

# FairCast: Fair Multi-Media Streaming in Ad Hoc Networks through Local Congestion Control

Gustavo Marfia\*, Paolo Lutterotti<sup>‡</sup>, Stephan J. Eidenbenz<sup>†</sup>, Giovanni Pau\*, Mario Gerla\*

\*Computer Science Department - University of California, Los Angeles, CA 90095, US

e-mail: {gmarfia|gpau|gerla}@cs.ucla.edu

<sup>†</sup>CCN5 - Los Alamos National Laboratories - Los Alamos, NM, US

e-mail: eidenben@lanl.gov

<sup>‡</sup>Istituto Superiore Mario Boella, Torino, 10138, Italy

e-mail: lutterotti@ismb.it

*Abstract*—Multicast streaming is gaining increasing importance in wireless ad hoc networks, in part because ad hoc scenarios often include team activities and the requirement for distribution of audio, video and situation awareness to the members. At the network level, techniques for routing the multimedia streams are quite mature. Much more challenging is the allocation of resources, the fair sharing among streams and the control of congestion.

While in rate adaptive UNICAST streams congestion control and fair sharing are accomplished with end-to-end feedback techniques inspired to TCP, the feedback does not scale well in MULTICAST. In fact, it leads to the well known ACK/NAK “implosion” problem and unfair penalties for heterogeneous receivers.

These limitations can be overcome using backpressure from congestion points to the sources - but this approach suffers of latency and cannot rapidly adjust to changes in traffic. Another solution is multilayer adaptive coding. Namely, the encoding adaptation is done locally by dropping layers. It does not require end-to-end feedback nor changes in input rates. Multi-resolution codes are now becoming attractive due to the progress in technology; we expect these to become the prevalent techniques in large scale media distribution. One issue, however, that still remains to be locally resolved is the fair sharing among competing multicast streams.

In this paper we address the congestion control AND fair sharing in a multilayer multicast scenario. We show that lack of proper fairness provisions in the “local adjustments” can lead to serious “capture” situations, especially in heterogeneous traffic mixes (e.g. voice and video). We

then propose a FAIR local adjustment that targets a fair dropping of packets in each interference domain. We show that the scheme can be interpreted as a distributed implementation of a utility function minimization, where the utility is the packet loss subject to fairness bounds across flows. This formulation guarantees stability and convergence of the distributed algorithm.

The main contributions of this paper are the low overhead design of the local fairness enforcement algorithm, the utility function framework and the demonstration of convergence via simulation in representative scenarios.

## I. INTRODUCTION

We here propose a novel approach to multicast congestion control and fairness, based on local flow interaction. Differently from traditional models, where a flow is controlled by end-to-end feedback signals (e.g. packet loss and delay), we build an in-network, distributed flow interaction mechanism where no explicit control packets are delivered from destinations to sources. Competing flows locally interact, exchanging information on loss rates and adapting to each other’s performance. This approach is appropriate for ad hoc networks where the cost of sending end-to-end control packets is expensive. It is also more responsive to highly dynamic traffic and topology changes. In this study we focus on fairness. In particular we examine the fairness problem for real-time multimedia multicast flows. The fairness problem was thoroughly studied in the case of unicast best effort (e.g. TCP) and real time sources for wireless ad hoc networks. However, to our knowledge, little progress was done so far in efficient and fair multimedia multicast.

Ad hoc networks have been proposed for a variety of applications from battlefield missions to disaster recovery and vehicular communications. In any of these cases, streaming-based applications play an important role. In a disaster recovery application, for example, team leaders

*Reference Author: Gustavo Marfia, Computer Science Department, University of California Los Angeles, CA 90095, e-mail: gmarfia@cs.ucla.edu* This work is partially supported by the Italian Ministry for Research via the ICTP/E-Grid Initiative and the Interlink Initiative, the National Science Foundation through grants Swarms and Whynet, and the UC-Micro Grant MICRO 04-05 private sponsor STMicroelectronics. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

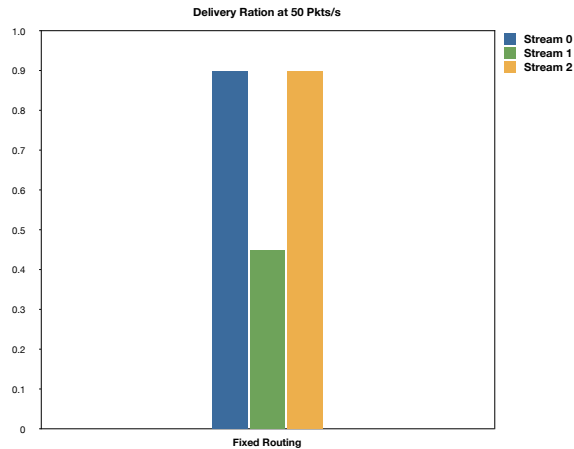


Fig. 1. *Unfairness between three flows.*

will need to multicast orders by voice to coordinate rescue teams, while the members of a rescue team will want to multicast a situation update of the disaster scene. Voice over IP (VoIP) and peer-to-peer streaming will also become popular in VANETs and metropolitan mesh networks. In brief, we can expect that the multi-hop wireless networks of the future, both fixed and mobile, will carry a great deal of voice/video streaming. For an efficient utilization of such networks, in terms of the number of streaming flows they are able to serve, rather than the aggregate throughput they deliver, a fair sharing of resources between flows is required.

