SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2010)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/sec.183

SPECIAL ISSUE PAPER

MANET QoS support without reservations

Soon Y. Oh*, Gustavo Marfia and Mario Gerla

Department of Computer Science, University of California, Los Angeles, CA 90095, U.S.A.

ABSTRACT

An inelastic flow is a flow with an inelastic rate, i.e., the rate is fixed, and it cannot be dynamically adjusted to traffic and load condition as in elastic flows like TCP. Real time, interactive sessions, and video/audio streaming are typical examples of inelastic flows. Reliable support of inelastic flows in wireless *ad hoc* networks is extremely challenging because flows and routes dynamically change and flows compete for the shared wireless channel. Bandwidth must be reserved for inelastic flows at session set up time. To avoid repeated attempts to set up reservations in a 'volatile' network and prevent serious network capacity degradation due to call set up overhead, a Call Admission Control strategy robust to mobility must be developed. In this paper we propose ProbeCast, a probe based call admission control scheme with QoS guarantees for inelastic flows. ProbCast was designed for multicast streams but can also work, by default, for unicast. In ProbeCast, a path (or a tree) is probed for capacity availability. If an intermediate link along the probed path fails to meet the QoS requirement, the flow is 'pushed back' via backpressure upstream to an intermediate branch or possibly to the source. The backpressure principle is simple; however, its implementation requires some care to avoid unfairness and eventual capture by one of the flows sharing a congested bottleneck. We show that proportional fairness among inelastic contenders will prevent capture. To achieve this, we have developed the Neighborhood Proportional Drop (N-PROD) scheme. N-PROD guarantees fair rejection of unfeasible flows and maintains the same proportional drop rate among surviving flows in the same contention domain. We demonstrate the efficacy and robustness of ProbeCast for unicast as well as multicast scenarios using the Qualnet simulation platform. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS

MANET; QoS; call admission control; inelastic flow

***Correspondence**

Soon Y. Oh, Department of Computer Science, University of California, Los Angeles, CA 90095,U.S.A.
E-mail: soonoh@cs.ucls.edu

1. INTRODUCTION

In emergency and tactical wireless Mobile Ad Hoc Networks (MANETs), audio and video streaming are essential requirements for group interoperation. Even commercial MANETs (e.g., vehicular networks) will be faced with multimedia streaming because of the popularity of applications such as P2P TV and YouTube. We can expect that future MANETs will be designed to handle 'inelastic' flows, both uni and multicast with QoS (bandwidth and delay) requirements, in addition to 'elastic' flows such as TCP.

Most of the previous work on QoS in MANETs is based on resource reservation on a selected path before transmission can commence [1–5]. Well known limitations of traditional Call Admission Control (CAC) strategies are: the complexity of the available bandwidth estimation in a shared wireless environment (where each allocation impacts several other flows in different ways depending on their relative positions and transmission ranges), the volatility of the 'available' bandwidth estimation due to the rapidly changing topology, the need to frequently reallocate resources due to node mobility, and the overhead introduced by the frequent updates. However, if inelastic calls are accepted without any attempt to verify the availability of resources, the situation is even worse: congestion will set on and no inelastic calls can get through!

As a solution to this dilemma, we propose ProbeCast, a 'probing' based CAC scheme for inelastic flows that does resource verification and allocation at the same time without prior reservations. Namely, a new flow first probes the availability of resources, e.g., evaluating packet drop rate at intermediate nodes without any notion of resource reservation. If an intermediate link fails to meet the QoS requirements, the flow is 'pushed back' by sending a backpressure message upstream to the source (or intermediate branch) — no time and effort was spent so far for reservations. As a result of backpressure, the incoming flow is either rerouted or rejected. Once the inelastic flow is established,

it cannot be displaced by incoming inelastic flows because of a built in priority of incumbent versus incoming flow.

The backpressure strategy works only if the congested link is shared with *proportional fairness* among inelastic contenders in the same contention domain. To understand this concept, note that in the wired Internet, when a link becomes congested and the queue overflows, the packet drop rate of each flow is proportional to its rate, namely, the drop probability is uniform across flows. Without loss of generality, assume all flows are inelastic. If a new probing flow finds enough capacity on intermediate links and suffers no loss, it successfully completes the call set up and is promoted to established flow with higher 'priority'. If the new flow 'does not fit' in the bottleneck, i.e., it causes congestion, and drops packets, its drop probability is equal to that of the incumbent flows. By setting a lower drop probability threshold on new flow than on incumbent flows, the new flow is automatically discriminated and backpressured, leaving the incumbent flows undisturbed; this is exactly the property we seek in a Call Acceptance Control scheme.

A CAC approach inspired to probing was proposed several years ago for Internet VoIP streams [6]. In the Internet, where competing flows share a single common queue in the router, proportional fairness, and more generally resource allocation are rather straightforward. In the wireless medium, there is no single common queue. In fact, there are several queues that independently adjust their MAC parameters (including retransmission rates) in case of loss. Thus, there is the risk of unfairness and channel capture by 'big flows' and by flows with a relative 'interference graph' advantage when the wireless medium becomes congested. Clearly, a distributed proportional fairness scheme must be developed for wireless channels to overcome the lack of a centralized control point. To this end, we have complemented the probing scheme with a distributed fairness scheme, *Neighborhood Proportional Drop (N-PROD)* which enforces uniform drop probabilities among flows competing in the same contention domain. Each node estimates own packet drop probability and propagates this information by piggybacking to neighbors. As mentioned earlier, in ProbeCast, the incoming flow has by design a lower drop probability threshold than the incumbent flows. If during probing, the new flow drop rate increases beyond the threshold, the flow is backpressured toward the source node and the flow is rerouted. If backpressure pushes the flow back to the source and all alternate routes are exhausted, ProbeCast rejects the incoming flow.

The problem of fair sharing among inelastic multicast flows and the concept of proportional fairness was introduced in a companion paper that appeared in MSWIM 2008. This paper extends that work by adding a Call Admission Control scheme based on backpressure. The major contribution of ProbeCast is to enable CAC and fair allocation of inelastic flows in MANETs, *for both unicast and multicast streams*, without requiring prior resource reservation and thus overcoming the overhead limitations of traditional MANET reservation and allocation of CAC schemes.

The rest of the paper is organized as follows: Section 2 illustrates the related works, Section 3 describes the details of the ProbeCast, Section 4 exhibits analytical model of ProbeCast, and Section 5 presents simulation results. Finally, Section 6 contains conclusions and future work.

2. RELATED WORK

ProbeCast builds upon FairCast [7] adding a CAC scheme to FairCast. In fact, FairCast's rationale is to monitor flows and adjust their rates, locally, so that each flow receives an 'acceptable' share of bandwidth. 'Acceptable' means that a sufficient amount of bandwidth is available for multimedia error correction schemes to recover. This may not always be the case, in some cases the bandwidth demand may exceed any network and coding capability. ProbeCast's contribution, then, is to guarantee a minimum QoS level in the network following FairCast's philosophy of implementing a completely distributed algorithm.

A number of *ad hoc unicast* QoS support protocols and algorithms have been proposed in the literature, e.g., INSIGNIA [8], SWAN [9], and FQMM [10]. However, relatively few MANET *multicast* QoS schemes have been introduced. We classify existing MANET multicast QoS schemes into three categories based on their resource measurement and reservation methods.

2.1. Bandwidth estimation and resource reservation

Ad hoc QoS Multicasting (AQM) [1,2] achieves multicast QoS support by tracking available neighbor nodes resources. Nodes periodically broadcast a hello message including own bandwidth usage. Upon receiving the hello message, nodes record neighbor information in a neighborhood table which is used to calculate the total bandwidth allocation to existing multicast sessions. When starting a multicast session, a node floods an initiation packet. Intermediate nodes forward it on feasible links based on the neighborhood table. AQM hello messages introduce considerable overhead in a mobile network, interfering with QoS support.

