a gateway).

Fig. 18 summarizes the results. As expected, all the N-hop TCP flows are starved for vanilla 802.11 due to frequent TCP exponential backoff. The counter-starvation policy improves fairness for $N = 1$ and $N = 2$, but not for $N > 2$. In fact, we found that the 4-hop TCP flow occasionally got starved, as the packet loss rate was still relatively high. While 802.11 with retransmissions can prevent starvation, the N-hop TCP flows suffer significant throughput degradation because the underlying MAC unfairness is not addressed. FairMesh with retransmission improves the TCP throughput for the N-hop TCP flows significantly, especially for large N. An interesting observation is that, because FairMesh seeks to achieve per-link fairness, it achieves almost equal throughput for $N = 1$ and $N = 2$, i.e., where all the links compete with each other and end up sharing the air time equally.

5 Related Work

Prior work on PLC mostly focused on characterizing its effect. Lee et al. studied Wistron 802.11a adapters with the Atheros chipset [28], which have similar characteristics as our adapters. Ganu et al. were the first to evaluate how manually changing various MAC parameters affects the unfairness arising from PLC, but they did not consider the topologies with hidden-node collisions and offered no solution for automatically adjusting these parameters to improve fairness [15]. Chang et al. modelled the effect of PLC on slotted ALOHA MAC [11]. Han et al. [20] also studied the effect of PLC on 802.11 WLANs, but they did not address hidden nodes collisions. PLC has been widely observed in other 802.11 adapters [10, 25, 27, 36], and even in sensor radio [38] and cellular systems [43].

Much prior work on improving the fairness in 802.11 networks exists. One category of proposals *directly* tackles the intrinsic MAC unfairness by adjusting the contention window. Liu et al. proposed a utility-based optimal CSMA scheme [29], but testbed experiments in [33] showed that this scheme does not solve the unfairness

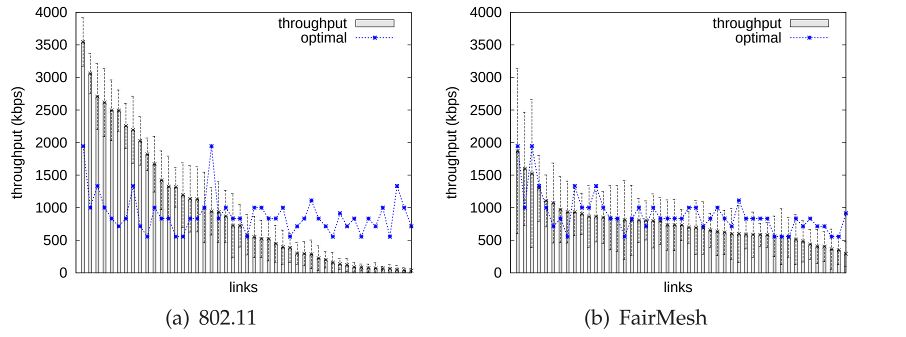


Fig. 16. Comparing FairMesh to 802.11 (with BEB) via simulation. The links are sorted according to their throughput results in descending order.

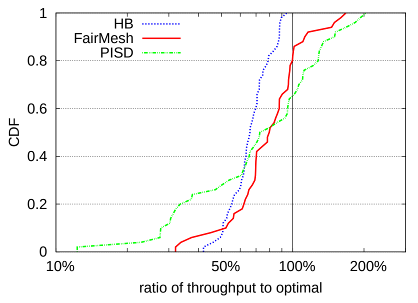


Fig. 17. CDF of throughput ratio to optimal, of the large-scale simulation experiment.

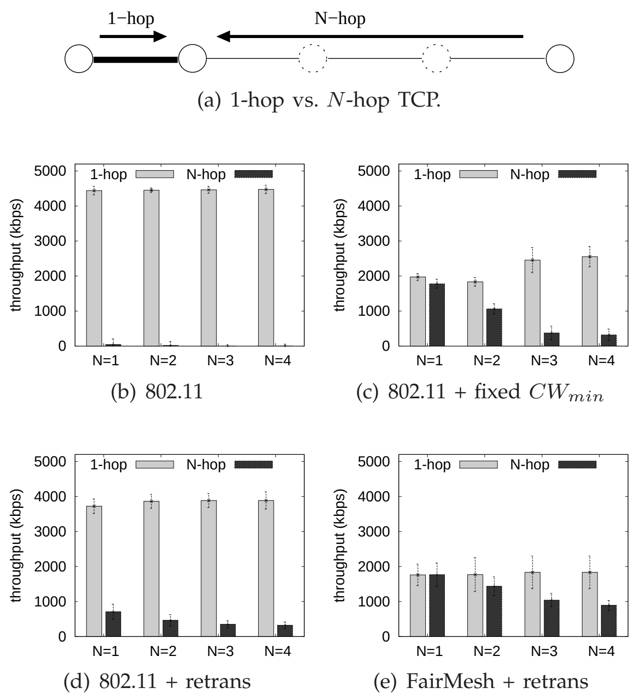


Fig. 18. Impact of FairMesh on TCP: 1-hop TCP vs. N-hop TCP. The 1-hop flow has higher RSSI and its packets can be captured by the common destination. Maximum 1-hop TCP throughput is 4,600 kbps.

problem when the senders are hidden from each other. Haas et al. [17] and Heusse et al. [21] considered the case with no hidden node collisions, while Huang and Bensaou [22] and Jian and Chen [24] considered more general scenarios with hidden nodes. As shown in Section 4, FairMesh works with hidden nodes and achieves better performance than the latter two schemes [22, 24].

