ProbeCast: MANET Admission Control via Probing

Soon Y. Oh, Gustavo Marfia, and Mario Gerla
Department of Computer Science, University of California, Los Angeles
Los Angeles, CA 90095,USA
soonoh@cs.ucla.edu, gmarfia@cs.ucla.edu, gerla@cs.ucla.edu

ABSTRACT

An inelastic flow is a flow with inelastic rate: i.e., the rate is fixed, 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 the source. The backpressure principle is simple; however its implementation requires some care to avoid unfairness and eventually 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 the same proportional drop rate among all flows competing 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.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication* ; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*

General Terms

Algorithm, Design, Performance

Keywords

MANET, Call Admission Control, QoS, Inelastic flow

1. INTRODUCTION

In emergency and tactical 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 [3, 4, 5, 11, 13]. 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 allocate 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 allocation 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 allocation. If an intermediate link fails to meet the QoS requirements, the flow is "pushed back" by sending a backpressure message upstream to the source - 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.

The backpressure 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, a 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 drop threshold. If the new flow "does not fit" in the bottleneck, i.e., it causes congestion, than it 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. We can easily discriminate and backpressure the new flow.

A similar CAC approach was proposed several years ago for Internet VoIP streams [2]. 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 are independently adjusting 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 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 reject 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 appears in MSWIM 2008. This paper extends that work by designing 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 CAC schemes.

The rest of the paper is organized as followed: Section II illustrates the related work; Section III describes the details of the ProbeCast; Section IV presents simulation results. Conclusion and future work will be on Section V.

2. RELATED WORK

A number of ad hoc *unicast* QoS support protocols and algorithms have been proposed in the literature, e.g., INSIGNIA [7], SWAN [1], and FQMM [6]. However, relatively few MANET *multicast* QoS schemes have been introduced. We classify existing MANET multicast QoS schemes into 3 categories based on their resource measurement and reservation methods.

2.1 Bandwidth Estimation and Resource Reservation

Ad hoc QoS Multicasting (AQM) [3, 4] 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 (LTM) [5] 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) [11] is an on-demand mesh based protocol that uses a forwarding mesh like On-Demand Multicast Routing Protocol (ODMRP) [8]. 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

Multicast- Call Admission Protocol (M-CAMP) [10] 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 [2] M-CAMP employs 3 priority levels among packets: *real time, probe, 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 [13] 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 main drawback of both 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 [9]. The main focus of FairCast is fair sharing across multicast flows. FairCast does not exercise Call Admission Control, i.e., it cannot reject flows when congestion sets up. To cope with this situation, FairCast assumes that inelastic flows have sufficient erasure coding redundancy so that they can tolerate even substantial losses. Alternatively, the inelastic flows can be adaptively rate adjusted. FairCast uses only local flow interactions and packet dropping to achieve fairness; no end-to-end feedback. FairCast first introduced the concept of distributed "proportional fairness" in wireless channels. Flows locally interact, exchanging (piggybacked) information on packet loss rates and selectively drop packets to equalize their performance. FairCast differs from ProbeCast in that it does not address Call Admission Control.

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
    $D_i^r(t) = \frac{DqN_i^r(t)+DlN_i^r(t)}{RN_i^r(t)+DlN_i^r(t)}$
    $P_i^r(t) = \alpha P_i^r(t-1) + (1-\alpha)D_i^r(t)$
    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. 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 (eg, 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 it drops 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:

$$DN_i^r(t) = DqN_i^r(t) + DlN_i^r(t) \tag{1}$$

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

$$P_i^r(t) = \alpha P_i^r(t-1) + (1-\alpha)D_i^r(t) \tag{3}$$

where:

- t is the t-th 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 2 N-PROD Algorithm

```
DEFINITIONS: $D_i^r$ is packet drop rate 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 $r$. $pkt^r$ 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**
```

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 [14] 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 100Kbps and 60Kbps respectively and N-PROD control stabilizes at 20% drop probability, flows A and B throughputs stabilize at 80Kbps and 48Kbps respectively. In contrast, TCP fairness strives to equalize flows.

