A Simple Rate-Control Fairness Algorithm for Wireless Mesh Networks

Szymon Jakubczak*, Alex T. K. Lau†, Lily Li‡, Paul A.S. Ward§
*Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: S.Jakubczak@elka.pw.edu.pl
†School of Computer Science, University of Waterloo,
200 University Ave. W., Waterloo Ontario, Canada N2L 3G1
Email: atklau@alumni.uwaterloo.ca
‡Department of Electrical and Computer Engineering, University of Waterloo,
200 University Ave. W., Waterloo Ontario, Canada N2L 3G1
Tel: 1-416-666-4016
Email: l7li@shoshin.uwaterloo.ca
§Department of Electrical and Computer Engineering, University of Waterloo,
200 University Ave. W., Waterloo Ontario, Canada N2L 3G1
Email: pasward@ccng.uwaterloo.ca

Abstract

The use of 802.11-based multi-hop wireless mesh networks as an alternative technology for last-mile broadband Internet access is growing rapidly. However, without a suitable fairness algorithm such networks exhibit extreme network-layer unfairness causing various nodes to starve or have extremely poor throughput.

Based on the hypothesis that if the fair bandwidth achievable by each node is known and enforced at each node, network-layer fairness can be achieved, we propose a simple three-step rate-control algorithm that requires no changes to the underlying 802.11 MAC.

In this paper we formally describe the algorithm, proving that it converges rapidly. We also provide simulation results showing that the algorithm achieves excellent fair sharing of the network while responding rapidly to changes in the network state. Finally, our simulations show that our algorithm often achieves more than 95% of the theoretical maximum fair-share aggregate throughput, while offering much more attractive fairness than current schemes.

Index Terms

Wireless mesh network, multihop wireless network, fairness, rate control, congestion control

A Simple Rate-Control Fairness Algorithm for Wireless Mesh Networks

***Abstract*— The use of 802.11-based multi-hop wireless mesh networks as an alternative technology for last-mile broadband Internet access is growing rapidly. The primary advantages of such an approach are ease of deployment and lower cost. However, such networks have a critical problem, *viz.* they exhibit extreme network-layer unfairness. Without a suitable fairness algorithm, various nodes can starve or have extremely poor throughput. The more distant a node is from its destination, the greater the likelihood of this problem occurring. This problem occurs even with the use of TCP, which attempts to provide fair sharing.**

Based on the hypothesis that if the fair bandwidth achievable by each node is known and enforced at each node, network-layer fairness can be achieved, we propose a simple rate-control algorithm. Our three-step method consists of distributed acquisition of network-state data, computation of the fair-share rate, and self-policing. The algorithm requires no changes to the underlying 802.11 MAC.

In this paper we formally describe the algorithm, proving that it converges rapidly in theory. We then simulate the algorithm, showing that it achieves excellent fair sharing of the network, and that it responds rapidly to changes in the network state. Finally, our simulations show that our algorithm provides this fairness at minimal cost, often achieving more than 95% of the theoretical maximum fair-share aggregate throughput, while offering much more attractive fairness than current schemes.

I. Introduction

Wireless mesh networks (WMNs) have received significant attention as an alternative technology for last-mile broadband Internet access [1]. Such networks belong to a particular subclass of wireless multi-hop ad-hoc networks. The distinctive properties of WMN are:

- Stationary: All nodes of a WMN are immobile. This implies that the topology of a WMN is mostly static. Topological changes are caused not by mobility but by incremental deployment and node failure. To avoid pre- and re-configuration, WMNs have to support self-configuration when the topology changes, though this should be relatively rare.
- Traffic pattern: WMNs consist of regular nodes that act as hosts and/or routers, and gateways that bridge between the WMN and a fixed network (typically the Internet). Unlike the general peer-to-peer paradigm of *ad hoc* networks, the traffic in WMNs is not normally between pairs of arbitrary nodes, but between hosts and their designated gateway.

Fairness is a critical property in computer networks [2]. However, current WMNs based on the IEEE 802.11 MAC and standard network-layer protocols cannot provide fairness to each user. In particular, it has been demonstrated that nodes close to the gateway can starve those that are more hops away [2]. Although significant research has been done to address fairness issues over MAC-layer flows within a single-hop, we are not aware of any solutions that efficiently provide fairness over network-layer flows in multi-hop networks.

Jun and Sichitiu [1] suggest that "once how much bandwidth each node can receive is known, this value can be enforced at each node," and network-layer fairness can be achieved. Inspired by their observation, we present a simple rate-control algorithm that can be used to achieve network-layer fairness while obtaining reasonable aggregate channel utilization. Our three-step algorithm is able to adjust flow rates dynamically as various nodes start or stop transmitting, while requiring no changes to the 802.11 MAC.