The Lantern Tree based QoS on-demand Multicast (LTM) [3] relies on a multipath structure, called a lantern-path. LTM employs the lantern tree as a routing path in *ad hoc* multicast and it uses a CDMA-over-TDMA model at the MAC layer to allow the superposition of many flows. LTM exploits CDMA orthogonal multiuser capability to allocate an extra flow in an already occupied network. QoS is guaranteed only to the extent that the load is kept in check (else losses escalate). Main implementation drawback is the need for a non standard CDMA-over-TDMA MAC with distributed time synchronization requirements.

QoS Multicast Routing Protocol (QMR) [4] is an on-demand mesh based protocol that uses a forwarding mesh like On-Demand Multicast Routing Protocol (ODMRP) [11]. QMR defines Forward Nodes (FNs) which

establish a forwarding mesh and provide multiple paths. FNs reserve bandwidth for a multicast session if they can accept QoS route request (QREQ) from the source. Upon receiving data packets from multicast sessions WITHOUT reservations, they forward them only if 'shared' bandwidth is available. To implement this hybrid scheme, nodes divide bandwidth into *fix reserved* and *shared* bandwidth. The mesh structure can guarantee good delivery ratio via redundant forwarding. However, flood type redundancy may lead to congestion and excessive overhead effecting QoS performance of reserved flows.

2.2. End-to-End probing

Call Multicast Admission-Protocol for MANETs (M-CAMP) [12] is an end-to-end probing protocol in which a source, before transmitting the data stream floods probing packets to test bandwidth availability along the multicast tree. Only the receivers participate in the CAC decision by submitting an accept/refuse decision to the source based on the received quality. Similar to PCP [6] M-CAMP employs three priority levels among packets: *real time, probe, and best effort*. Level 2 probing packets do not affect existing QoS flows. To cope with mobility, after topology changes, a new resource probing process is started to rebuilds the tree and the allocation.

QAMNet [5] establishes a QoS aware mesh using Join-Probe and Probe-Response control packets. QAMNet exploits MAC layer feedback to estimate available bandwidth. Like QMR, a source floods a Join-Probe packet when it has packets to transmit. Intermediate nodes update a bandwidth field in the Join-Probe packet to reflect minimum available bandwidth along the path. After collecting one or more Join-Probe packets, a receiver sends a Probe-Response to source if a feasible path is found.

The common limitation of all the above end to end schemes is the inability to prevent unfairness and capture. In addition, QAMNet incurs the burden of local available bandwidth estimation.

2.3. Bandwidth fair sharing

As mentioned earlier, Marfia *et al*. designed an algorithm, called FairCast [7]. The main focus of FairCast is congestion control and bandwidth fair sharing across multicast flows. To do this, FairCast uses local flow interactions and packet dropping; no end-to-end feedback or backpressure. Each flow has a packet loss threshold that corresponds to the minimum acceptable quality. If a flow experiences a loss rate that exceeds the desired threshold, it promptly reacts against the competing flows requesting them to drop at a certain rate. Flows exchange (piggybacked) loss information and employ selective packet drops to equalize their loss rates. This is called Distributed Gentleman Agreement (DGA) algorithm [7]. DGA algorithm is appropriate for *ad hoc* networks and brings the following advantages: no bandwidth is wasted due to end-to-end feedback, flows are faster in adapting to highly dynamic traffic changes, the solution is fully distributed, and it achieves 'proportional fairness' in wireless multicast. ProbeCast N-PROD protocol enforces equal drop rates among flows within the collision domain. It is inspired by DGA. Once the loss rate exceeds the threshold, N-PROD stops selective packet drop. At this point, ProbeCast initiates back-pressure of selective flows toward the source to request rejection or reroute. In FairCast, DGA also starts packet drop when the loss rate reaches the threshold; however, it does not attempt to reroute or reject the flow. Namely, the main difference between the two protocols is that FairCast does not exercise Call Admission Control.

3. PROBECAST

In this section, we present a detailed description of ProbeCast.

3.1. Assumptions

In ProbeCast, a key underlying assumption is that inelastic flows are protected by some form of end to end FEC (e.g., erasure coding, fountain codes, raptor coded, etc.). Namely, a sender adds redundancy to its stream, in the form of error correcting code which allows a receiver to detect and correct errors (within some limits) at the expense of some extra delay, without the need for retransmission. This is critical in MANET multicast sessions since conventional ACK and retransmission techniques between a sender and receivers may cause 'ACK/NACK' implosion. We also assume that inelastic flows are classified into several priority levels. Each level is given a maximum tolerable loss rate which (in the erasure code implementation) corresponds to a packet drop threshold. The packet drop threshold is carried in the packet header. An intermediate node backpressures the flow when the packet drop rate exceeds the threshold. ProbeCast is independent of the underlying multicast routing protocols (in our simulation experiments we will use ODMRP). ProbeCast works equally well with unicast (a special class of multicast). It can coexist with lower priority best effort traffics (e.g., TCP), and will throttle best effort traffics to make room for inelastic, higher priority flows.

3.2. Packet drop probability

ProbeCast delivers drop probability information via piggybacking in the packet header. To calculate the packet drop probability, each intermediate node keeps a time window based revolving count of received and lost packets. In addition to the sequence number stamped by the source and used to discard duplicates, each intermediate node keeps track of flows and assigns local sequence numbers to packets in each flow. Local flow bookkeeping is generally unacceptable in the wired Internet because of scalability considerations. In our case, scalability is not violated due to the rather limited

number of simultaneous inelastic flows in a single MANET node. When transmitting a data packet, an intermediate node updates the local sequence field in the packet header. Upon receiving a packet, a down-stream node increments the number of received packets and monitors the local sequence number in the packet header. If there is a gap, a packet was lost. A node also tracks the number of packets dropped from its queue. It does not, however, attempt to monitor packet drops on outgoing links to neighbors. This count is the responsibility of the downstream neighbors, which eventually report the loss to upstream nodes. Every time unit, a node estimates its packet drop rate. Since the estimate fluctuates, ProbeCast uses a weighted average to smoothen fluctuations. Drop probability computation follows:

$$DN_i^r(t) = DqN_i^r(t) + DlN_i^r(t) \quad (1)$$

$$D_i^r(t) = \frac{DN_i^r(t)}{RN_i^r(t) + DlN_i^r(t)} \quad (2)$$

$$P_i^r(t) = \alpha P_i^r(t-1) + (1-\alpha) D_i^r(t) \quad (3)$$

where

- t is the tth time interval
- DN_i^r is the total dropped packet rate at node i, flow r
- DqN_i^r is the dropped packet rate at the queue at node i, flow r
- DlN_i^r is the dropped packet rate on the incoming link to node i, flow r
- RN_i^r is received packet rate at node prior to drop i, flow r
- D_i^r is the calculated packet drop rate at node i, flow r
- P_i^r is the packet drop probability at node i, flow r
- α is the constant value, called step constant
- Thr^r is the drop probability threshold for flow r.

Algorithm 1 summarizes the Node Drop Probability computation based on the above formulas. If a node relays multiple inelastic flows, each flow may have a different packet drop probability. To reduce overhead, instead of sending all drop probabilities, it suffices for a node to propagate just the highest value. For convenience, we call this value the Node Drop Probability hereafter. Upon hearing the neighbor Node Drop Probability, a node sets own Node Drop Probability by neighbor's value if neighbor's Node Drop Probability is higher than its own value. Node Drop Probability values are timed out and refreshed to account for 'lossy' neighbors that move away.