In our implementation 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.

When the packet drop rate is over the threshold, the flow is backpressured towards 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 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. SIMULATIONS

In this section, we validate N-PROD and ProbeCast using Qualnet v3.9.5 [12], a packet level network simulator. We implement N-PROD and ProbeCast in Qualnet and compare the performance of ProbeCast to pure ODMRP.

4.1 Simulation Setup

We use 802.11b with 376m effective reception range and 2Mbps channel capacity. The packet size is 512 bytes and the maximum queue size is 50Kbyte (about 100 packets). We use Qualnet default values for MAC and Physical layer configuration parameters.

ProbeCast and N-PROD 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 rebroad-

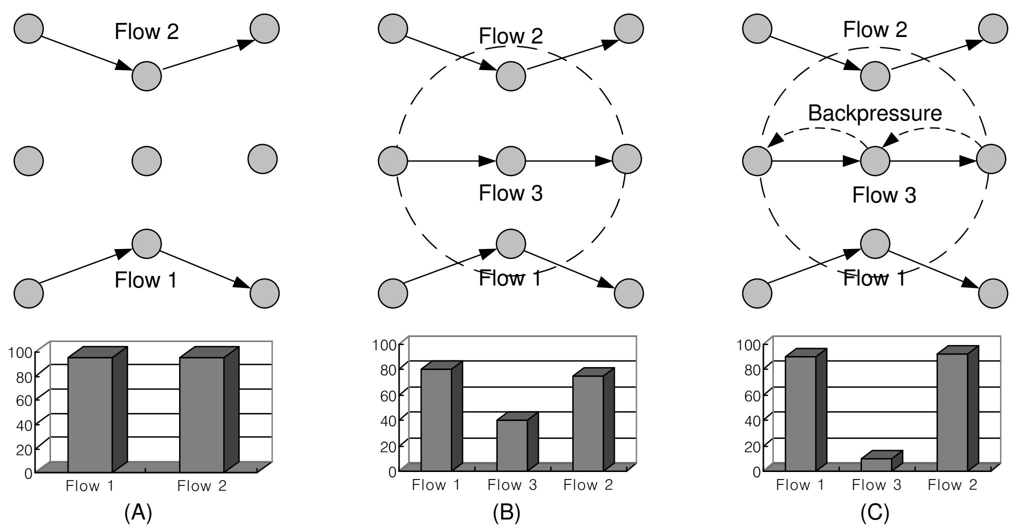


Figure 1: 3 flows in the simple topology. Lower graphs show packet delivery ratios, presented by percentage. (A) Two flows are present both with have high delivery ratios over 90%. (B) Flow 3 starts transmitting and other flows' delivery ratios decrease because of channel contention. (C) Flow 3 packet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

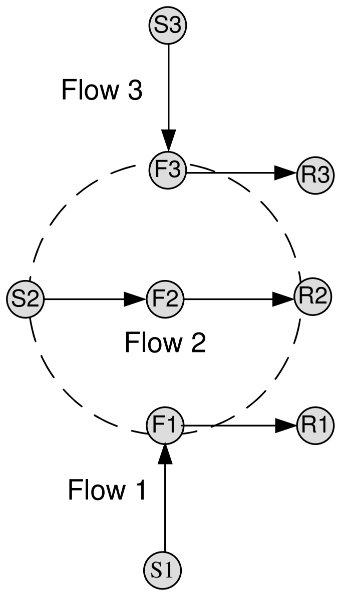


Figure 2: 3 parallel inelastic flows topology example. Intermediate nodes, F1, F2 and F3 are within radio range and they compete with each other. Sources, S1, S2, and S3 are outside of other's radio range.

casts 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 seconds and the forwarder life time is 3 times the Join Query refresh interval.

We use three metrics: Throughput is the total received data bits divided by the total simulation time; Packet Delivery Ratio is the fraction of received data; 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.