The remainder of this paper is organized as follows. In Section II we describe our three-step algorithm. We then formally prove it correct. In Section IV we present simulation results that demonstrate the efficacy of our algorithm. We discuss related work in Section V, comparing it with our approach. We conclude by observing what issues remain open.

II. Algorithm

In this section we formally describe our algorithm. We first state our assumptions, which allows us to develop a formal graph model of WMNs. We then state our algorithm in the context of this model.

We first assume that routing is effectively static, based on the fact that in WMNs nodes are stationary, and likely quite reliable. What we mean by "effectively static" is that changes in routing will be significantly less frequent than changes in stream activity. This assumption implies three things: node failure is rare, new node addition is rare, and load balancing is not being used. Within WMNs the first two implications are certainly true. The third is an undesirable restriction, but one that we hope to remove in the near future.

We further assume that the WMN has a single gateway. This is almost certainly not true, but, given static routing, for each node there will be a single gateway. We thus partition a multi-gateway WMN into disjoint WMNs, each of which has a single gateway. While there may be interference between the resulting set of WMNs, this is a problem that must already be dealt with insofar as there may be interference from any number of other sources.

Finally we assume that the routing is symmetric. As with the limitation to static routing, this is a restriction we expect to address soon.

Given these assumptions, we model the WMN as a graph in which the vertices are the wireless nodes and an edge exists between any two vertices iff the nodes they represent are within transmission range. The traffic pattern combined with symmetric routing implies that the subgraph of routes is a spanning tree.

We define a *link* as any edge in the spanning tree. Two links *contend* if they cannot be used simultaneously. We then define a *stream* as a directed path in the spanning tree over which data flows, and whose endpoints are the gateway and a regular node. It is relatively easy to see that all edges in a stream are links, and any edge that is not part of a stream is not used to transmit data. For the remainder of this paper we will consider the data that is being transmitted over the stream as being synonymous with the stream.

If a stream originates at the gateway, and terminates at a regular node, we term it a *downstream*. Conversely, if the stream originates in the WMN, and terminates in the gateway, we term it an *upstream*. It then follows that, if there are N regular nodes, there can be a maximum of $2N$ streams, N upstreams and N downstreams.

With these assumptions and this model in mind, we can now state the algorithm. It consists of three major steps:

1) gather information required to compute fair-share bandwidth
2) compute the fair-share rate for each stream
3) enforce the computed rate at stream origins

For the third step we choose a simple solution: the originating node places all packets belonging to one stream in a paced queue. It is a variation of a leaky bucket, but with an adjustable rate. The next packet is scheduled to be released after a delay of $lastPacketSize/lastRate$.

We now address the substantially more difficult first and second problems. This gives us both a method to calculate the fair-share bandwidth, and knowledge of what information must be distributed to the various nodes. Having that knowledge, we will present in Section II-B our distribution mechanism.

A. Computational Model

The algorithm relies on the computational model used to compute the fair bandwidth allocation for each stream. This is an optimization problem subject to the feasibility model for the network, the network state, and the fairness criterion adopted.

The feasibility model reflects the rate constraints imposed by the network. It consists of a set of linear constraints determined by how streams use the wireless medium, which can be determined by the routing (*i.e.* usage of links by streams) and link-contention (*i.e.* usage of medium by links). Because of location-dependent contention, the medium can be conceptually divided into resources of limited capacity. Thus we built a network resource model in the style of Kelly *et al.* [3].

Link contention can be captured by a link contention graph $G = (V, E)$, where V is the set of all links, and $\{u, v\} \in E$ iff links u and v contend. While various authors have proposed to divide the medium into resources according to cliques in the contention graph (*e.g.* [4]), we adopted the simpler model of collision-domains [1] to reduce node computation requirements. A collision-domain is associated with each link and contains all its contenders. We combined this collision-domains model with a variation of the two-hop link-contention model [1], [5]. In our model two links contend if one endpoint of one link is within transmission range of one endpoint of the other link. Note that since transmission range is often much less than interference range, this model underestimates the level of contention. However, since the collision-domain overestimates contention compared to the more accurate cliques, our combined model offers acceptable accuracy, with computational simplicity. As for the capacity of a collision domain, we adopt Jun and Sichitiu's definition of the nominal MAC-layer capacity B as the throughput that can be achieved at the MAC layer in a one-hop network with infrastructure [6].