A simple case of unfairness due to spatial contention is illustrated in Fig. 1. We plot the delivery ratio for three flows, on a  $12 \times 3$  grid topology. Flow  $i$  originates from node  $(1, i)$  and terminates at node  $(12, i)$ , so that each packet travels at least 11 hops. The delivery ratio of three competing CBR flows is plotted for three types of routing. The ideal routing solution, represented by parallel fixed routes, penalizes the goodput of the central stream, although improving overall utilization, as there is no routing overhead and inefficiency. If these were video streams capable to tolerate up to 30% loss thanks to a robust coding technique, 90% delivery ratio for flows 1 and 3 and 45% delivery ratio for flow 2 signifies that only flows 1 and 3 will be adequately served. Now, if flows 1 and 3 were reduced by 20%, say, thus freeing enough capacity for flow 2 to increase by 25% its delivery ratio, we would on one hand loose in aggregate utilization while on the other increase “flow utilization” by 50% (i.e. three flows instead of two would be adequately served). The unfairness problem is even more severe when considering multiple multicast flows. In such case

a multicast stream can choke others.

To underline the problem we show a case of unfairness between two multicast groups routed by the On Demand Multicast Routing Protocol (ODMRP) [13], a multicast protocol which forwards packets through a mesh structure, thus providing multiple routes from a source to each destination. ODMRP was chosen as it is popular and readily available in simulation platforms - the same behavior is expected of any ad hoc multicast scheme.

We examine how two multicast groups coexist by first running one multicast group by itself and then adding the second. We here show the results for the two scenarios. The bit rates of the two sources of the two multicast groups are chosen to emulate a low quality video multicast and an audio multicast. The MAC layer we use throughout the paper is 802.11b, with rate fixed at 1Mbps, thus disabling auto rate fallback. In the first test only the video multicast group is active. In the second experiment both, video and audio, are active. The streams are supposed to be layer encoded. For simplicity, in the following we will assume a very fine layer grain (in the limit, an infinite number of layers, each with infinitesimal rate), such that we do not have to be concerned about the discrete nature of the adaptive multilayer control. Moreover, we assume that within each flow the packets are dropped by layer priority. The network is composed of thirty static nodes, the terrain is a 1000 x 1000 meters square and nodes are displayed using Qualnet’s uniform placement model (i.e. the terrain is divided in as many cells as the nodes are and, within each cell, a node is randomly placed). Nodes are approximately placed as in Fig. 2. Both multicast groups have a single source, while nine receiving nodes are in the first group and three in the second. Moreover, no receiver or source in one group is a receiver or a source in the other group. Simulations last 560 second. The source of the first multicast group sends packets at a rate of 50pkts/sec or 200kbps, while the source of the second multicast group sends packets at a rate of 10pkts/sec or 40kbps. The figure of merit is the delivery ratio that is directly related to the received video quality. Results are averaged over the 200 simulations that are run per each set and show the overall delivery ratio per each receiver. Results for the first set of simulations, where only multicast group one is active, are shown in Fig. 3. Results for the second set of simulations are shown in Fig. 4. The delivery ratio of multicast group one doesn’t significantly change between experiments. Multicast group two is heavily penalized in the second set of simulations. The members of multicast group two, mainly due to the contention with multicast group one, loose most of the packets along the way. At this point the question is whether a more fair distribution

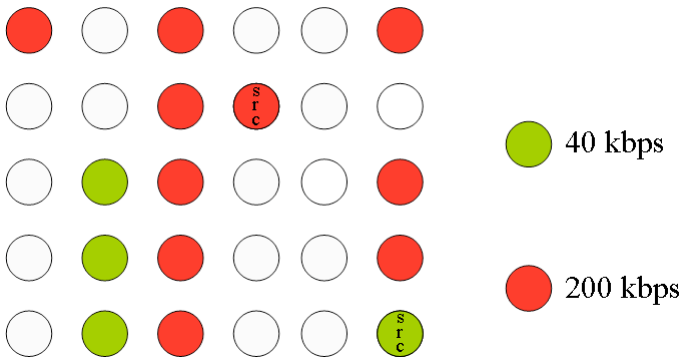


Fig. 2. Multicast example topology.

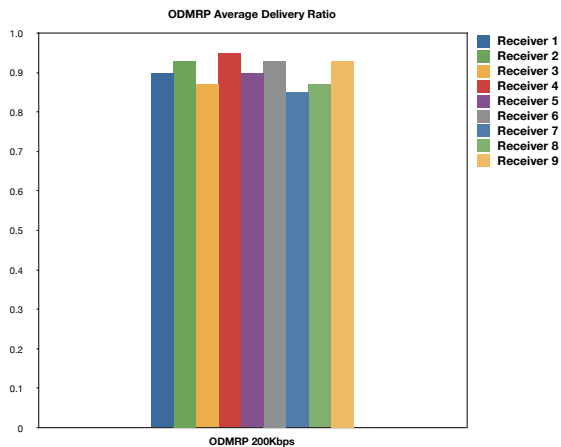


Fig. 3. Single video (i.e. 200kbps) multicast stream case. All receivers receive approximately the same delivery ratio, no relevant unfairness is observed.

of resources could be guaranteed. The answer is probably affirmative, since we see that between experiment one and experiment two multicast group one's delivery ratio is almost unaffected. Almost all members of multicast group one suffer at most 20% loss. If multicast group one receivers were willing to take a 30% rate reduction, there would likely be enough space for multicast group two as well, thus achieving a more fair access to the wireless resource.