Algorithm 1 Calculates the node drop probability

DEFINITIONS: For each flow r, DqN_i^r is the number of dropped packets at the queue in a unit time, DlN_i^r is the number of dropped packets on the link, and RN_i^r is the number of received packets at node i prior to queue drop. Node monitors DqN_i^r, DlN_i^r, and RN_i^r. P_i^r is packet drop probability of flow r at node i. $Prob_i$ is the Node Drop Probability of node i and Thr_i^r is the drop threshold of flow r.

```
for each flow r in i do
  D_i^r(t) = (DqN_i^r(t) + DlN_i^r(t)) / (RN_i^r(t) + DlN_i^r(t))
  P_i^r(t) = α P_i^r(t − 1) + (1 − α) D_i^r(t)
  if P_i^r(t) > Thr_i^r then
    Prob_i = P_i^r(t)
    SetBackpressureFalg(r)
  else if P_i^r(t) > Prob_i or Prob_i is timeout then
    Prob_i = P_i^r(t)
    RecordtheUpdatedTime(Prob_i)
  end if
end for
```

3.3. Neighborhood proportional drop

N-PROD allows inelastic flows to acquire resources in a 'fair' and totally distributed manner without resource reservation. It enforces proportional drop rates among flows competing in the same contention domain. Note that proportional fairness is not generally desirable in elastic flows such as best effort data sessions controlled by TCP. In fact, a popular TCP fairness scheme called NRED [13] enforces *uniform* drop probability so that all the TCP flows in the same contention domain achieve the same throughput. *Proportional drop* in N-PROD can enforce different throughputs for different inelastic flows with different nominal rates. For example, if flow A and B send 100 Kbps and 60 Kbps, respectively, and N-PROD control stabilizes at 20% drop probability, flows A and B throughputs stabilize at 80 Kbps and 48 Kbps respectively. In contrast, TCP fairness strives to equalize flows.

In our implementation Section 3.2, each node reads Node Drop Probability values from overheard packets and adjusts its Node Drop Probability value. If Node Drop Probability of the neighbor node is higher than its value, a node replaces own Node Drop Probability by neighbor's value; otherwise, the node ignores it. To enforce drop probability, before forwarding a packet, the node generates a random number to compare it with the target Node Drop Probability. If the number is smaller than the Node Drop Probability, the packet is dropped from the queue; otherwise, it is forwarded. As a 'heuristic' exception, the packet is not dropped if it belongs to the flow with highest Node Drop Probability, in order not to further hurt that flow.

3.4. Backpressure

The flow packet drop threshold depends on traffic class, encoding rate and age of the flow. For example, assume three inelastic flows have 50%, 40%, and 30% drop thresholds, respectively. The first flow is more loss tolerant than the others. It will be more difficult to reroute or reject it once it is established. By the same argument, a new entering flow typically has a lower drop threshold than existing flows and thus it is the first to be rejected in case of overload.

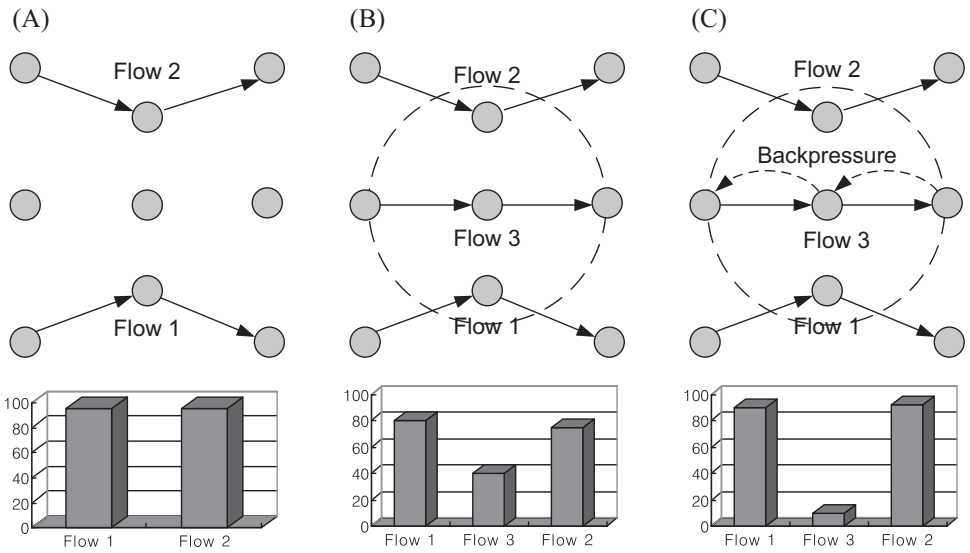

Figure 1. Three flows in the simple topology. Lower graphs show packet delivery ratios, presented by percentage. (A) Two flows are presented both with have high delivery ratios over 90%. (B) Flow 3 starts transmitting and other flows' delivery ratios decrease because of channel contention. (C) Flow 3 packet drop rate is exceeded the threshold and backpressure start.

When the packet drop rate is over the threshold, the flow is backpressured toward its source. The backpressure mechanism uses piggybacking to reduce overhead. Upon getting a backpressure signal from a neighbor, the node checks if the neighbor is one of its downstream forwarders for that flow. If so, it will remove the downstream node from the list. It will then check the list to determine if there are any other downstream forwarders or local receivers for the flow in question. If there are none, the node will forward the backpressure signal to its upstream node. This way, all non productive branches of the multicast tree are pruned. If the backpressure signal reaches the source, the flow is rejected (i.e., there is no receiver ready for it). Alternatively, the source can attempt to construct a new multicast tree/mesh by searching for lightly loaded paths.

Figure 1 illustrates an example of new flow rejection via backpressure. In Figure 1 (A), two inelastic flows, Flow 1 and 2, are initially allocated to two non interfering paths. In Figure 1 (B), a new flow, Flow 3 is injected by ProbeCast. It starts transmitting packets which interfere with Flow 1 and 2. Thus, Flow 3 shows low delivery ratio since it must compete against the other two flows. Finally, Flow 3 drop rate goes over the drop threshold and in Figure 1 (C) backpressure and rejection occurs. The other flows are restored to their original rates.

4. PROBECAST OPERATION MODEL

ProbeCast can be inserted in a similar optimization framework as authors do for FairCast [7]. In fact both protocols rely on the definition of interaction rules and selective drops. ProbeCast is a superset of FairCast in which it also implements admission control (besides providing fairness among feasible flows of equal priority) and thus prunes unfeasible branches in a multicast distribution graph. As in Reference [7], we formulate the problem considering a single multicast communication. Multiple unicast and multiple multicast models can be derived by extending this basic model.

In the sequel 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^r_{loss} is a vector of $|L|$ rows, for receiver r. Row $p^r_{\text{loss},l}$ represents the average fraction of packets lost, for receiver r, on link l.
- $p^r_{\text{loss},0}$ and x^r_0 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^r_l represents the average rate of flow r on link l.
- x_{source} is a scalar. This is the maximum rate at the source.

4.1. Optimization problem

An in depth description of the optimization formulation and of the techniques used to design the distributed algorithm can be found in Reference [7]. We first describe the rationale behind the optimization function's design choice and then introduce the main optimization problem.

Different optimization functions could have been chosen instead of Equation 4: minimize the maximum drop probability, jointly minimize the average drop probability and the maximum drop's standard deviation, etc. Our choice can be explained by our aim at implementing a completely distributed algorithm. In fact, we trade an optimal solution

for a distributed solution. For example, the choice of minimizing the maximum drop probability would have required a network-wide knowledge of each flow's drop rate. In the formulation of Equation 4, instead, it is possible to relax the problem and decouple the variables so that drop choices are taken at each node.

ProbeCast adds a call acceptance control scheme to FairCast, therefore this should be taken in account in ProbeCast's model. ProbeCast behaves as FairCast until, through probing, it discovers that it is not possible to adjust a new flow in the network without severely damaging the others; we will assume that FairCast's model holds until the acceptance control phase does not kick in. We will now concentrate on ProbeCast's behavior, once ProbeCast flows decide that a flow should be shutdown. ProbeCast's model can be written as follows:

$$\min \sum_r \mathbf{1}^T p^r_{\text{loss}} \tag{4}$$

subject to:

$$p^r_{\text{loss}} \in \{0, 1\} \tag{5}$$

$$x_r \geq 0, \forall r \in R_i \tag{6}$$

$$\sum_r x_r \leq |R_i| x_{\text{source}} \mathbf{1}. \tag{7}$$

where vectors $p^r_{\text{loss}} = p^{r,c}_{\text{loss}} + p^{r,d}_{\text{loss}}$, $p^{r,c}_{\text{loss}}$ represents the fraction of packets lost due to contention, $p^{r,d}_{\text{loss}}$ represents the fraction of packets lost due to FairCast drop decisions. Clearly, $p^{r,d}_{\text{loss}}$ are the variables in the optimization problem. As pointed out in Reference [7], an equation of type $x^r_l = F(x, p)$ is missing. Namely, we lack an accurate analytic model that relates the flow on each link to flows and loss rates in the rest of the network. As in Reference [7] we solve this problem in the numerical optimization procedure by using data extracted from simulation. Namely we are here defining a Simulation-Optimization problem, a generalization of a Deterministic Optimization problem, where one or more of the constraints are observable through a stochastic simulation. A simulation–optimization approach is an approach where one or more functions of an optimization problem are implemented in simulation. In our case, QualNet implements the function $x^r_l = F(x, p)$. Observing p^r_{loss} at each node, each node sets new values for $p(r, d)$ loss according to Algorithms 1 and 2 [14].