We extend the feasibility model by the network state, which encompasses stream activity. Additional constraints requiring that no stream should receive more throughput than requested are included. We consider only binary activity state: the stream is either silent (requests zero) or never satisfied (requests infinity).

The fairness criterion implements the selected fairness model. In this paper we experiment with absolute fairness (*i.e.* all active streams receive same rates). However, any mathematically tractable criterion could be adopted. That said, as described below, we use one bit per stream to describe the network state, which restricts the amount of information about nodes' transmission intentions. Should more-dynamic state information be necessary, the encoding will need to be extended.

The resulting computational problem is to find the bandwidth allocation vector satisfying the adopted fairness criterion, subject to network feasibility and stream-activity constraints. The knowledge that must be obtained for this computation is, then, which nodes are transmitting at any given instance, and what is the network topology.

B. Network-State Distribution

Given our assumption that the topology and routing information of WMN are relatively static, this information can be propagated to the various nodes using any suitable information-dissemination protocol. Network state (*i.e.* the activity status of every possible stream), however, can change more rapidly. While there are a large number of ways in which this information can be gathered, we imposed various constraints on any acceptable solution. First, we did not want to change the MAC layer, since this would substantially limit the applicability of our solution. Second, we did not wish to use control packets, partly because this would represent an extra overhead to the solution, but also because we wanted the finer granularity of being able to react to every data packet transmitted. Third, we wanted a fully-distributed solution. More precisely, we did not want our solution to depend on the behaviour of some specific node, since this would likely reduce the ability of the system to react quickly, and could present a single point of failure. For the same reason, group decision making schemes were rejected. Rather, each node is required to collect data, independently compute its fair-share rate, and police itself. Based on these considerations, we propose a distribution algorithm that works on top of the existing IEEE 802.11 MAC by piggybacking the activity information in every data packet transmitted.

The activity information of all possible streams in the WMN is encoded in bitmaps. Each bit uniquely represents the activity status of one stream (*i.e.* active or inactive). The association of bits to streams is relatively static, since the topology is relatively static, and thus not the focus of this paper. The activity information of the whole WMN is then distributed with every data packet. Note that since the IEEE 802.11 MAC data frame allows 2300 bytes per packet, and the typical MTU for a network-layer packet is 1500 bytes or less, there is plenty of room to insert this activity information. Moreover, since only one bit is required per stream, this activity information is of negligible size (*e.g.* for N = 24 the bitmap uses 6 bytes) and not likely to significantly impact network performance.

The receiver of a data packet extracts the activity information from the packet and updates itself if the activity information in the packet is more up-to-date. Since a node can hear packets transmitted by its neighbours, we have it snoop these packets. This assures that even if there are no data packets transmitted from node m to its neighbouring node n, n can still get the more up-to-date information from m.

In order to decide whose information is more up-to-date we implement a version control rule which states that the node which is closer to n is more up-to-date on the activity of streams originating from n. The distance is measured in number of links on the spanning tree.

Algorithm 1 Piggybacking algorithm

```
Event localChange(stream, activity)
    info[stream] := activity

Event received(packet, neighbour)
    if not packet.info = info then
      for all stream do
        if nextHopTo(stream.origin) = neighbour then
          info[stream] := packet.info[stream]

Event silentUplinkNode()
    for all stream do
      if nextHopTo(stream.origin) = uplink node then
        info[stream] := active

Event silentLink(child)
    for all stream do
      if nextHopTo(stream.origin) = child then
        info[stream] := inactive
```

The pseudocode of our implementation is presented in Algorithm 1. At each node the array `info` stores the local knowledge of the network state. Each time it is updated, the fair-share rate of local streams is recomputed. The `received` event implements the version control rule. Note that the `neighbour` is hop-wise closer to the `node` iff `neighbour` is the next hop to `node` in the link tree.

1) Silent and Passive Subtrees: Piggybacking requires existing data transmissions. As a result, nodes that are within a subtree with no active traffic, will not be kept up-to-date. We refer to such a subtree as a *silent subtree*. However, this brings up the question as to what activity information a node in a silent subtree should use when it activates. In our solution we take the conservative approach of having such nodes assume full activity of by streams for which it has no up-to-date information. This is consistent with the TCP slow start approach, and should not affect the performance of our system when it is used in conjunction with TCP. The detection of a silent subtree is handled by the `silentUplinkNode` event shown in Algorithm 1. If the node's uplink (next hop to the gateway) neighbour becomes silent, the node is the root of a silent subtree, and the `info` on streams it does not relay (on which it cannot obtain up-to-date information) is adjusted.