The problem of ensuring fairness has been long studied in wired networks, especially in the case of elastic TCP sources. TCP strives to maximize the sum of source utility functions and the fairness characteristics of a flow can be directly drawn from the utility function it implements. These results are not easily mapped to contention-based wireless networks, due to a major complexity of the network model. Unfairness between competing TCP flows is aggravated, in multi-hop wireless networks, by the channel “capture” effect. Nodes, with a different

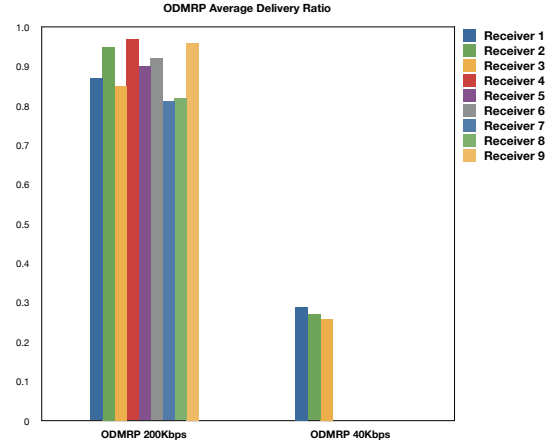


Fig. 4. Two multicast streams, a video (i.e. 200kbps) multicast stream and an audio (i.e. 40kbps) multicast stream are sent from two different sources. We can observe an inter-multicast flows unfairness.

number of interfering neighbors, will receive a different share of resources. Flows, with no nodes in common along their paths, may still be competing for resources due to spatial contention. This leads some flows to capture most of the channel's bandwidth and others to fall into starvation. A number of solutions have been proposed to balance this effect for TCP sources [5], [6]. Authors in [5] extend the Random Early Drop (RED) [7] concept to ad hoc networks, by observing that the algorithm should be enforced on the distributed neighborhood queue rather than on a single node's queue. FairCast is a mechanism which balances resource-sharing among flows. In this spirit, it is similar to the approach outlined in the NRED algorithm (though in our case we deal with drop rates rather than queue lengths). A new TCP source algorithm, designed to fairly work on wireless multi-hop networks is proposed in [6].

A second stream of work looks at fairness between neighboring nodes [8]–[12]. The main approach in the proposed solutions is to design new MAC layer protocols or to modify the existing de-facto standard, 802.11a/b/g, by modifying the scheduling and/or the backoff algorithm.

The main contributions of this work are: (a) to enforce overall network fairness on adaptive rate multimedia flows by implementing a local, distributed algorithm, and; (b) to introduce a new feedback paradigm, based on flow interaction, which is meant to add onto and not to substitute network feedback as packet loss and delay.

## II. ALGORITHM

The main assumptions in designing the algorithm are that flows compete for network resources and that flows can tolerate some degree of loss. The loss that a flow can tolerate depends both on the application and the encoding overlaid on it (e.g. in our case the number of layers required for acceptable reception). This loss rate is upper-bounded by a threshold that corresponds to the minimum acceptable quality. In case a flow experiences a loss rate that exceeds the desired threshold because of other flows aggressiveness, it promptly reacts against the competing flows requesting them to drop at a certain rate. This is the Distributed Gentlemen Agreement (DGA) implemented in FairCast. This means that if two flows compete and achieve different delivery ratios, the most penalized flow complains with the other and they both agree to a local fair share. To relate this to the traditional wired network congestion control framework, the shared wireless channel is here interpreted as a drop-tail router; flow reaction may be interpreted as the price of exceeding the capacity of such router (i.e. packet loss).

We may find the following advantages in such approach: (a) no bandwidth is wasted due to end-to-end feedback; (b) flows are faster in adapting to highly dynamic traffic changes (i.e. flows adapt to congestion and to traffic/topology changes on the fly, avoiding any end-to-end delay between source and receiver); and (c) it is easy to implement (i.e. no modifications are required below the network layer). We also find an equal number of disadvantages: (a) local decisions on limited information imply this is a suboptimal solution; (b) a packet which is voluntarily dropped at some node to be “nice” to a competing flow wastes the bandwidth used on the path traveled so far; and (c) some overhead is added to data packets (FairCast flows locally exchange congestion signals with data packets, but this overhead may also include, depending on the application, extra coding to make the flow robust up to a desired loss threshold).

We divide the algorithm implementation in two successive steps. The first step assumes each node knows the average packet loss rate threshold per each flow. By knowing this information each node will try to keep the packet loss rate below this value and will not react if this is, on average, bounded by this value. The second step is to understand how to set the threshold. We will better understand how these algorithms are derived in the Sections III and IV, where we derive the DGA algorithm and discuss the convergence properties of the threshold algorithm.