4.2 3 Unicast Inelastic Flows - No Backpressure

We first tested N-PROD ability to enforce proportional unfairness. In the process, we also show the difference between uniform and proportional fairness. The scenario is a unicast traffic scenario shown in figure 2. The 3 inelastic flows in figure 2 use disjoint paths, but they still interfere with each other. For each flow, the source and destination are located out of each others' transmission range and communicate only with intermediate nodes, more precisely node F1, F2 and F3. The distances between node F1, F2 and F3 are 350m 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 = 500Kbps. Flow 1 starts transmitting data 1 second after simulation initialization. Flow 2 and 3 start data sending T =10 second and 20 second 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. The result is shown in Figure 3. The

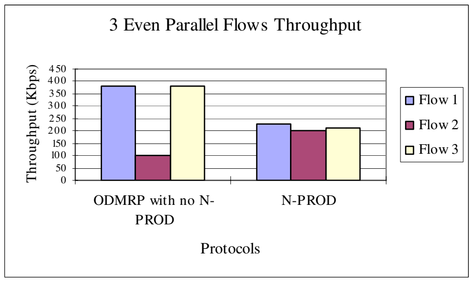


Figure 3: Throughput of 3 inelastic flows. Uniform nominal rate = 500Kbps.

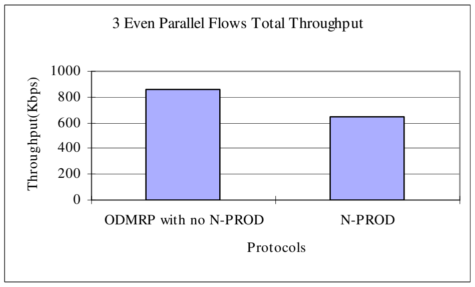


Figure 4: Total throughput in the network in 3 inelastic flows case. Data rate = 500Kbps.

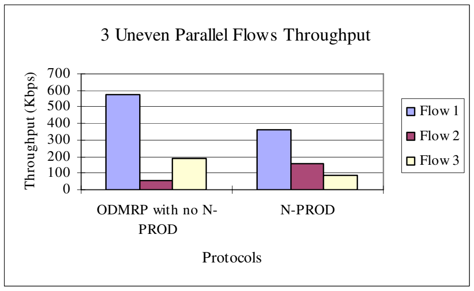


Figure 5: Throughput of uneven 3 inelastic flows case. Flow 1, 2, and 3 are 800Kbps, 400Kbps, and 200Kbps respectively.

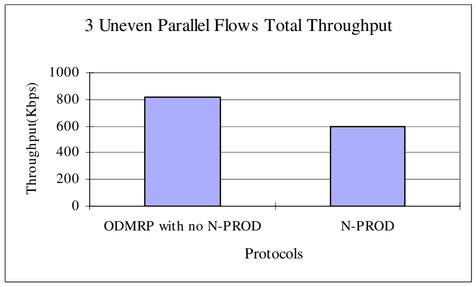


Figure 6: Total throughput in the network of uneven 3 inelastic flows case. Flow 1, 2, and 3 are 800Kbps, 400Kbps, and 200Kbps respectively.

application of N-PROD restores fairness as shown in Figure 3. This result is very similar to the result reported in paper [14]. This is not surprising since with uniform inelastic rates, N-PROD is equivalent to NRED.

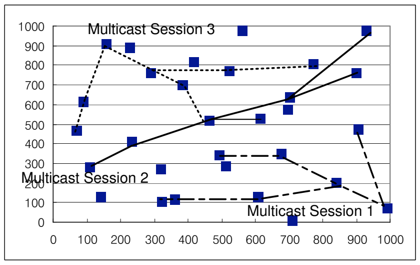
Figur 4 shows total throughput respectively. As already noted in [14], fairness comes at the cost of degraded total throughput.