We refer to a subtree that has only downstream traffic as a *passive subtree*. Unlike silent subtrees, passive subtrees will be kept up-to-date. That said, insofar as they do not originate data, it is not as critical to keep the nodes in a passive subtree up-to-date.

2) Deactivation Detection: New activation information can be naturally piggybacked by the packets of the activated stream, or other streams, and distributed throughout the network. The arrival of the first packet signals activation.

However, the detection of stream deactivation is trickier. It can only be signaled when inactivity (*i.e.* lack of packets) is detected by a node that is transmitting (either originating or relaying) packets. We set a time threshold for inactivity, which is adjusted to $k * packetSize/fairShare$. It is thus the estimated time required to observe k packets of the monitored stream. This k factor effectively controls the sensitivity of the algorithm with respect to stream activity. High sensitivity will result in weaker stability and possible false positives that could lead to unfairness. On the other hand, low sensitivity will result in wasted bandwidth, not utilized until the deactivation event is raised and distributed.

If the deactivating node is not within a passive subtree, the relayed upstream packets can piggyback the deactivation information. However, if a node remains within a passive subtree after deactivation, the parent of that passive subtree will notice the lack of upstream packets from its child triggering the `silentLink(child)` event. The deactivation of the stream can be thus detected and distributed in the active part of the network. We later refer to this mechanism as *delegated deactivation detection*.

III. Theoretical Analysis

In this section we demonstrate how the network-state distribution algorithm presented in the previous section converges when network state changes. We consider distribution by both direct data transmissions and packet snooping. We show what factors influence the speed of convergence. We conclude that our algorithm is sufficient to ensure fairness.

A. Proof of Correctness

Given a network with some nodes that have out-of-date `info`, we show that Algorithm 1 converges to a state where all active stream origins are fully up-to-date if no more activity change events occur. The version control rule effectively isolates the activity information of each stream, hence we only need to show that the distribution of activity information of a single stream converges.

We consider a discrete temporal model where packet transmission is instantaneous, but preceded by a non-zero delay. Let $r(u,v)$ denote the path (a sequence of nodes) in the link tree connecting node u to node v, and let $|r(u,v)|$ denote the number of hops on that path. We let n denote the origin of the stream activity change event. In the case of *delegated deactivation detection*, n is the parent of the passive subtree.

Theorem 1: It takes exactly $|r(n,p)|$ subsequent successful packet transmissions over subsequent links of the path $r(n,p)$ to bring p up-to-date with n.

Proof: We use induction on $|r(n,p)|$.

$|r(n,p)| = 0$ implies $p = n$, and n is already up-to-date.

In general, due to the version control rule, p always learns the information from its up-to-date neighbour $nextHopTo(n) = q$. Exactly one successful packet transmission between q and p is necessary. Since $|r(n,q)| < |r(n,p)|$ the induction hypothesis implies q has learned the up-to-date information in $|r(n,q)|$ transmissions. Finally, $|r(n,q)| + 1 = |r(n,p)|$. ■

It is important to note that, thanks to snooping, the transmission from q does not need to be addressed to p, provided that p can receive it. As a result the distribution of activity information is still possible even if there are no transmissions in the desired direction. For example, in the case when out-of-date nodes only originate upstream traffic, but have no downstream traffic directed to them.

We are therefore particularly interested in the possibility of successfully capturing a data packet when there is no downstream traffic to an active subtree. Let p denote its root, and q the parent of p. Suppose q is transmitting, then no child of p could transmit to p due to the two-hop contention. Furthermore, as there are no downstream transmissions in the whole subtree, the children of p are silent. We conclude that p is likely to succeed in capturing q's transmission, and will eventually do.

Theorem 1 shows that it takes a finite number of packet transmissions to bring any node p up-to-date with n. If we disregard the passive subtrees, and assume at least one downstream is active in the WMN, there is always a packet transmission pending at each node. Therefore, Algorithm 1 is guaranteed to converge. It follows, that if the model used to compute the optimal rates is accurate, the selected fairness model is achieved.

B. Time Delay

The time of convergence is the time necessary to propagate recent network state changes across the network and bring all nodes up-to-date. We consider the propagation time of an individual event. According to Theorem 1 the total delay of propagation over a path is the sum of the delays at each subsequent hop of the path. The convergence time will be determined by the slowest propagating path.