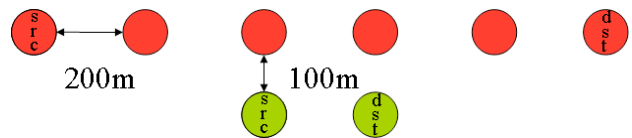


Fig. 5. Topology for the asymmetric unicast evaluation of FairCast. A 5-hop flow competes with a 1-hop flow.

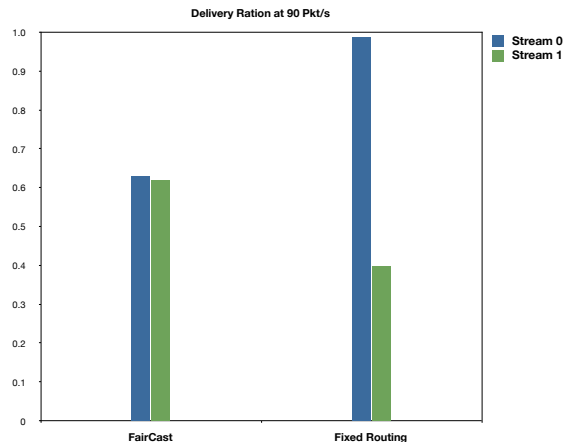


Fig. 6. Unfairness between the two flows in Fig. 5.

### A. Distributed Gentlemen Agreement

With FairCast flows locally agree on how to share bandwidth by exchanging packet loss rate information. In each interval a flow measures the amount of packets it received and the amount of packets it was able to send. This way a flow, on a per node basis, is able to determine its local packet drop rate. If the local packet drop rate, say, exceeds the threshold of flow  $r$  on node  $n$  for a certain time, flow  $r$  will ask the flows that have an impact on its own performance, in particular, the flows traveling with it and the flows traversing the neighbors of node  $n$  to lower their rates.

As a simple illustration of this concept we consider the topology shown in Fig. 5. We use unicast flows for simplicity. There is a long and a short flow. Without fairness control, the short stream (stream #1) wins. FairCast forces the flows to share the channel fairly as shown in Fig. 6.

### B. Algorithms

In each time interval  $\Delta t$ , which in the following formula is normalized to 1 for simplicity, each node calculates, per each flow, the following value:

$$\lambda_{r,n}(t+1) = [\lambda_{r,n}(t) + \gamma(p_{loss,n}^r(t+1) - thr_{loss,n}^r(t+1))]^+ \quad (1)$$

where:

- $\lambda_{r,n}(t)$  is the drop probability set by flow  $r$ , at node  $n$ , for its competing flows
- $\gamma$ , a constant, is the step-size of the gradient algorithm
- $p_{loss,n}^r(t)$  flow  $r$ 's packet loss rate
- $thr_{loss,n}^r(t)$  the threshold above which flow  $r$  reacts to packet loss

Flow  $r$  may exceed the maximum tolerated packet loss rate  $thr_{loss,n}^r(t)$ , which we assume to be known at each node, per each flow. Periodically flow  $r$  calculates  $\lambda$ , a measure of how much flow  $r$  is loosing with respect to threshold  $thr_{loss,n}^r(t)$ , implementing Algorithm 1:

---

#### Algorithm 1 Lambda Algorithm

---

**Ensure:** Each flow  $r$ , within node  $n$ , reacts to packet loss by computing  $\lambda_{r,n}$

$r.d[k]$  is number of packets which node  $n$  failed to forward in the  $k$ -th interval. This value includes both the packets that have not been acknowledged and packets that have been dropped by FairCast.

$r.routed[k]$  is the number of packets that node  $n$  received to forward in the  $k$ -th interval.

$\alpha$  is the exponential filter constant.

$\gamma$  is the gradient-algorithm's step-size.

**for** each flow  $r$  in  $n$  **do**

**if**  $r.d[k] > 0$  **then**

$$d\_rate[k] = r.d[k]/r.routed[k]$$

$$r.d\_rate[k] = (1 - \alpha)d\_rate[k] + \alpha r.d\_drop\_rate[k - 1]$$

$$\lambda_{r,n}[k] = \lambda_{r,n}[k - 1] + \gamma * (r.d\_rate[k] - r.thr[k])$$

**end if**

**end for**

---

Once  $\lambda_{r,n}$  is computed, this is then inserted in any outgoing packet of flow  $r$ . We avoid the overhead of sending a new control packet at the price of wasting some space on each data packet. What then happens is that some node  $m$ , interfering with node  $n$ , receives this packet. If the lambda node  $m$  receives is greater than the last received lambda or if the last received lambda expired, node  $m$  saves this value. It then runs the algorithm we show in Algorithm 2 every time it forwards a packet.

We can summarize what happens with Algorithm 2 as follows: before sending a packet of a flow a node compares the local lambda of this flow to both the flow's threshold and the last valid lambda received at this node. If the local flow's lambda is smaller than both these values, it means that the flow may tolerate losing some packets. In such case, the node will lower the flow's

---

#### Algorithm 2 Receive Lambda Algorithm

---

**Ensure:** Node  $m$ , a neighbor of node  $n$ , receives  $\lambda_{r,n}$ . Node  $m$  reacts with a local drop, if certain conditions are met.