Another category of work attempts to directly limit the rate at which packets are passed down to the MAC layer, thereby reducing the number of backlogged senders and circumventing the MAC unfairness problem. Xu et al. tried to limit this rate by setting appropriate values for TCP congestion window [40]. Adaptive pacing [14] tries to control the sending rate from the gateway by scheduling a packet every "four-hop propagation delay" down to the MAC of the gateway. The Cooperative Neighborhood Airtime-limiting (CNA) [23] scheme continuously allocates airtime usage based on the rate at which a node is sending packets to its MAC layer. FairMesh is compatible and complementary to these higher-layer fairness solutions.

Besides improving MAC fairness, CW_{min} adjustment can improve the performance of the transport layer in wireless meshes. Previous works have adopted this technique to prevent TCP starvation [34], to reduce congestion along a multi-hop path [37], and to smoothen out the fluctuation of the queue length along a multi-hop path [5]. None of these address the unfairness issue *comprehensively* – they either assume a specific topology [34] or do not explicitly address the unequal channel access opportunity at the MAC layer [5, 37].

6 Conclusion

In this paper, we investigate the MAC unfairness problem for 802.11 mesh networks arising from physical layer capture. By improving throughput estimation, and employing a per-neighbor CW_{min} adjustment mechanism, FairMesh is able to achieve a channel access allocation that closely approximates optimal max-min fairness in a distributed manner. We show that FairMesh can effectively mitigate unfairness in various practical scenarios. Since the root causes (asymmetric topologies and capture effect) of the MAC unfairness in 802.11 are also present in many other wireless technologies, we believe that the basic design of FairMesh would be broadly applicable to other networks beyond 802.11 mesh networks.

References

[1] ALIX system board (ALIX2c2) by PCEngines. http://www.pcengines.ch/.
[2] Cisco visual networking index forecast (2012-2017). http://www.cisco.com/go/vni.
[3] miniPCI Wireless Adapters (WLM54AG-23) by Compex. http://www.compex.com.sg/.
[4] NUS Mesh Networks. http://mesh.ndslab.net.
[5] A. Aziz, D. Starobinski, P. Thiran, and A. El Fawal. EZ-Flow: removing turbulence in IEEE 802.11 wireless mesh networks without message passing. In *Proceedings of CoNEXT '09*, Dec. 2009.
[6] A. Bar-Noy, A. Mayer, B. Schieber, and M. Sudan. Guaranteeing fair service to persistent dependent tasks. In *Proceedings of SODA '95*, Jan. 1995.