Since we abstracted from the details of the MAC protocol, exact values cannot be provided. However, the delay of a packet transmission from a node is on average inversely proportional to total (regardless of direction) output packet rate of that node. Hence nodes that relay little traffic are effectively the bottlenecks of the distribution. However, such nodes are actually parents of subtrees that contribute little traffic. It is acceptable that activity information propagates slowly through these low-activity subtrees. Silent-subtree pruning is the extreme case where zero activity results in no information propagation need. Although the delegated detection of deactivation does not require any transmissions, the *signal* of stream deactivation travels at the same speed as its packets according to the previous rate allocation.

It is important to note that any stream packet can participate in the distribution of the activity information. As a result, the information can travel significantly faster than the data in the streams. In particular, the higher stream activity, the smaller the fair-share rate. However, the number of packets being transmitted in the network will remain at the capacity of the network. The result is that the speed of network state distribution is relatively higher as more streams participate.

The impact of propagation delay is different in the cases of stream activation and deactivation. After an activation a new load is introduced into the network which was not accounted for in the computation of the current fair-share rate. A long propagation delay could thus lead to unfairness. In contrast, slow propagation of a deactivation event cannot disturb the fairness, but rather cause a loss of utilization.

IV. Experiments

We simulated our algorithm for both UDP and TCP traffic using the *ns-2* [7] simulator. No changes to its implementation of the 802.11 MAC protocol were made. For the spanning-tree routing, we use static shortest-path routing. The core of the implementation follows the description from Section II. We set $MacDataRate$ = 1 Mbps, B = 860 kbps, $packetSize$ = 1500 bytes.

For the UDP experiments, we use CBR-generate traffic over UDP transport. However, rather than use the leaky-bucket variant described previously, we control the rates of the CBR generators directly.

In our TCP experiments a stream consists of a single one-way FTP over TCP connection between the gateway and a regular node. In actual deployment, however, a stream should encompass all data flow between a pair of WMN nodes.

TCP stream is, by nature, bidirectional. However, if the actual "application" data flows only in one direction, there is a substantial asymmetry in the bandwidth consumptions of the two directed components. In our experiments we use a simple solution: we only control the rate at the data source, and "reserve" some bandwidth for the ACK packets. If the coordination overhead is equivalent to c bytes, then this reservation is obtained by scaling $B' = \frac{|data|+c}{|data|+|ack|+2c}B$. In our TCP experiments we set B' = 735 kbps.

Recall from the previous section, we set an inactivity threshold equivalent to k packets using the current fair-share rate. This threshold will be used when monitoring both relayed and self-originated traffic. We chose to be more conservative and used $k = 16$ to avoid false positives. These would be particularly undesirable in combination with TCP congestion control forcing the subject stream to back off. The trade-off is some waste of the bandwidth.

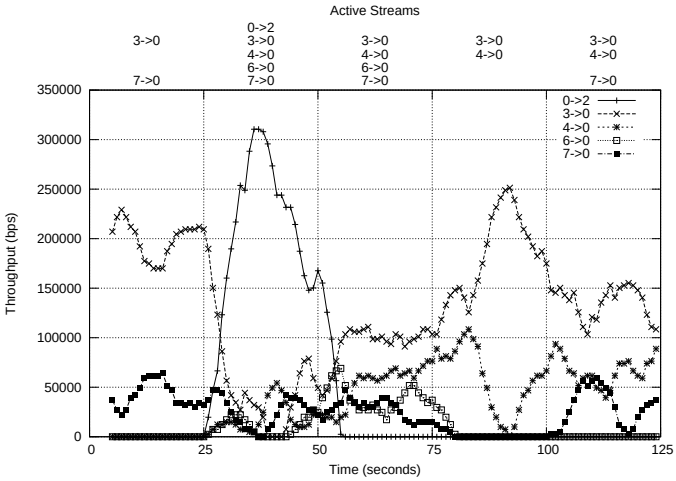

Fig. 1. Simulation results for chain topology with plain TCP

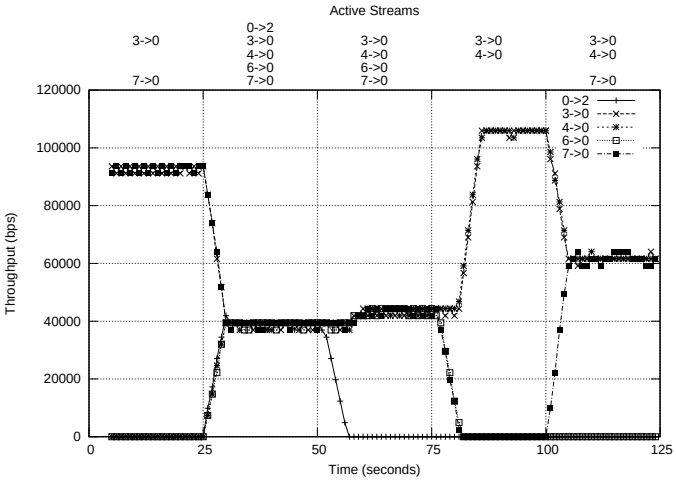

Fig. 2. Simulation results for chain topology (TCP) with fairness algorithm

In our experiments, we simulate stream dynamics by scheduling stream activity changes either manually or randomly. These events divide the entire time-line into time intervals of static activity.