**for** next packet that is going to be sent **do**

  next packet to be sent is  $pkt_s$  of flow  $s$

$\lambda_{r,n}$  = Last received lambda

**if**  $(\lambda_{s,n} < \lambda_{r,n}) \&\& (\lambda_{s,n} < s.thr) \&\& (r \neq s)$

**then**

$prob\_drop = uniform\_random[0, 1]$

**if**  $prob\_drop > 1 - \lambda_{r,n}$  **then**

$dropped\_packets = dropped\_packets + 1$

      DropPacket( $pkt_s$ )

**else**

**if**  $\lambda_{r,n} > max\_local\_lambda$  **then**

        set forward lambda =  $\lambda_{r,n}$

**else**

        set forward lambda =  $max\_local\_lambda$

**end if**

**end if**

**end if**

**end for**

---

access rate to the wireless medium (thus increasing its own loss rate), leaving some space for the competing node's flow that is suffering from contention. If the condition is not met with any flow, it is not possible to redistribute flow loss by penalizing a local flow and the originator of contention (to be reduced) should be found elsewhere. For this reason, the node still checks if the received lambda is higher than the locally calculated lambda. If this condition is met, the node will forward the received lambda instead of its own. In this way, a lambda can be propagated two hops from each node and it is likely that the congestion origin can be reached.

### III. GENTLEMEN AGREEMENT MODEL

We here define an optimization model that will lead us to define the FairCast algorithm and protocol. The result of this section will be the following result: we will derive "flow interaction" rules and "selective drops" from a utility function optimization which will lead to an overall minimum packet loss and a bounded maximum packet loss per flow on each hop. We will interpret the resulting algorithms in the wireless ad hoc domain, showing how flows can adjust their rates to mitigate the unfairness due to spatial contention between nodes. In the following for simplicity of notion we formulate the problem considering a single multicast communication. Multiple unicast and multiple multicast models are straightforward extensions or reductions of this model.

From now on we will use the following notation:

- $L$  is the set of links of the network,  $N$  is the set of nodes and  $R_i$  is the set of receivers for source  $i$ .
- $p_{loss}^r$  is a vector of  $|L|$  rows, for receiver  $r$ . Row  $p_{loss,l}^r$  represents the average fraction of packets lost, for receiver  $r$ , on link  $l$ .
- $thr_{loss}^r$  is a vector of  $|L|$  rows, for receiver  $r$ . Row  $l$  represents the maximum fraction of packets lost that receiver  $r$ , that can be tolerated on link  $l$  before “complaining”.
- $p_{loss,0}^r$  and  $x_0^r$  are scalars that both depend from the data rate and the packet size of flow  $r$ .
- $x_r$  is a vector of  $|L|$  rows, for receiver  $r$ . Row  $x_l^r$  represents the average rate of flow  $r$  on link  $l$ .
- $x_{source}$  is a scalar. This is the maximum rate at the source.
- $\theta_0^r$  is a constant. This is a constant per each flow and relates the average loss rate to the average rate received by a flow.

#### A. Optimization Problem

We define our problem as follows:

$$\min \sum_r \mathbf{1}^T p_{loss}^r \quad (2)$$

subject to:

$$p_{loss}^r \leq thr_{loss}^r, \quad \forall r \in R_i \quad (3)$$

$$p_{loss,l}^r \geq p_{loss,0}^r - \theta_0^r(x_l^r - x_0^r), \quad \forall l \in L, \quad \forall r \in R_i \quad (4)$$

$$x_r \geq 0, \quad \forall r \in R_i \quad (5)$$

$$\sum_r x_r \leq |R_i| x_{source} \mathbf{1} \quad (6)$$

where vectors  $p_{loss}^r = p_{loss}^{r,c} + p_{loss}^{r,d}$ ,  $p_{loss}^{r,c}$  represents the fraction of packets lost due to contention,  $p_{loss}^{r,d}$  represents the fraction of packets lost due to FairCast drop decisions. Clearly,  $p_{loss}^{r,d}$  are the variables in the optimization problem. Please notice one important fact, an equation of type  $x_l^r = F(x, p)$  is missing. Such equation models how a rate of flow  $r$  over link  $l$  depends from the rates and loss rates of other flows on competing links. The definition of such equation is out of the scope of this work, such equation depends on a number of factors and a closed form for such equation is still an open research problem. We will here assume  $x$  and  $p_{loss}^c$  are observable at each link. This assumption is

consistent in the measure that we simulate the network using Qualnet and feed the observed  $x$  and  $p_{loss}^c$  vectors into our optimization problem. We are then here defining a Simulation-Optimization problem, a generalization of a Deterministic Optimization problem, where one or more of the constraints is observable through a stochastic simulation.

The objective, expressed by eq. (2), is to minimize the fraction of packets lost in the network. Constraint (3) fixes a packet loss upper bound at links. Constraint (4) linearly models the relation between packet loss and rate at each link, approximating it in a neighborhood of steady state. Eqs. (5) and (6) are feasibility constraints, where eq. (6) limits the total rate at receivers to be bound by the rate at source by the number of receivers.