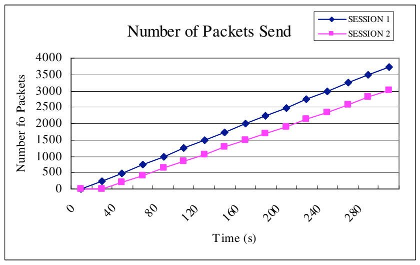
In the next simulation experiment, we use the same layout but now the inelastic flows send data packets at different rates. Namely, Flow 1 = 800Kbps and Flow 2 = 400Kbps and Flow 3 = 200Kbps. Like in the previous experiment, Flow 1 starts transmitting data at T =1 second. Flow 2 and 3 start transmissions at T = 10 second and T=20 second respectively. Without N-PROD, Flow 1 and 3 capture the channel and Flow 2 is starved. Flow 3 delivers more than 90% of the sent packets while only 10% of packets are reached at the destination in Flow 2. Consequently, Flow 3 throughput, about 180Kbps, is higher than Flow 2 throughput, 55Kbps, while S2 sends packets with higher rate than S3. With N-PROD 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 5. This result is clearly different from what could be achieved with NRED. The figure 6 represents total throughput in the network without and with N-PROD. 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.

4.3 More Realistic 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 non uniform inelastic flow. Moreover, the scenario was unicast, and no backpressure was enacted.

In this section, we report on multicast experiments with ProbeCast, this time combining N-PROD, backpressure and CAC. Figure 7 is the topology example we used. 30 nodes are randomly distributed in a 1000m by 1000m area; 3 multicast sessions are established. Each multicast session has one source and 3 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, 500Kbps 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 second and session 2 and 3 start at T =10 and 20 second, respectively. In figure 8, we report the result for an experiment with only two multicast sessions (session 1 and 2). We performed several simulation run changing seed numbers. In almost all the runs, both sessions survive and manage to transmit their full rates. However, when all three sessions are injected, the results reported in figure 9 shows 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 thresh-





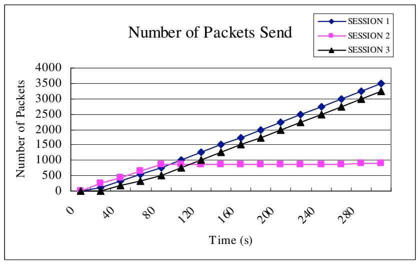


Figure 7: 3 multicast sessions in the 1000m by 1000m area. Each session has 1 source and 3 members.

Figure 8: 2 multicast sessions can be simultaneously supported.

Figure 9: 3 multicast sessions are present. Session 2 is rejected.

old and triggering backpressure on the newcomer (with lower drop threshold).

4.4 Inelastic Flow vs. Elastic Flow

The next experiment is designed to show that probing allows an inelastic flow to preempt an elastic flow (say TCP) by properly exercising the proportional drop threshold. Figure 10 represents a very simple network topology where an elastic video stream flow coexists with an inelastic TCP flow. The TCP sender, S2, starts at t =1 second; the inelastic sender, S1, starts at t=10 second. Video stream rate is 500Kbps. 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 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 even if it does not hear the inelastic N-PROD signals (i.e., current drop rate). The interference and subsequent loss rate will suffice to slow down TCP.

Figure 11 shows the number of sent packet at flow sources and figure 12 illustrates receiver's packet drop rate of the inelastic flow, R1. In figure 11, the middle line (triangle markers) represents the number of packet sent by TCP when it is alone. Comparing the two TCP lines (before and after) we notice a 3:1 degradation in TCP rate. Figure 12 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. The inelastic loss is in necessary to keep TCP at bay. The TCP flow in fact experiences a comparable loss rate on the shared channel.

5. CONCLUSIONS AND FUTURE WORKS

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 scheme called ProbeCast

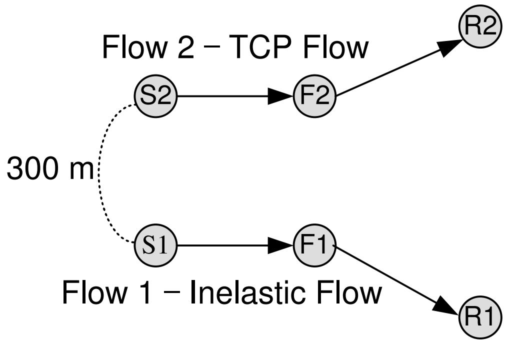


Figure 10: Two flows. Flow 1 is an inelastic flow and Flow 2 is an elastic flow.

which supports efficient call admission control and QoS in the MANET without requiring bandwidth estimation and reservations. ProbeCast uses probe 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. Major contributions of ProbeCast are: the robust and efficient CAC mechanism based on probing; the ability to handle both uni and multicast inelastic CAC, and; the ability to handle both inelastic and elastic flows at the same time. Simulation results show confirm our claims. Future work will examine the use of QoS multicast tree construction heuristics to facilitate rerouting in case of CAC blocking.