We should observe one important fact, the problem defined in the above equations defines a integer program, differently than in Reference [7]. This takes into account the fact that, implementing an acceptance control scheme, ProbeCast implements everything or nothing policy for its flows. Clearly, the model summarizes the call acceptance control effect in Equation 5, which defines that a receiver can either lose everything or nothing. In FairCast, in contrast, the flows' loss rates can be increased without limits, until fairness is achieved.

Algorithm 2 N-PROD algorithm

DEFINITIONS: D^r_i is packet drop rate of flow r at node i. $Prob_i$ is the Node Drop Probability of node i and Thr^r_i is the drop threshold of flow r. pkt^r and pkt^s are the packet of flow r and s, respectively.

Node i receives pkt^s from Node j
Node Drop Probability of i is $Prob_i$
$NumReceivedPkt = NumReceivedPkt + 1$
if $Prob_j > Prob_i$ and $Prob_j$ and $Prob_i$ are different flows **then**
  $Prob_i = Prob_j$
  RecordtheUpdatedTime($Prob_i$)
end if

Node i sends pkt^r
Node Drop Probability of i is $Prob_i$ and it is flow s
while queue is not empty **do**
  if packet is pkt^r and $Prob_i$ is not flow r **then**
    if $Prob_i > uniformRandom[0, 1]$ **then**
      PacketDrop(pkt^r)
    end if
  end if
  $pkt^r \rightarrow DropProb = Prob_i$
  PacketSend(pkt^r)
end while

The objective, expressed by Equation 4, is to minimize the fraction of packets lost in the network. Constraint Equation 5 defines the acceptance control model. Equations 6 and 7 are feasibility constraints, where Equation 7 limits the total rate at receivers to be bound by the rate at source by the number of receivers.

4.2. Convergence discussion

In FairCast and consequently in ProbeCast we trade the search of an optimal solution for the search of a distributed solution. FairCast converges to an optimal solution under two main assumptions. The first assumption is that the mobility pattern and, therefore, routes between source–destination pairs are stable compared to the network traffic dynamics. The second assumption is that the objective function in Equation (7), the total loss rate in the network is convex in the flows, and the constrained set is a convex set. The second property is very difficult to prove in general, given the complexity of the loss rate expression in a multihop wireless environment. Thus, FairCast generally converges to a sub-optimal solution. More details are found in Reference [7].

ProbeCast's convergence to a local minimum (in terms of aggregate loss rate) derives from the convergence of FairCast. Following that path, we can state that ProbeCast can reach a local minimum in the case the incoming flow is accepted and a steady state solution is reached. If the incoming flow is rejected (or better, if several branches of the

incoming multicast flow are blocked), ProbeCast converges to the same, stable solution regardless of the control parameters used in the 'probing' and 'backpressure' procedures. Namely, the ProbeCast algorithm is stable. ProbeCast convergence is also guaranteed if one posed a slightly different problem and considered a set of F flows already present in a network, each with different drop threshold depending on priority and age. If ProbeCast is applied to this set, it will drop the same flow regardless of control parameters. This situation is of practical importance in a MANET where nodes move and network 'capacity' changes constantly. In fact, this is where ProbeCast outperforms traditional, reservation based acceptance control schemes. The traditional schemes, once they accept a flow, can no longer reject it. ProbeCast instead constantly readjust flow membership to reflect changes in topology and capacity. However, the careful reader will note that, if F flows are competing to enter the same network, the final set of accepted flows will depend on the order in which the flows are coming in. This is because we tighten the drop threshold for an incoming flow, making it easier for it to be dropped.

5. SIMULATIONS

In this section, we validate ProbeCast using Qualnet v3.9.5 [15], a packet level network simulator. To this end we implemented ODMRP, QAMNet, FairCast, and ProbeCast all in Qualnet. Comparison with ODMRP and FairCast teaches us how ProbeCast successfully builds on prior schemes (which are actually its ingredients) to achieve the desired goals. Comparison with QAMNet gives an indication of how ProbeCast fares against competitors. The FairCast version used in this section is based on the N-PROD module. It is a very close approximation of the FairCast scheme published in Reference [7].

5.1. Simulation setup

We use 802.11b with 2 Mbps channel capacity and 376 m effective reception range. Interference range is not given as input, rather it is generated by the simulator. It is approximately twice as large as the reception range. Channel propagation model is the two-ray ground reflection model. The packet size is 512 bytes and the maximum queue size is 50 Kbyte (about 100 packets). We use Qualnet default values for MAC and Physical layer configuration parameters.

ProbeCast and FairCast can run on any *ad hoc* multicast routing protocol. In our simulation, we chose ODMRP. In ODMRP, a source periodically floods a Join Query packet into the whole network. Upon receiving a non-duplicate Join Query packet, every node in the network stores the upstream node address for reverse path learning and rebroadcasts it. When the Join Query reaches a multicast receiver, the receiver creates and broadcasts a Join Reply packet to its neighbors. This Join Reply packet is relayed all the way back to the source following the learned reverse path. Nodes on the reverse path become the forwarding group. Data is delivered along the mesh consisting of the forwarding group nodes. We use the ODMRP implementation included in the Qualnet package. The Join Query refresh interval is 3 s and the forwarder life time is three times the Join Query refresh interval.

We chose the *ad hoc* multicast call admission control protocol QAMNet as it is representative of the QoS traditional resource reservation approaches. Moreover, QAMNet is built on ODMRP, making the comparison with ProbeCast easier. QAMNet employs distributed resource probing and admission control as well as adaptive rate control of elastic flows in *ad hoc* multicast. The design of QAMNet is based on existing approaches: mesh based *ad hoc* multicast protocol ODMRP and *ad hoc* unicast QoS protocol SWAN [9]. Like ODMRP, QAMNet is on-demand style and periodically refreshes the route, e.g., every 3 s, exchanging control packets which include available resource information as well as route information. Bandwidth availability estimation at the local node is calculated similar to SWAN. Namely, a new inelastic flow is admitted during the first multicast route establishing period and the admitted flow bypasses a local rate controller to guarantee required QoS. If resources are insufficient, the inelastic flow is classified as a non-admitted flow and is degraded to 'elastic status'. Traffic rates of elastic flows and non-admitted inelastic flows are controlled by the local rate controller which uses Additive Increase Multiplicative Decrease (AIMD) algorithm to achieve fairness and efficiency in allocating resources. Every T seconds, the rate controller monitors local MAC layer backoff delay of packets and adjusts the traffic rate. For example, if the controller detects excessive delay, the traffic rate decreases multiplicatively; otherwise the rate additively increases. This AIMD algorithm is inspired by SWAN rate control scheme. From the above we note that QAMNet is not strictly a CAC mechanism since it does not reject unfeasible flows.

In our comparison we use four metrics: Throughput is the total received data bits divided by the total simulation time, Packet Delivery Ratio is the fraction of received data, End-to-end delay is the average delay from the source to receivers, and Number of Packet Sent is the aggregated number of packets sent by a source. All numbers are averaged over 100 simulation runs except for the Number of Packet Sent and confidence level is 95.