For each interval we gather the following measures:

- sd/avg = standard deviation / average of throughputs (among active streams). This is a measure of how varied the received service is. Value 0 means perfect fairness. The higher the value, the more variation among the streams.
- min/avg = minimal throughput / average throughput. It illustrates the unfairness in the form of starvation of the most hindered streams. Value 0 means complete starvation. Value 1 means perfect fairness.
- avg/fs = average throughput / $fairShare$ according to the computational model. This measure evaluates overestimation of the computational model. The desired value is 1 provided the above fairness measures are satisfactory. Note that if the computational model is sufficiently accurate, it is infeasible to satisfy the fairness criterion when this measure exceeds 1.

We then compute the scenario average weighing each sample by interval length.

A. Chain Topology

In our simulations with a chain topology the transmission range is one hop and the interference range is two hops. The nodes are ordered by index: 0,1...,7 where 0 is the gateway. Each experiment is 125 seconds long and is divided into 5 equally lengthed intervals by manually scheduling stream activity changes. Fig. 1 and Fig. 2 show the throughput of each TCP stream without and with our fairness algorithm respectively. We measure throughput at each stream sink over each second, and average it over last 5 seconds.

We can see that the throughputs in Fig. 1 are very bursty. Furthermore, there is a big difference between the maximum and minimum throughput throughout the simulation. In fact, stream $4 \rightarrow 0$, $6 \rightarrow 0$ and $7 \rightarrow 0$ get almost completely starved at times, demonstrating the unfairness of the network as studied by many researchers [1], [2], [8].

The throughputs depicted in Fig. 2, on the other hand, shows very small variations ($sd/avg = 0.004$, $min/avg = 0.997$) across the whole simulation period, representing that fairness was achieved with our algorithm.

Our measurements also show that the aggregate throughput (i.e. bandwidth utilization) without fairness is higher than that with fairness. Various researchers [1], [9] have pointed out that fairness is in conflict with the bandwidth utilization in multi-hop wireless networks. For example, in a 2-Hop chain the aggregate throughput cannot be more than $\frac{2}{3}B$ if absolute fairness is achieved. However, the aggregate throughput can be maximized at B if we sacrifice fairness and let the farther node starve. In our experiments the aggregate throughput with fairness is 73% of that without fairness.

We now demonstrate how our distribution algorithm works when presented with activation and deactivation events.

If activation or deactivation occurs at a node where traffic is being relayed (When streams $0 \rightarrow 2$, $4 \rightarrow 0$ and $6 \rightarrow 0$ become active at time 25s, and when stream $0 \rightarrow 2$ becomes inactive at time 50s), the up-to-date activity information are easily distributed via the piggybacked packets.

The propagation delay of activation is less than 0.24s, which amounts to 1.83 packet according to the throughput at the time. Note that in Fig. 2 the convergence time seems to be always about 5s. This is because we measure the throughputs every second and average over 5 second. The plot resolution is not high enough to visualize the convergence.

The convergence time after the deactivation of stream $0 \rightarrow 2$ is much longer (6.23s). Note that this value includes the inactivity time threshold as described in the previous sections.

The more interesting scenario is when streams $7 \rightarrow 0$ and $6 \rightarrow 0$ are deactivated at time 75s. Node 7 and 6 then become part of the silent subtree 7-6-5. The deactivation will not be propagated until node 4 detects the silent subtree. Note that silent subtree detection does not take longer than normal deactivation detection as the whole network gets updated within 6 seconds.