Acknowledgment

This research is supported through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-06-3-0001, and; by ARMY MURI under funding W911NF0510246.

6. REFERENCES

[1] G.-S. Ahn, A. T. Campbell, A. Veres, and L.-H. Sun. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks

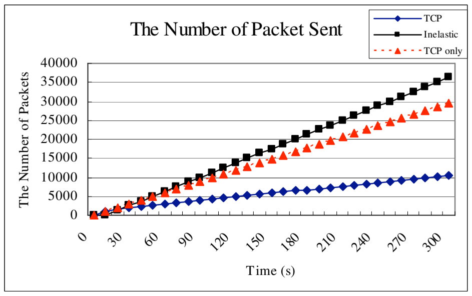


Figure 11: The number of packet sent by the inelastic and the elastic sources.

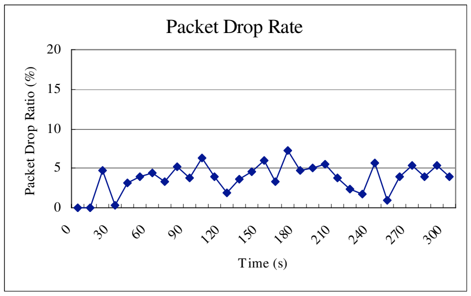


Figure 12: The packet drop ratio at the inelastic flow receiver in the two inelastic and elastic flow case.

(swan). *IEEE Transactions on Mobile Computing*, 1(3):192–207, 2002.

[2] F. Borgonovo, A. Capone, L. Fratta, M. Marchese, and C. Petrioli. Pcp: A bandwidth guaranteed transport service for ip networks. In *IEEE International Conference on Communications 1999 (ICC 99)*, volume 1, pages 671–675, 1999.

[3] K. Bür and C. Ersoy. Multicast routing for ad hoc networks with a multiclass scheme for quality of service. In *19th International Symposium on Computer and Information Sciences (ISCIS)*, pages 187–197, 2004.

[4] K. Bür and C. Ersoy. 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)*, pages 1000–1004, 2004.

[5] Y.-S. CHEN and Y.-W. KO. 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.

[6] H.Xiao, K.chua, W. Seah, and A. Lo. A flexible quality of service model for mobile ad-hoc networks. In *Proceedings of Vehicular Technology Conference (VTC)*, pages 445–449, 2000.

[7] S.-B. Lee, G.-S. Ahn, X. Zhang, and A. T. Capbell. Insignia: an ip-based quality of service framework for mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 60(4):374–406, 2000.

[8] S. J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications*, 7(6):441–453, 2002.

[9] G. Marfia, P. Lutterotti, S. J. Eidenbenz, G. Pau, and M. Gerla. 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.

[10] E. Pagani and G. P. Rossi. 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)*, pages 2–11, New York, NY, USA, 2001. ACM.

[11] M. Saghir, T. C. Wan, and R. Budiarto. Load balancing qos multicast routing protocol in mobile ad hoc networks. In *Proceedings of Asian Internet Engineering Conference (AINTEC 2005)*, pages 83–97, 2005.

[12] Scalable Networs Inc. *QualNet*. `http://www.scalble-networks.com`.

[13] H. Tebbe, A. J. Kassler, and P. M. Ruiz. Qos-aware mesh construction to enhance multicast routing in mobile ad hoc networks. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks (InterSense '06)*, page 17, New York, NY, USA, 2006. ACM.

[14] K. Xu, M. Gerla, L. Qi, and Y. Shu. Tcp unfairness in ad hoc wireless networks and a neighborhood red solution. *Wirel. Netw.*, 11(4):383–399, 2005.