The rationale behind eq. (3) is that in a resource constrained environment such as an ad hoc network an information loss decision should be distributed. An end-to-end formulation would bring to an optimal solution of the problem, but would also introduce overhead, resource consumption and feedback delay. This would lead to an infeasible practical resolution of the problem. We assume eq. (4) to hold in an interval of the average bandwidth required to send a flow at each link. Bandwidth may heavily oscillate in an ad hoc network, we will therefore see how this assumption holds from our simulation results. In general, there may be heavy fluctuations around the average bandwidth in contention based wireless networks.

We simplify the problem by merging (4) and (6) and relaxing the constraints. The new set of constraints is:

$$p_{loss}^r \leq thr_{loss}^r, \quad \forall r \in R_i \quad (7)$$

$$\sum_r \frac{1}{\theta_0^r} (p_{loss,0}^r - p_{loss}^r - \theta_0^r x_0^r \mathbf{1}) \leq |R_i| x_{source} \mathbf{1} \quad (8)$$

$$0 \leq p_{loss}^r \leq p_{loss,0}^r \quad (9)$$

We now decouple the problem in terms of flows, deriving  $|R_i|$  problems. Following the approaches discussed in [15], by relaxation, we define one sub-problem per each receiver and a single master problem to coordinate them. The  $|R_i|$  sub-problems are:

$$\min_{p_{loss}^r} \mathbf{1}^T p_{loss}^r + \mu^T \frac{1}{\theta_0^r} (p_{loss,0}^r - p_{loss}^r - \theta_0^r x_0^r \mathbf{1}) \quad (10)$$

subject to:

$$0 \leq p_{loss}^r \leq thr_{loss}^r \quad (11)$$

The master problem, which updates the  $\mu$  dual variable in the sub-problems, is:

$$\max_{\mu} \sum_r f_r(\mu) - |R_i| x_{source} \mu^T \mathbf{1} \quad (12)$$

subject to:

$$\mu \geq 0 \quad (13)$$

The  $\mu$  dual variable, dimensionally [1/bps], is the link congestion signal which binds flow link rates to not exceed the available rate. A higher  $\mu$  increases the fractional packet loss of one or more flows (i.e. we can see it as the equivalent of packet loss in TCP).

We are now interested to build upon the sub-problems, represented by eqs. (10) and (11). We first fix  $\mu_l$  on each link and then write the Lagrangian of eqs. (10) (11):

$$\begin{aligned} L(p_{loss}^r, \lambda_s) &= \mathbf{1}^T p_{loss}^r + \\ &+ \mu^T \frac{1}{\theta_0^r} (p_{loss,0}^r - p_{loss}^r - \theta_0^r x_0^r) + \\ &+ \lambda^T (p_{loss}^r - thr^r) + \\ &+ \alpha (p_{loss}^r)^T p_{loss}^r \end{aligned} \quad (14)$$

the new term,  $\alpha (p_{loss}^r)^T p_{loss}^r$ , is a regularization term which for  $\alpha \rightarrow 0$  returns the Lagrangian to the original and the original solution is preserved. By adding this term we obtain feasible primal solution to the original problem. The dual of eqs. (10) (11) is (here  $\mu$  is a constant, not a variable):

$$\max_{\lambda} L(p_{loss}^r, \lambda_r) \quad (15)$$

The resulting gradient algorithm to solve the dual problem in (15) is:

$$p_{loss,l}^r(t) = \frac{1}{2\alpha} \left[ \frac{\mu_l}{\theta_0^r} - 1 - \lambda_{r,l} \right]^+ \quad (16)$$

$$\lambda_{r,l}(t+1) = [\lambda_{r,l}(t) + \gamma (p_{loss,l}^r - thr_{loss,l}^r)]^+ \quad (17)$$

where  $\gamma$  is a positive step.

$\mu$  here represents the link reaction to congestion, while  $\lambda_r$  a node's  $s$  flow reaction to packet loss. The  $\lambda_r$  reaction decreases the packet loss for flow  $r$ . To this point, we don't have any intuition of how a flow may decrease its packet loss when this exceeds certain bounds. To understand this, we need to recall that there are two main causes of packet loss: (a) wireless link errors; and (b) interference due to flow contention. In this study we only account for interference losses. In practice, in a friendly (as opposed to hostile) environment we expect

wireless spatial contention to be the dominant effect, so that packet loss is mainly caused by interference between flows and  $\lambda_r$  is the flow reaction which leads flows to "well behave".

#### IV. THRESHOLD ADAPTATION MODEL

##### A. Threshold Adaptation

A problem we have not yet solved until now is how to choose the threshold values. We have shown that by implementing the algorithm derived in eq. (17) we are able to ensure fairness between competing flows, but threshold values were statically set to achieve the desired results. The threshold should instead be automatically set, dynamically adapting to traffic conditions.

In fact, with adaptive layers, the threshold has a very precise meaning: the quality degradation a user is willing to tolerate - or, the min quality - minus a proper margin. Setting the threshold to minimum quality level, with zero margins, would guarantee that each flow gets at least the minimum acceptable level (i.e. we find a feasible solution). For the solution to be optimally fair, however, the threshold must be as tight as possible.