5.2. Three even unicast inelastic flows

We first tested the four protocols using three parallel flow topology shown in Figure 2. In the process, we also show the difference between uniform and proportional fairness. The scenario is that three inelastic flows in Figure 2 use disjoint paths, but they still interfere with each other. For each flow, the source and the destination are located out of each others' transmission range and communicate only with intermediate nodes, more precisely node F1, F2, and

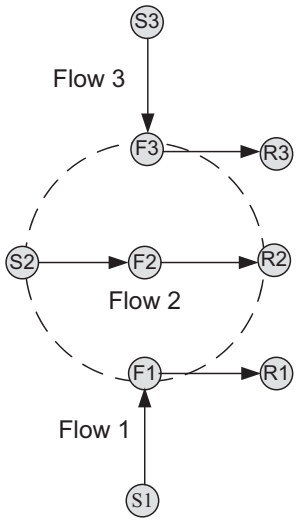

Figure 2. Three parallel inelastic flows topology example. Intermediate nodes, F1, F2 and F3 are within radio range and they compete with each other. Sources, S1, S2, and S3 are outside of other's radio range.

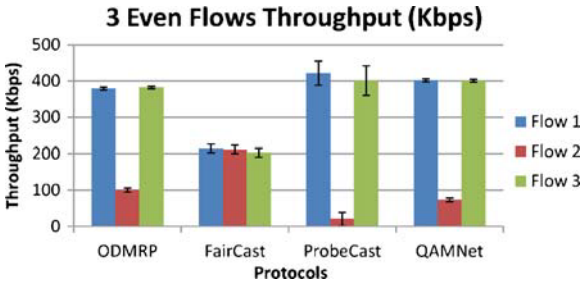

Figure 3. Throughput of three inelastic flows. Uniform nominal rate = 500 Kbps.

F3. The distances between node F1, F2, and F3 are 350 m and thus they hear each other and compete for the medium. The flows are inelastic; the sources, S1, S2, and S3, send data at a constant, uniform rate = 500 Kbps. Flow 1 starts transmitting data 1 s after simulation initialization. Flow 2 and 3 start data sending $T = 10$ s and 20 s, respectively. Because node F2 is located within F1 and F3's transmission range, Flow 2 packets are at a disadvantage and are dropped at F2 at a higher rate than the other flows. As a result, only a few Flow 2 packets reach the destination. This behavior is clearly exhibited by ODMRP in Figure 3.

The application of FairCast restores fairness as shown in Figure 3. This result is very similar to the result reported in paper [13]. This is not surprising since with uniform inelastic rates, FairCast is equivalent to NRED. ProbeCast and QAMNet exhibit a behavior that resembles that of ODMRP in Figure 3. The main difference, however, is the way they deal with Flow 2. Recall that ODMRP does not have admission and rate control mechanism, thus S2 transmits packets as 500 Kbps rate. However, Flow 2 is interfered by Flow 1 and 3 so that most of packets are dropped. Since there is not enough bandwidth, QAMNet does not admit Flow 2 so

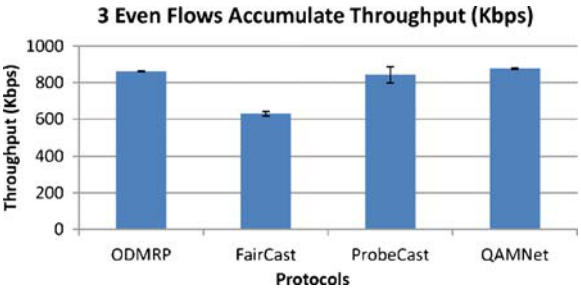

Figure 4. Accumulated total throughput in the network in three inelastic flows case. Uniform nominal rate = 500 Kbps.

that it becomes a non-admitted flow. Its rate is controlled by S2 and F2 without rejection. Thus, in QAMNet Flow 2 throughput is less than in ODMRP while it is higher than in ProbeCast for the reason discussed next. In ProbeCast, in fact Flow 2 is rejected. When Flow 2 starts, ProbeCast tries to balance Flow 1 and Flow 2 drop rate, but Flow 2 drop rate exceeds the threshold due to lack of bandwidth. Figure 1 presents Flow 2 situation just before rejection.

Figure 4 shows total aggregate throughputs. As already noted in Reference [13], fairness comes at the cost of degraded total throughput. ProbeCast and QAMNet rejects/controls Flow 2 traffic and thus Flow 1 and 3 earn bandwidth. Consequently, aggregate throughput for the latter is the same as ODMRP.

Figure 5 illustrates end-to-end delay respectively. End-to-end delays of ODMRP, FairCast, and ProbeCast are around 20 ms consistently for all three flows. In QAMNet end-to-end delay is higher than for the other schemes. In particular, Flow 2 and 3 end-to-end delays are extremely high: Flow 2 delay is over 2 s and Flow 3 delay is over 0.1 s. The reason is that ODMRP, N-PROD, and ProbeCast have no traffic rate control so that packets are dropped at intermediate nodes. This packet drop does not affect end-to-end delay. On the contrary, QAMNet controls traffic rate if the flow is not accepted. In Figure 2, since Flow 2 is non-admitted flow, S2 and F2 control traffic rate. Channel is busy and thus MAC layer backoff increases at S2 and F2. Due to AIMD algorithm, traffic rate decreases and Flow 2 end-to-end delay increases enormously. Moreover,

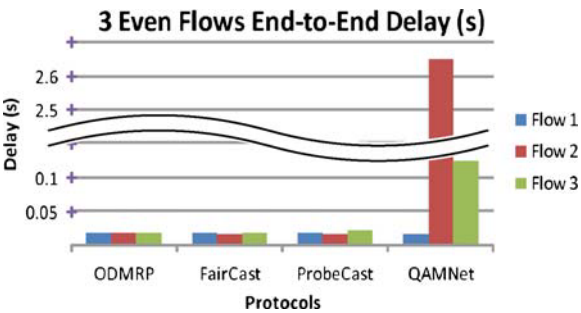

Figure 5. End-to-end delay of three inelastic flows. Uniform nominal rate = 500 Kbps. (QAMNet Flow 2 delay is over 1 s so it is removed from the graph).

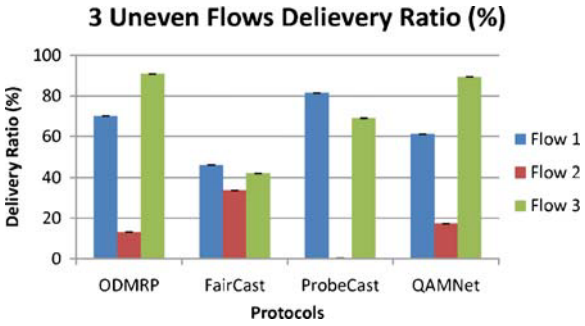

Figure 6. Delivery ratio of uneven three inelastic flows case. Flow 1, 2, and 3 are 800 Kbps, 400 Kbps, and 200 Kbps, respectively.

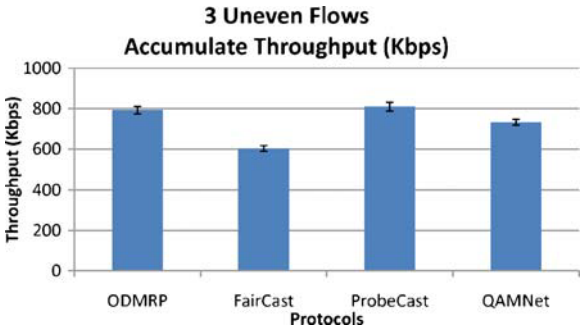

Figure 8. Accumulated total throughput in the network of uneven three inelastic flows case. Flow 1, 2, and 3 are 800 Kbps, 400 Kbps, and 200 Kbps, respectively.

QAMNet sometimes seems to accept all three flows or does not admit Flow 3 instead of Flow 2 due to incorrect and unreliable resource monitoring. In these cases, Flow 3 end-to-end delay increases significantly (above Flow 1) because of contention and traffic control at F3 as shown in Figure 5. The abnormal delays of QAMNet are a definite cause of concern in the support of inelastic flows (such as video streaming, say) with important delay constraints.

5.3. Three uneven unicast inelastic flows

In the next simulation experiment, we use the same layout but now the inelastic flows send data packets at different rates. Namely, Flow 1 = 800 Kbps, Flow 2 = 400 Kbps, and Flow 3 = 200 Kbps. Like in the previous experiment, Flow 1 starts transmitting data at $T = 1$ s. Flow 2 and 3 start transmissions at $T = 10$ s and $T = 20$ s, respectively.