In the last interval of Fig. 2 node 7 is active again. Since it is in a silent subtree it assumes that all the nodes in the upstream are active and starts slowly. It takes only about 0.34s for node

TABLE I
RANDOM SCENARIO FAIRNESS RESULTS

fairness	transport	dynamic	sd/avg	min/avg	avg/fs
-	TCP	No	1.30	0.07	1.60
		Yes	1.02	0.04	1.50
SRCF[a]	UDP	No	0.05	0.84	0.97
		Yes	0.10	0.81	0.91
	TCP	No	0.03	0.87	0.99
		Yes	0.08	0.82	0.85
RRFQ[b]	UDP	No	2.54	0.13	1.07
		Yes	1.71	0.13	0.87
	TCP	No	0.34	0.33	1.17
		Yes	0.39	0.22	1.20

[a]the proposed Simple Rate-Control Fairness
[b]round-robin fair-queing

TABLE II
RANDOM SCENARIO DISTRIBUTION DELAY RESULTS

transport	$AVG(p_a)$	$SD(p_a)$	$AVG(p_d)$	$SD(p_d)$
UDP	1.2	1.0	19.9	5.6
TCP	0.8	1.4	29.5	13.4

7 to get completely updated about the whole network status and speed up to the fair bandwidth.

B. Random scenarios

Each measure is obtained by averaging across 200 scenarios. Each scenario consists of a random 15-node topology of average diameter of 7.5 hops. There are 14 upstreams and 14 downstreams running for 1000.0 seconds. We simulate dynamics by independent random stream activity. Average interval length between any activity changes is *ca.* 5.5 seconds, and there are on average *ca.* 14.7 streams active at any time. Without dynamics there is just a single interval with all streams active.

The results are presented in Table I. Experiments without dynamics evaluate the validity of the computational model and the concept of source rate control as a whole. It seems that it can deal well with various topologies and stream configurations. The streams receive similar service and the minimal service is well above 80% of average. Also in the static case TCP cooperates well with proposed rate control. High accuracy indicates negligible overestimation. With stream dynamics the results slightly degrade, but it should be noted that when activation is concerned the subject stream starts from zero throughput. Combined with congestion control mechanisms of TCP (*e.g.* slow start) this affects the total service that the activating stream can receive in the short measurement interval.

The results for plain TCP (*i.e.* without fair rate control) display severe unfairness ($min < 0.1avg$) but also increased utilization (+50% of average thruput). We also compare the aggregate throughput for each scenario with and without fair rate control. The average ratio is 63% and 58% for static and dynamic cases respectively. We conclude that the proposed fairness algorithm offers significant improvement in terms of fairness at the price of total TCP throughput reduced by *ca.* 40%.

We have also performed experiments with round-robin fair-queuing as proposed in [2] for reference comparison. The results in Table I indicate that fair-queing is not sufficient to attain the requested level of fairness and place FQ between plain TCP and our solution in terms of both fairness and utilization. The reason for that is the inherent unfairness of the 802.11 MAC.

For dynamic scenarios we compute a normalized measure of propagation delay: $p_a, p_d = \frac{propagationDelay * fairShare}{packetSize}$ averaged for intervals started by activation (a) and deactivation (d) separately. It should be interpreted as the number of packets that could be sent during the propagation time using the fair-share. The results are gathered in Table II. Since the deactivation threshold is adjusted to the fair-share as described in Section II-B.2, p_d includes k in contrast to p_a. This explains the difference between those two measures.

According to the analysis in Section III-B, these normalized measures increase with the network diameter but decrease with the number of active streams. It is thus presumed that they will decrease with WMN size. From the results we conclude that the convergence time is small, and the propagation speed is faster than it would be possible with typical feedback mechanisms of congestion control.

V. REVIEW OF PREVIOUS WORK

Fair bandwidth allocation in wireless ad hoc networks has been given attention in many previous works. However, most of them address the problem at the level of the link-layer [4], [10]–[16], i.e. where all relevant flows span single hop only. Only few works addressed fairness issues at the network-layer of wireless ad hoc networks.

Woo and Culler [8] adapt the p-persistence MAC scheme proposed in [4] aiming for proportional fairness. The scheme is tailored for the multi-hop setting by separating the originated stream from the relayed streams into two outbound flows managed independently by the p-persistence scheme. Additionally, through manipulation of parameters they are virtually assigned different weights. These are basically proportional to the number of relayed streams. The resulting allocation does not conform to any well-defined model of fairness.

Jun and Sichitiu [2] explore the relationship between link-layer fairness guarantees and network-layer fairness. Specifically, they investigate the performance of several fair queuing schemes employed in the routers of a WMN. The goal set is absolute (or weighted) fairness. Their results *inter alia* rationalize the weighing applied in [8]. Furthermore, they indicate that in order to achieve fairness a queue per each relayed stream needs to be maintained. Additionally, it is shown that the achieved throughput could be increased with QoS-like MAC guarantees. However, they base on the assumption that the MAC in use is fair. Given our experimental results with FQ, we conclude it is not sufficient to provide end-to-end fairness in an 802.11-based WMN.