We here design a heuristic algorithm to solve this problem. The rationale behind the algorithm is as follows. The wireless channel, the clique on which flows contend, is the bottleneck for competing flows, just as a bottleneck router in the Internet. An Internet router, when the incoming flow exceeds its bandwidth, drops packets, thus sending a TCP source a congestion signal. In a similar way a flow that implements FairCast, say flow A, sends a congestion signal to other flows that share the same wireless channel when flow A is suffering from contention. Other flows compare the congestion measure they receive from flow A (i.e.  $\lambda$ ) and decide whether flow A is in a better or worst situation and whether they should drop packets. In the wired case a TCP flow, receiving a congestion notification (i.e. 3 DUPACKs), lowers the slow start threshold to mitigate network congestion. A FairCast flow should then, in a similar manner, increase its threshold to be more tolerant to loss. Here follows the algorithm:

$$thr_r(t+1) = \begin{cases} \min\{thr_r(t) + a, 1\}, & \text{if } \lambda_c(t) > \lambda_r(t) \\ \max\{thr_r(t) - b, 0\}, & \text{otherwise} \end{cases} \quad (18)$$

where  $a$  and  $b$  are fixed parameters we will shortly investigate,  $\lambda_c(t)$  the highest of the lambdas received from competing nodes at flow  $r$  and  $\lambda_r(t)$  the lambda computed by flow  $r$ . In other words, if nobody complains, flow  $r$  keeps tightening up its threshold. We can

represent the fluid flow dynamics of the threshold as follows:

$$thr_r = ax_c(t)P\{\lambda_c > \lambda_r\} \quad (19)$$

$$- bx_c(t)P\{\lambda_c \leq \lambda_r\} \quad (20)$$

At the steady state, averaging over time and assuming that  $thr_r$  stabilizes in  $(0, 1)$  (i.e. boundaries excluded), we have that eq. (20)

$$P\{\tilde{\lambda}_c > \tilde{\lambda}_r\} = \frac{b}{a+b} \quad (21)$$

Where  $\tilde{\lambda}_c = E\{\lambda_c(t)\}$  and  $\tilde{\lambda}_r = E\{\lambda_r(t)\}$ , the expected values of  $\lambda_c(t)$  and  $\lambda_r(t)$  at steady state. The values of  $a$  and  $b$  should reflect the fact that a higher rate of flow  $r$  generates higher values of  $\lambda_c$  as a feedback from competing nodes. For this reason we use a heuristic that implements this concept, setting the values of  $a$  and  $b$  as follows:

$$a = k_1 \quad (22)$$

$$b = k_2 \frac{x_r^{out}}{x_r^{in}} \quad (23)$$

where  $k_1$  and  $k_2$  are constants we set by testing a wide number of scenarios in simulation,  $x_r^{in}$  is the incoming rate per flow  $r$  and  $x_r^{out}$  is the outgoing rate per flow  $r$ . With the values set in eqs. (22) and (23) we have that at steady state  $P\{\tilde{\lambda}_c > \tilde{\lambda}_r\} = k_2 \tilde{x}_r^{out} / (k_1 \tilde{x}_r^{in} + k_2 \tilde{x}_r^{out})$ . A high relative value of  $x_r$  induces a high probability that the received  $\lambda_c$  is higher than  $\lambda_r$ .

In Fig. 7 we have an example of the performance of three FairCast flows, in the same scenario discussed in Fig. 1, when implementing the discussed dynamic threshold algorithm. As we can observe, both fairness and aggregate utilization are well preserved.

## V. EVALUATION

We here show some preliminary results obtained using FairCast. In fact, it is possible to increase the number of flows that coexist by slightly lowering aggregate throughput and increasing fairness. The results are encouraging, indeed with a simple flow interaction protocol it is possible to improve the total QoS of the network.

The results we here show have been obtained using a threshold adaptation algorithm in the unicast scenarios. Intuitively, a flow with a lower loss rate at a node has a higher average threshold, while a flow with a higher loss rate a lower average threshold. We instead use fixed pre-set values in the multicast scenario. As we shall discuss in the following subsection, more work is needed to integrate FairCast and ODMRP.

## A. Results

We now repeat the same simulation shown in Fig. 1 adding FairCast. We can see in Fig. 7 that the middle flow, flow two, increases its goodput from as little as 40% to 70%, thus achieving a 75% improvement. In this simulation the symmetry of the topology makes flow react at the sources. In general, we observe that in the case of fixed routing FairCast is able to improve fairness between flows, redistributing channel access and leading spatially disgraced flows to achieve acceptable performance.

The next step is to integrate FairCast with a routing protocol. This must be carefully done, since this interaction may lead to unexpected results. We here test FairCast over ODMRP. Recall that ODMRP builds a mesh routing path between a source and its receivers. When FairCast runs over ODMRP, different branches will achieve different delivery ratios, as expected in an adaptive layer scheme. In fact, this is an important feature of FairCast, namely, the ability to adjust individual branch rates to local bandwidth availability. This feature is essential in heterogeneous receiver scenarios.

We also note that in Fig. 3, the overall delivery ratio for the video multicast application has decreased, while the audio rate has increased. This is a well known property of fairness - the total aggregate throughput is generally lower than that of the unfair solution.

## VI. CONCLUSION

In a network, flows generally adapt to network feedback such as loss and delay. This paper shows that flows can actually interact and collaborate adapting to in-network signals and achieving a local fair share of resources.