[7] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.
[8] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a media access protocol for wireless LAN's. In *Proceedings of SIGCOMM '94*, Aug. 1994.
[9] G. Bianchi, I. Tinnirello, and L. Scalia. Understanding 802.11e contention-based prioritization mechanisms and their coexistence with legacy 802.11 stations. *IEEE Network*, 19(4):28–34, Jul.-Aug. 2005.
[10] J. Camp, V. Mancuso, O. Gurewitz, and E. W. Knightly. A measurement study of multiplicative overhead effects in wireless networks. In *Proceedings of INFOCOM '08*, Apr. 2008.
[11] H. Chang, V. Misra, and D. Rubenstein. Fairness and physical layer capture in random access networks. In *Proceedings of SECON '07*, Jun. 2007.
[12] Q. Chen, F. Schmidt-Eisenlohr, D. Jiang, M. Torrent-Moreno, L. Delgrossi, and H. Hartenstein. Overhaul of IEEE 802.11 modeling and simulation in ns-2. In *Proceedings of MSWiM '07*, Oct. 2007.
[13] Click. The click modular router. http://read.cs.ucla.edu/click/.
[14] S. M. ElRakabawy, A. Klemm, and C. Lindemann. Gateway adaptive pacing for TCP across multihop wireless networks and the Internet. In *Proceedings of MSWiM '06*, Oct. 2006.
[15] S. Ganu, K. Ramachandran, M. Gruteser, I. Seskar, and J. Deng. Methods for restoring MAC layer fairness in IEEE 802.11 networks with physical layer capture. In *Proceedings of REALMAN '06*, May 2006.
[16] M. Garetto, J. Shi, and E. W. Knightly. Modeling media access in embedded two-flow topologies of multi-hop wireless networks. In *Proceedings of MobiCom '05*, Aug. 2005.
[17] Z. Haas and J. Deng. On optimizing the backoff interval for random access schemes. *IEEE Transactions on Communications*, 51(12):2081–2090, Dec. 2003.
[18] B. Han, L. Ji, S. Lee, R. R. Miller, and B. Bhattacharjee. Channel access throttling for improving WLAN QoS. In *Proceedings of SECON '09*, Jun. 2009.
[19] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller. Maranello: practical partial packet recovery for 802.11. In *Proceedings of NSDI '10*, Apr. 2010.
[20] S.-J. Han, T. Nandagopal, Y. Bejerano, and H.-G. Choi. Analysis of spatial unfairness in wireless LANs. In *Proceedings of INFOCOM '09*, Apr. 2009.
[21] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs. In *Proceedings of SIGCOMM '05*, Aug. 2005.
[22] X. L. Huang and B. Bensaou. On Max-Min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation. In *Proceedings of MobiHoc '01*, Oct. 2001.
[23] K.-Y. Jang, K. Psounis, and R. Govindan. Simple yet efficient, transparent airtime allocation for TCP in wireless mesh networks. In *Proceedings of CoNEXT '10*, Nov. 2010.
[24] Y. Jian and S. Chen. Can CSMA/CA networks be made fair? In *Proceedings of MobiCom '08*, Sep. 2008.
[25] G. Judd and P. Steenkiste. Characterizing 802.11 wireless link behavior. *Wireless Networks*, 16(1):167–182, Jan. 2010.
[26] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, Mar. 1998.
[27] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11b. In *Proceedings of ICNP '04*, Oct. 2004.
[28] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of WiNTECH '07*, Sep. 2007.
[29] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. Poor. Towards utility-optimal random access without message passing. *Wireless Communications and Mobile Computing*, 10(1):115–128, Jan. 2010.
[30] B. Milic and M. Malek. NPART - node placement algorithm for realistic topologies in wireless multihop network simulation. In *Proceedings of Simutools '09*, Mar. 2009.
[31] A. Mondal and A. Kuzmanovic. Removing exponential backoff from TCP. *SIGCOMM Computer Communication Review*, 38(5):17–28, Oct. 2008.
[32] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of MobiCom '00*, Aug. 2000.
[33] B. Nardelli, J. Lee, K. Lee, Y. Yi, S. Chong, E. W. Knightly, and M. Chiang. Experimental evaluation of optimal CSMA. In *Proceedings of INFOCOM '11*, Apr. 2011.
[34] J. Shi, O. Gurewitz, V. Mancuso, J. Camp, and E. W. Knightly. Measurement and modeling of the origins of starvation in congestion controlled mesh networks. In *Proceedings of INFOCOM '08*, Apr. 2008.
[35] I. Tinnirello, D. Giustiniano, L. Scalia, and G. Bianchi. On the side-effects of proprietary solutions for fading and interference mitigation in IEEE 802.11b/g outdoor links. *Computer Networks*, 53(2):141–152, Oct. 2009.
[36] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behaviour in 802.11 adhoc networks. In *Proceedings of ICC '00*, Jun. 2000.
[37] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. DiffQ: Practical differential backlog congestion control for wireless networks. In *Proceedings of INFOCOM '09*, Apr. 2009.
[38] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of EmNets '05*, May 2005.
[39] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of MobiCom '06*, Sep. 2006.
[40] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proceedings of MobiCom '03*, Sep. 2003.
[41] G. Yu, W. Wang, J. Yong, B. Leong, and W. T. Ooi. Adaptive antenna adjustment for 3D urban wireless mesh networks. In *Proceedings of SECON '13*, Jun. 2013.
[42] A. Zhou, M. Liu, Z. Li, and E. Dutkiewicz. Modeling and optimization of medium access in CSMA wireless networks with topology asymmetry. *IEEE Transactions on Mobile Computing*, 11:1559–1571, Aug. 2011.
[43] M. Zorzi and F. Borgonovo. Performance of capture-division packet access with slow shadowing and power control. *IEEE Transactions on Vehicular Technology*, 46(3):687–696, Aug. 1997.

Wei Wang is a Ph.D. candidate in Computer Science at the National University of Singapore. He received his M.Eng. and B.Eng. from Nanyang Technological University in 2008 and 2004, respectively. His research interests are on developing practical 802.11 wireless systems, including large-scale mesh networks and UAV-based networks.

Ben Leong is an Assistant Professor of Computer Science at the School of Computing, National University of Singapore. He received his Ph.D., M.Eng. and S.B. degrees from the Massachusetts Institute of Technology in 2006, 1997 and 1997 respectively. His research interests are in the areas of computer networking and distributed systems.

Wei Tsang Ooi received his B. Sc. (Hon.) degree from National University of Singapore in 1996, and Ph. D. in Computer Science from Cornell University in 2001. He spent a year as postdoc at Berkeley Multimedia Research Center in U.C. Berkeley, before re-joining NUS in 2002, where he is currently an Associate Professor in the Department of Computer Science. Wei Tsang's research focuses on interactive multimedia systems, including zoomable videos and networked graphics.