Tirthapura and Velayutham propose a distributed rate-control algorithm to achieve a new model of network-layer

fairness [5]. The model resembles max-min fairness in that less constrained streams can receive higher rates. The algorithm is based on dynamic estimation of the flow-contention level and allocates each flow a rate inversely proportional to the estimate. The whole stream receives a rate equal to the most hindered among its flows. A vote-like algorithm to forward this minimum rate candidate to the origin is proposed.

It is important to note that the speed of information distribution in our solution can be substantially faster, because the stream activity is not inferred from its presence in the neighbourhood, but explicitly distributed in the network. Additionally, our algorithm does not restrict the fairness model.

VI. Conclusion and Future Work

In this paper, we have presented a simple rate-control algorithm to address the fairness problem of network-layer streams in wireless mesh networks. Our algorithm consists of three components:

1) A distributed algorithm for the delivery of stream-activity information. The data is piggybacked on data frames, and thus requires no changes to underlying 802.11 MAC. We have given a proof of correctness of this algorithm and also shown that it will converge within d packet transmissions, and frequently much less, where d is the diameter of the network.
2) A computational task, performed at each node, for determining the fair-share rate for each stream.
3) A self-policing algorithm, that limits output rate to the computed fair-share rate.

We have studied our algorithm through simulations over various topologies. The simulation results show that our algorithm provides network-layer fairness with both UDP and TCP traffic at minimum cost. We observe significant improvement in fairness when compared with plain TCP, and TCP backed by fair-queuing. In the simulations with UDP our algorithm achieves above 95% of the computed fair-share.

We have observed that if the fair share is overestimated due to the inaccurate computational model, the fairness will be destroyed and the packets loss rate will be increased. We plan to deal with this issue by adapting the fair share rate to the packets loss.

The piggybacking scheme proposed here is simple and reliable but is still subject to improvements reducing the messaging overhead. The impact of the time threshold for deactivation detection as discussed in Section II-B.2 needs to be further investigated.

In our system model we assume static routing and with a single gateway, which may not always be true in real-world WMN environments. In case dynamic routing is adopted with multiple gateways used, load-balancing algorithms are needed to better utilize the network resources. This complicates both the fairness model, which will interact with any load-balancing scheme, and the distribution of network state information. We are currently investigating these problems.

References

[1] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE Wireless Communications*, pp. 8–14, October 2003.

[2] ——, "Fairness and QoS in multihop wireless networks," in *Proc. of the IEEE Vehicular Technology Conference (VTC)*, October 2003.

[3] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[4] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC layer fairness in wireless packet networks," in *Proc. of The 6th Annual International Conference on Mobile Computing and Networking*, 2000, pp. 87–98.

[5] S. Tirthapura and A. Velayutham, "Fairness and throughput optimization of end to end streams in multihop ad hoc wireless networks."

[6] J. Jun, P. Peddabachagari, and M. L. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," in *Proc. of the Second IEEE International Symposium on Network Computing and Applications*, April 2003, pp. 249–256.

[7] "The network simulator - ns-2," available at http://www.isi.edu/-nsnam/ns/.

[8] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proc. of the 7th Annual Internation Confererence on Mobile Computing and Networking*, 2001, pp. 221–235.

[9] A. K. Somani and J. Zhou, "Achieving fairness in distributed scheduling in wireless ad-hoc networks," in *Conference Proceedings of the 2003 IEEE International. Performance, Computing, and Communications Conference*, April 2003, pp. 95–102.

[10] M. Ergen, "IEEE 802.11 tutorial," June 2002.

[11] V. Bharhgavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol wireless LANs," in *Proc. of SIGCOMM*, 1994, pp. 249–256.

[12] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access (FAMA) for packet-radio networks," in *Proc. of SIGCOMM*, 1995, pp. 262–273.

[13] Y. Wang and J. J. Garcia-Luna-Aceves, "Channel sharing of competing fbws in ad hoc networks," in *Proc. of The IEEE Symposium on Computers and Communications (ISCC '03)*, July 2003, pp. 87–98.

[14] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks," in *Proc. of INFOCOM (3)*, vol. 10, June 2002, pp. 320–328.

[15] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *Proc. of The 6th Annual Internation Conference on Mobile Computing and Networking*, 2000, pp. 76–86.

[16] H. Luo and S. Lu, "A topology-independent fair queueing model in ad hoc wireless networks," in *Proc. of the International Conference on Network Protocols*, November 2000, pp. 325–335.