Past research on fairness in ad hoc networks only focused on unicast TCP flows and a number of optimizations have been introduced along the years to improve fairness and utilization of such protocol in wireless scenarios. We here present an algorithm by which it is possible to control the behavior, in a totally decentralized manner, between real-time multicast flows with adaptive layer adjustment/drop. FairCast flows are able to dynamically interact and acquire a fair share use of resources in a contention area.

Future work will aim at: (a) providing FairCast's advantages for both elastic and inelastic traffic; (b) integrate FairCast with a number of routing protocols; (c) optimize the decentralized algorithm for threshold adaptation for multicast flows.

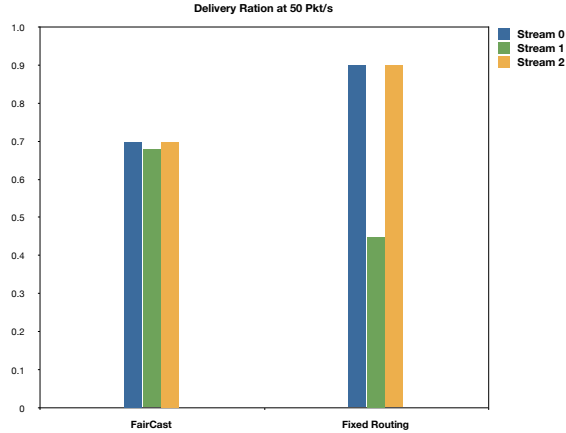


Fig. 7. Unfairness between three flows, FairCast performance compared with other protocols'

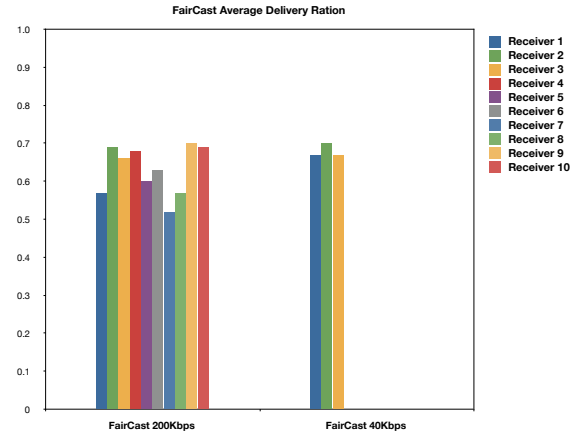


Fig. 8. Multicast scenario: two multicast flows, routed by ODMRP and running FairCast, compete in the same area. Multicast flow one adapts, by lowering its rate, and lets multicast flow two in.

## REFERENCES

- [1] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," in *RFC 2581*, April 1999.
- [2] M. Gerla, K. Tang, R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, February 25-26, 1999, New Orleans, Louisiana, USA.
- [3] C.E. Perkins, E.M. Royer, "Ad-Hoc On Demand Distance Vector Routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 90-100, February, 1999, New Orleans, Alabama, USA.
- [4] D.B. Johnson, D. Maltz, "Dynamic source routing in ad hoc wireless networks," in T. Imelinsky and H. Korth, editors, *Mobile Computing*, pp. 153-181. Kluwer Academic Publishers, 1996.
- [5] K. Xu, M. Gerla, L. Qi, Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'03)*, pp. 16-28, September 14-19, 2003, S. Diego, California, USA.
- [6] S. ElRakabawy, A. Klemm, C. Lindemann, "TCP with Adaptive Pacing for Multihop Wireless Networks," in *Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, Urbana-Champaign, IL, USA, May 2005.
- [7] S. Floyd, V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," in *IEEE/ACM Transactions on Networking*, V.1 N.4, August 1993, pp. 397-413.
- [8] H. Luo, S. Lu, V. Bharghavan, "A New Model for Packet Scheduling in Multihop Wireless Networks," in *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00)*, pp. 76-86, August 6-11, 2000, Boston, MA, USA.
- [9] J. Jun, M.L. Sichitiu, "Fairness and QoS in Multihop Wireless Networks," in *Proc. of the IEEE Vehicular Technology Conference (VTC'03)*, Orlando, FL, Oct. 6-9, 2003.
- [10] I. Gruber, A. Baessler, H. Li, "Fair WLAN Scheduling for Ad Hoc Networks with Access Points," Poster, in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'04)*, May 24-26, 2004, Tokyo, Japan.
- [11] A. Woo, D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks, in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, pp. 221-235, July 2001, Rome, Italy.
- [12] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs," in *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 212-225, September 1994, London, UK.
- [13] S.J. Lee, M. Gerla, and C. -C. Chiang, "On Demand Multicast Routing Protocol," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'99)*, pp. 1298-1302, September 1999.
- [14] S.J. Lee, W. Su, J. Hsu, M. Gerla, R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies of the IEEE (INFOCOM'00)*, March 26-30, 2000, Tel Aviv, Israel.
- [15] D. Palomar, M. Chiang, "Alternative decompositions for distributed maximization of network utility: framework and applications," in *Proc. IEEE Infocom*, Barcelona, Spain, April 2006.
- [16] D. Bertsekas, "Nonlinear Programming," Belmont, MA, Athena, 1995.