Figure 6 shows packet delivery ratios of protocols and Figure 7 illustrates throughputs of flows.

In ODMRP, without fairness or admission mechanism, Flow 1 and 3 capture the channel and Flow 2 is starved. Consequently, Flow 3 throughput, about 180 Kbps, is higher than Flow 2 throughput, 55 Kbps, while S2 sends packets with higher rate than S3 in Figure 7. With FairCast, each flow drops packets proportionally to its demand. Thus, drop probabilities are uniform and achieved throughputs are staggered as the demands (in the ratios 8:4:2) as shown in Figure 7. This result is clearly different from the uniform throughputs achieved with NRED. As we expected, Flow 2 is rejected in ProbeCast while QAMNet Flow 2 delivery ratio is around 17%. In QAMnet, Flow 1 is often non-admitted since its rate is 800 Kbps and channel bandwidth is only 2 Mbps. As a result, Flow 1 throughput is lower than ODMRP and ProbeCast Flow 1 throughput in Figure 7. Since Flow 2 and 3 are usually accepted by QAMNet, their throughputs are similar to ODMRP. Even though Flow 1 is a non-admitted flow, Flow 1 traffic rate additively increase since Flow 2 rate is low, 400 Kbps.

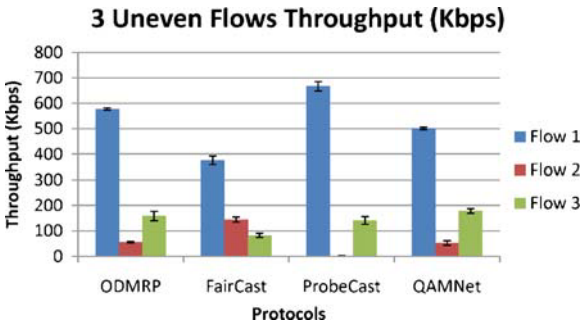

Figure 7. Throughput of uneven three inelastic flows case. Flow 1, 2, and 3 are 800 Kbps, 400 Kbps, and 200 Kbps, respectively.

Figure 8 represents total throughput in the network. In spite of the fact that individual throughputs are now proportional to demands, it appears that the total throughputs are rather insensitive to the actual distribution of demands. Like previous case, ODMRP, ProbeCast, and QAMNet total throughput is very close while FairCast loses total throughput to achieve fairness. End-to-end delays show the same pattern to the previous experiment, Figure 5, but Flow 1 delay is much higher than Flow 2 and 3 in all protocols since contention is very high (high queuing delay) due to high traffic rate, 800 Kbps.

5.4. Mobile three even unicast inelastic flows

In the previous experiments, we use a static topology. In this section, we evaluate experiment results when three flows in Figure 2 are moving. At the beginning of the simulation, flows are apart enough to remove interference and thus all flows are admitted. They start packet transmission at the same time, $T = 1$ s, with uniform rate = 500 Kbps and Flow 1 and 3 move toward Flow 2. Finally, they form Figure 2 at simulation time $T = 100$ s and stop moving.

When flows approach inside the radio range, FairCast starts proportional packet drop to achieve fairness among flows and ProbeCast starts admission control. ProbeCast rejects Flow 2 since it exceeds the drop threshold due to interference. Figure 9 presents Flow 1 and 3 achieve more than 450 Kbps throughput in ProbeCast and flow fairness in N-PROD is very high. The reason why ProbeCast through-

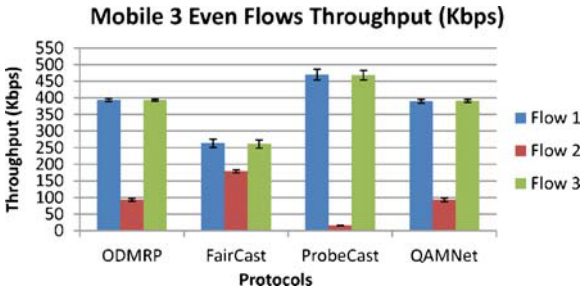

Figure 9. Throughput of three inelastic flows when they are moving. Uniform nominal rate = 500 Kbps.

put in Figure 9 is higher than the throughputs in Figure 3 is Flow 2 responds quickly to the network congestion. In Figure 3, Flow 2 starts transmission after Flow 1 and before Flow 3 starting transmission so that Flow 2 competes with Flow 1. However, in Figure 9, Flow 1 and 3 interfere in Flow 2 at the same time. When Flow 1 and 3 come inside the interference range, Flow 2 drop rate suddenly increases and ProbeCast rejects it exceeding the threshold. Therefore, Flow 1 and 2 throughputs increase in Figure 9 while Flow 2 throughput is lower than Figure 3. Contrarily, QAMNet records similar result to ODMRP without throughput gain. QAMNet has dynamic regulation mechanism to cope with false admission and available bandwidth fluctuation. For example, if more resources are actually consumed than reserved, a node randomly selects one of its inelastic flows and controls traffic rate of that flow. In our scenario, however, flows do not spend more resources than reserved so that no flow becomes falling back into a non-admitted flow. In our scenario, flows suffer from available bandwidth decreasing, but Reference [5] does not clearly describe this case. Consequently, all three flows are admitted and they compete capturing channel when they come inside the interference range. As a result, QAMNet throughput is almost same to ODMRP's.

In Figure 10, total throughputs illustrate the same pattern to previous results, Figure 4. ProbeCast and FairCast throughputs increase significantly, 70 Kbps and 110 Kbps, respectively. ODMRP throughput, however, increases only 10 Kbps and QAMNet throughput has no throughput gain.

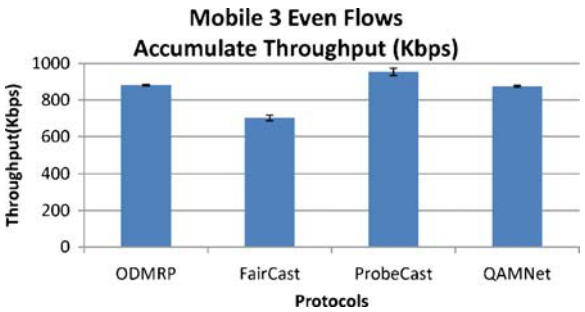

Figure 10. Accumulated total throughput in the network in three inelastic flows when they are moving. Uniform nominal rate = 500 Kbps.

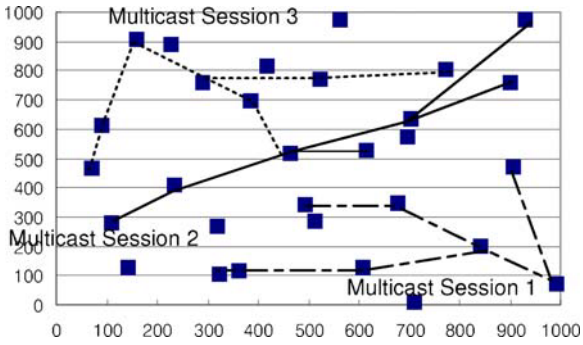

Figure 11. Three multicast sessions in the 1000 m by 1000 m area. Each session has one source and three members.

End-to-end delays of four protocols are around 20 ms so we skip the delay graph here.

5.5. ProbeCast in multicast scenario

In the previous experiment, the topology was simple and was specifically chosen to illustrate the difference between uniform and proportional dropping and the importance of the latter in the support of nonuniform inelastic flow. Moreover, the scenario was unicast.

In this section, we report on multicast experiments with ProbeCast. Figure 11 is the topology example we used. Thirty nodes are randomly distributed in a 1000 m by 1000 m area; three multicast sessions are established. Each multicast session has one source and three receivers and no common node belongs to two sessions. However, interference occurs at intermediate forwarding nodes in the field. Inelastic data rates are uniform for simplicity, namely, 500 Kbps for each flow. These sessions can tolerate up to 50% of packet loss (that is drop threshold is 50%). Multicast session 1 starts transmitting at $T = 1$ s and session 2 and 3 start at $T = 1$ s and 20 s, respectively. In Figure 12, we report the result for an experiment with only two multicast sessions (session 1 and 2). We performed several simulation runs changing seed numbers. In almost all the runs, both sessions survive and manage to transmit their

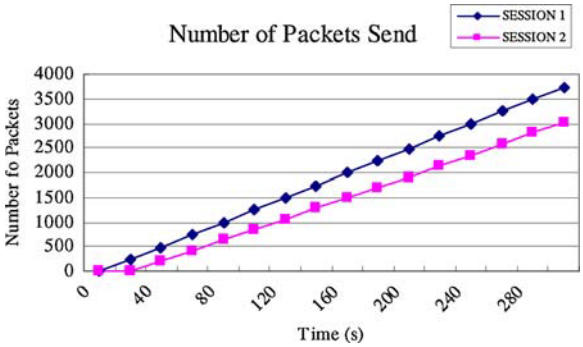

Figure 12. Two multicast sessions can be simultaneously supported.

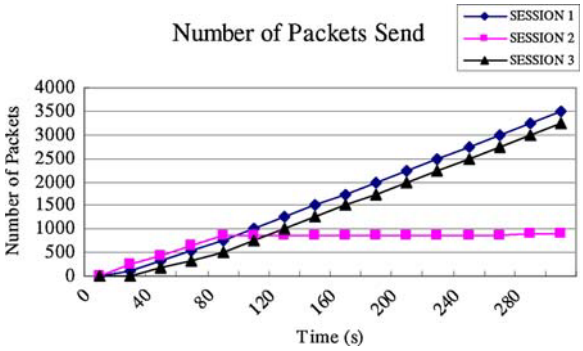

Figure 13. Three multicast sessions are present. Session 2 is rejected.

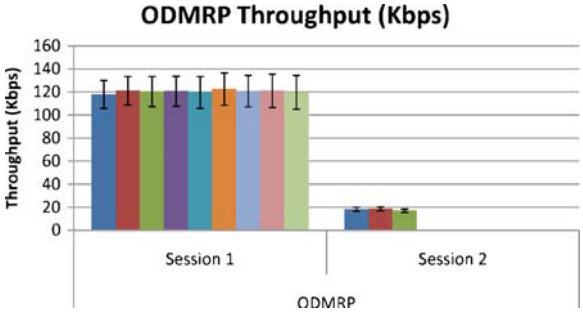

Figure 15. Throughput of ODMRP. Two multicast sessions: one is 200 Kbps video stream with nine receivers and the other is 40 Kbps audio stream with three receivers.

full rates. However, when all three sessions are injected, the results reported in Figure 13 show that one of three sessions is consistently rejected. At the beginning the sessions try to balance drop rates and partially succeed. However, as time progresses, this balance collapses. One session starts dropping packets in bursts and packet drop rate suddenly skyrockets, exceeding the threshold and triggering backpressure on the newcomer (with lower drop threshold).

5.6. Audio and video streams in a mobile scenario

In this experiment, 30 nodes are uniformly distributed in 1000 m by 1000 m area and nodes are moving based on random mobility model in which the maximum speed is 10 m/s with no pause time. Two inelastic flows are established: multicast session 1 is 200 Kbps video stream with nine receivers started at 0 s and session 2 is 40 Kbps audio stream with three receivers started at 10 s. The model is representative of a search and rescue operation with two teams communicating with audio and video means. Some of the members are motorized, others are on foot, justifying the random way point motion pattern. Figure 14 is a snapshot of the topology. An issue is the ability of the two flows, audio and video, to fairly compete and coexist. In particular, we are interested in verifying that ProbeCast can indeed exercise proper Acceptance Control and kill one of the streams (in this case the audio stream) when bandwidth is inadequate.

If nodes are static, channel capture is very serious in Reference [7]. However, in this scenario, nodes are moving so that unfairness is moderated. For example, in the static topology example discussed in Reference [7], multicast session 1 always captures the channel so that most packets in session 2 are dropped or lost. In our mobile scenario, the topology keeps changing and thus session 2 has a chance to deliver packets without serious contention. With ODMRP, in Figure 15, the audio stream delivery ratio is around 45% while video successfully transmits about 58% packets. Clearly this is unacceptable, audio should be improved or otherwise rejected. In the static experiment [7], ODMRP delivery ratio gap between audio and video is about 50% (session 1 is 80% and session 2 is 30%). In the mobile environment Figure 16, FairCast does slightly better, increases audio throughput up to 31 Kbps as well as video throughput to 125 Kbps. FairCast alleviates packet collision while forcing packet drop so that both multicast session throughput increases comparing to ODMRP throughput. Like we already observed in subsection 5.4, QAMNet performs the same as ODMRP, in Figure 17. QAMNet strategy to downgrade inelastic flows to elastic when resources are inadequate instead of rejecting them does deprive video of useful extra bandwidth. ProbeCast

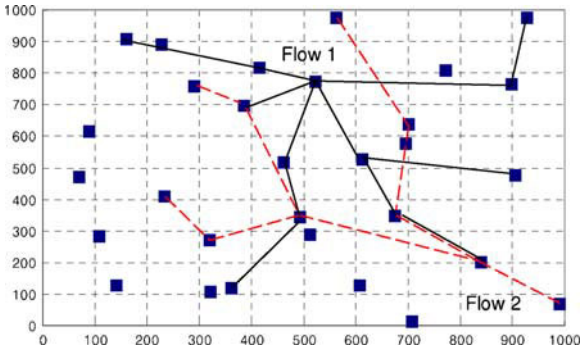

Figure 14. Topology example of 30 nodes and two multicast flows which are audio and video steaming.

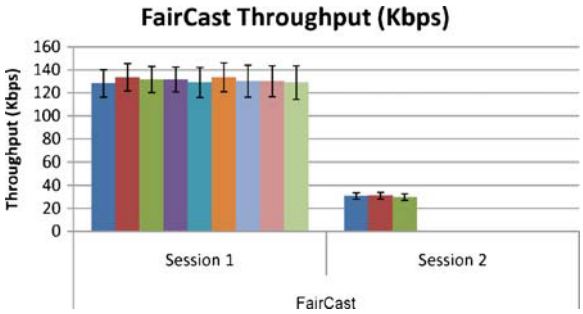

Figure 16. Throughput of FairCast. Two multicast sessions: one is 200 Kbps video stream with nine receivers and the other is 40 Kbps audio stream with three receivers.

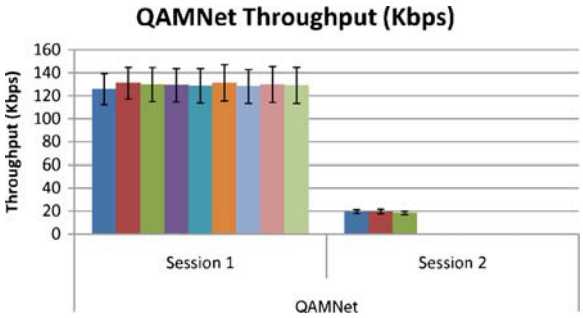

Figure 17. Throughput of QAMNet. Two multicast sessions: one is 200 Kbps video stream with nine receivers and the other is 40 Kbps audio stream with three receivers.

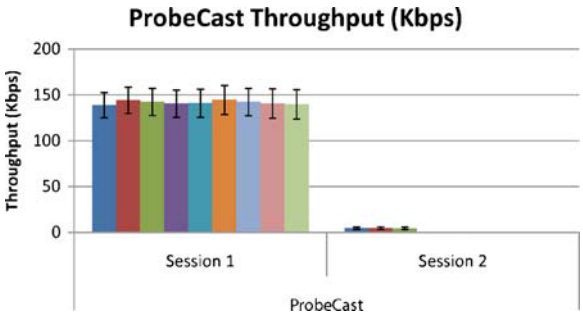

Figure 18. Throughput of ProbeCast. Two multicast sessions: one is 200 Kbps video stream with nine receivers and the other is 40 Kbps audio stream with three receivers.

does the right thing, considering the fact that audio cannot be guaranteed adequate bandwidth. It kills the audio while increasing video throughput to an impressive 140 Kbps, shown in Figure 18. This is exactly what we expect from an efficient Acceptance Control Scheme. The audio will resume after the video session is over.

5.7. Inelastic and elastic flow coexistence

The final experiment is about elastic and inelastic flow coexistence and ability of the inelastic flow to grab enough resources from the elastic flow to survive. We will see that ProbeCast enables an inelastic flow to grab bandwidth from an elastic flow (say TCP) by properly exercising the proportional drop threshold. Figure 19 represents a very

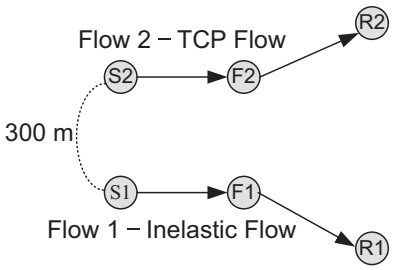

Figure 19. Two flows: Flow 1 is an inelastic flow and Flow 2 is an elastic flow.

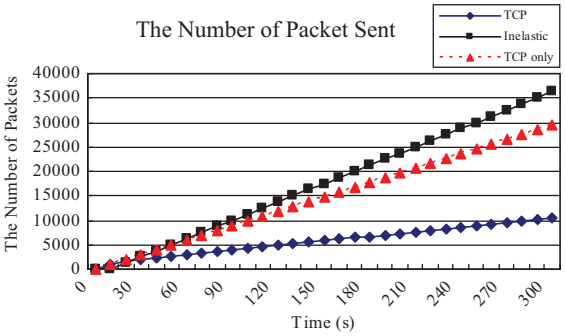

Figure 20. The number of packet sent by the inelastic and the elastic sources.

simple network topology where an inelastic video stream flow coexists with an elastic TCP flow. The TCP sender, S2, starts at $t = 1$ s and the inelastic flow sender, S1, starts at $t = 10$ s. Video stream rate is 500 Kbps. S1, S2, F1, and F2 are all within radio sensing range so that they interfere with each other but cannot decode each other transmissions. R1 and R2, however, are assumed far apart to reduce hidden terminal collisions (i.e., R1 is not interfered by F2 and vice versa). Note that the typical reservation based CAC scheme such as QAMNet does not work in this situation. When S1 monitors the channel for available bandwidth, it finds none. In fact, it cannot tell that the interferer is a lower priority best effort flow since distance exceeds reception range. On the other hand, the TCP flow (due to its greedy nature) completely fills the channel. Therefore, the inelastic flow is rejected. In contrast, ProbeCast lets the inelastic flow in, causing an increase in packet loss that in turn forces TCP to back off and leave enough room for the inelastic flow to achieve full rate. It is interesting to note that the TCP source will backoff and reduce the traffic rate even though it does not understand the packet header entry that ProbeCast adds (i.e., current packet drop rate). The interference and subsequent loss rate will suffice to slow down TCP.

Figure 20 shows the number of sent packet at flow sources and Figure 21 illustrates receiver's packet drop rate of the inelastic flow, R1. In Figure 20, the middle line (triangle markers) represents the number of packet sent by TCP when

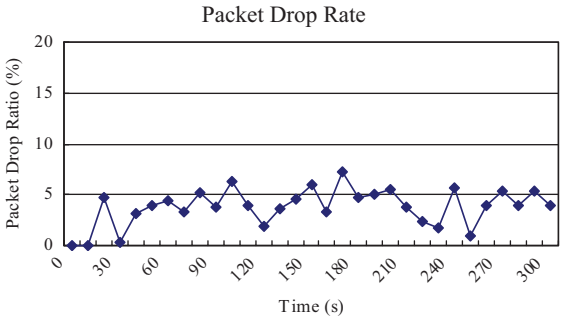

Figure 21. The packet drop ratio at the inelastic flow receiver in the two inelastic and elastic flow case.

it is alone. Comparing the two TCP lines (before and after) we notice a 3:1 reduction in TCP rate. Figure 21 shows an inelastic flow packet drop rate in the order of 5%. This loss is easily sustained as it is below the threshold and it is recovered by end to end erasure coding. It is the price to keep TCP at bay. The TCP flow experiences a comparable loss rate on the shared channel.

6. CONCLUSIONS AND FUTURE WORK

Inelastic multicast bandwidth allocation and CAC in MANETs is extremely challenging because of dynamic traffic and route changes and unreliable estimation of available bandwidth. In this paper, we have proposed a novel scheme called ProbeCast which supports efficient call admission control and QoS in the MANET without requiring bandwidth estimation and reservations. ProbeCast uses probing and backpressure mechanisms to accept feasible flows and reject the unfeasible ones. For backpressure to work, a congested link must be fairly (more precisely, proportionally) shared among inelastic contenders. We apply a distributed fairness scheme, N-PROD, inspired to an earlier TCP fairness scheme (NRED) and to the 'Distributed Gentlemen's Agreement' proposed in FairCast, to proportionally share bandwidth among flows. We compare ProbeCast with a traditional QoS multicast scheme, QAMNet and show via simulation that ProbeCast manages to reject flows in all situations when bandwidth is inadequate while QAMNet cannot decisively reject and wastes useful resources. Major contributions of ProbeCast are: the robust and efficient CAC in presence of mobility, the ability to handle both uni and multicast inelastic CAC, and the ability to manage the coexistence of elastic and inelastic flows.

REFERENCES

1. Bür K, Ersoy C. Multicast routing for ad hoc networks with a multiclass scheme for quality of service. In *19th International Symposium on Computer and Information Sciences (ISCIS)*, 2004; 187–197.
2. Bür K, Ersoy C. Multicast routing for ad hoc networks with a quality of service scheme for session efficiency. In *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC 2004)*, 2004; 1000–1004.
3. Chen Y-S, Ko Y-W. A lantern-tree-based qos on-demand multicast protocol for a wireless mobile ad hoc network(network). *IEICE Transactions on Communications*, **87**(3): 717–726, 20040301.
4. Saghir M, Wan TC, Budiarto R. Load balancing qos multicast routing protocol in mobile ad hoc networks. In *Proceedings of Asian Internet Engineering Conference (AINTEC 2005)*, 2005; 83–97.
5. Tebbe H, Kassler AJ, Ruiz PM. Qos-aware mesh construction to enhance multicast routing in mobile ad hoc networks. In *Proceedings of the 1st ACM International Conference on Integrated Internet Ad Hoc and Sensor Networks (InterSense '06)*, New York, USA, 2006; 17.
6. Borgonovo F, Capone A, Fratta L, Marchese M, Petrioli C. PCP: a bandwidth guaranteed transport service for ip networks. In *IEEE International Conference on Communications 1999 (ICC 99)*, Vol. 1, 1999; 671–675.
7. Marfia G, Lutterotti P, Eidenbenz SJ, Pau G, Gerla M. Faircast: fair multi-media straming in ad hoc networks through local congestion control. In *11th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile System*, 2008.
8. Lee S-B, Ahn G-S, Zhang X, Capbell AT. Insignia: an ip-based quality of service framework for mobile ad hoc networks. *Journal of Parallel and Distributed Computing* 2000; **60**(4): 374–406.
9. Ahn G-S, Campbell AT, Veres A, Sun L-H. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (SWAN). *IEEE Transactions on Mobile Computing* 2002; **1**(3): 192–207.
10. Xiao H, Chua K, Seah W, Lo A. A flexible quality of service model for mobile ad hoc networks. In *Proceedings of Vehicular Technology Conference (VTC)*, 2000; 445–449.
11. Lee SJ, Su W, Gerla M. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications* 2002; **7**(6): 441–453.
12. Pagani E, Rossi GP. A framework for the admission control of qos multicast traffic in mobile ad hoc networks. In *Proceedings of the 4th ACM International Workshop on Wireless Mobile Multimedia(WOWMOM '01)*, New York, USA, 2001; 2–11.
13. Xu K, Gerla M, Qi L, Shu Y. TCP unfairness in ad hoc wireless networks and a neighborhood red solution. *Wireless Network* 2005; **11**(4): 383–399.
14. Ólafsson S, Kim J. Simulation optimization: simulation optimization. In *WSC '02: Proceedings of the 34th Conference on Winter Simulation*, Winter Simulation Conference, 2002; 79–84.
15. Scalable Networs Inc. *QualNet*. Available at: http://www.scalble-networks.com