Joint Scheduling and Congestion Control in Mobile Ad-Hoc Networks

Umut Akyol
WINLAB, Rutgers University NJ
Email: umakyol@yahoo.com

Matthew Andrews
Bell Labs, Murray Hill NJ
Email: andrews@research.bell-labs.com

Piyush Gupta
Bell Labs, Murray Hill NJ
Email: pgupta@research.bell-labs.com

John Hobby
Bell Labs, Murray Hill NJ
Email: hobby@research.bell-labs.com

Iraj Saniee
Bell Labs, Murray Hill NJ
Email: iis@research.bell-labs.com

Alexander Stolyar
Bell Labs, Murray Hill NJ
Email: stolyar@research.bell-labs.com

***Abstract*— In this paper we study the problem of jointly performing scheduling and congestion control in mobile ad-hoc networks so that network queues remain bounded and the resulting flow rates satisfy an associated network utility maximization problem. In recent years a number of papers have presented theoretical solutions to this problem that are based on combining differential-backlog scheduling algorithms with utility-based congestion control. However, this work typically does not address a number of issues such as how signaling should be performed and how the new algorithms interact with other wireless protocols.**

In this paper we address such issues. In particular:

- **We define a specific network utility maximization problem that we believe is appropriate for mobile adhoc networks.**
- **We describe a *wireless Greedy Primal Dual* (wGPD) algorithm for combined congestion control and scheduling that aims to solve this problem.**
- **We show how the wGPD algorithm and its associated signaling can be implemented in practice with minimal disruption to existing wireless protocols.**
- **We show via OPNET simulation that wGPD significantly outperforms standard protocols such as 802.11 operating in conjunction with TCP.**

This work was supported by the DARPA CBMANET program.

I. Introduction

In this paper we consider the practical design of joint congestion control and MAC/scheduling algorithms for mobile ad-hoc networks. Consider a set of traffic flows within such a network. The congestion control problem involves deciding which packets should be injected into the flows in each time step. The MAC/scheduling algorithms determine what is the next packet to be transmitted by a wireless node and also resolve contention between sets of interfering nodes.

We begin by discussing the background to our work, including an overview of the more traditional separate approaches to congestion control and scheduling, a discussion of the more recent theoretical work that looks at these problems in combination and a description of a number of issues that were not directly addressed by this work but which are important to any practical realization of these schemes. The aim of our work is to define practical protocols for combined congestion control and scheduling in mobile ad-hoc networks that try to satisfy theoretical properties such as optimizing a suitably defined notion of aggregate network utility.

A. Background

Much of the traditional literature on congestion control and scheduling in communication networks has treated the two problems separately. A large body of work on congestion control was initiated by Kelly, Maulloo and Tan [15] who showed that TCP-like algorithms for congestion control in the Internet can be viewed as primal-dual algorithms for solving an associated network utility maximization problem. In particular, consider a set of flows passing through a set of servers in a wireline network. Let c_e be the capacity of server e and let S_e be the set of flows passing through server e. In addition, let x_f be the current injection rate into flow f and let $U_f(\cdot)$ be a utility function that represents the "benefit to the system" achieved by a given flow rate. The Kelly et al. work aims to solve the following *Network Utility Maximization* (NUM) problem,

$$\max \qquad \sum_f U_f(x_f)$$
$$\text{subject to}$$
$$\sum_{f\in S_e} x_f \quad \leq \quad c_e.$$

It was shown in [15] that by adjusting the injection rates x_f according to the extent to which constraints are violated, it is possible to obtain rates that optimize the above problem. This work has been extended in number of different ways. (See e.g. [20], [21], [30], [18].) In particular, Chiang [9] presented an extension for wireless networks in which injection rates and transmitter power settings are chosen so that aggregate utility is optimized subject to the injection rates lying in the capacity region of the network. For wireless networks this capacity region is typically a complex region involving the interference relationships between the nodes.

However, one property of the papers [15], [9] and related work is that they do not directly address the scheduling of the packets in the network. For example, in the Kelly et al. NUM problem described above, the network is considered to be underloaded as long as all of the capacity constraints are satisfied. In particular there is no explicit consideration of how the data is scheduled so that all packets reach their destinations. This issue can be important. In [5] an example was presented in which congestion control mechanisms similar to those in [15] were used. However, due to adverse interactions with the network schedulers, the injection rates did not converge to the optimal solution of the NUM.

The problem of scheduling in multi-hop wireless networks has also been extensively addressed in the literature. The seminal work of Tassiulas and Ephremides [26], [27] analyzed an algorithm (that we shall henceforth refer to as *Max-Weight*) that always tries to move data from large queues to small queues. They proved that for traffic flows with fixed injection patterns, if there exists an algorithm that is able to schedule all the data then the Max-Weight algorithm will do this.

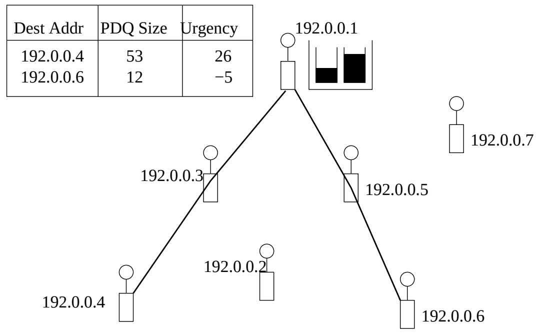
As already mentioned, if the congestion control and scheduling are performed in isolation then adverse effects can occur. In recent years a number of papers [24], [11], [23], [18] have addressed this issue by considering algorithms for *jointly* combining congestion control and scheduling. In particular, each of these algorithms defines a network control that determines when packets are injected into flows and which packets are transmitted by the nodes during each time interval. They aim to do this so as to maximize an aggregate utility function,

$$\sum_f U_f(x_f).$$

Our approach is based on the solution from the paper [24] that was called the Greedy Primal-Dual algorithm (GPD). In the version of GPD that we shall be considering, each node maintains a set of *per-destination* queues (PDQs). Then, according to the results of [24], this algorithm provides a control that solves the above optimization problem. However, there were a number of issues raised by the scheme of [24] that need to be addressed before it can be fully realized as a scheme for the control of mobile ad-hoc networks.

- The GPD algorithm requires exchange of PDQ length information between neighboring nodes. However, it does not specify the exact method for this information exchange.
- The interactions between the congestion-control mechanism of GPD and a reliability scheme (such as that provided by the TCP sequence-acknowledgment mechanism) were not explored.
- The "pure" form of the GPD scheme addresses the contention-resolution problem between interfering nodes as a centralized problem. In order to implement GPD in practice we must convert it to a form in which standard distributed techniques for contention-resolution can be applied.

The aim of our paper is to describe a practical version of GPD that addresses these issues and can be applied to wireless networks. We refer to the resulting scheme as the wireless Greedy-Primal-Dual (wGPD) algorithm. We begin by first defining the model of wireless ad-hoc networks that we consider in this paper and then describing the theoretical GPD algorithm in detail.



Dest Addr	PDQ Size	Urgency
192.0.0.4	53	26
192.0.0.6	12	−5

Fig. 1. A mobile ad-hoc network with two flows. The PDQs at one node are shown together with their associated sizes and urgency weights.

B. The Model

We consider a set of mobile nodes in an ad-hoc network. (See Figure 1.) We focus on the simplest case in which each node has a single radio and all communication is confined to a common channel. (This is the situation that is most often considered in studies of the standard 802.11 wireless protocols.) We assume that there are a fixed set of flows in the network. Each flow is classified as inelastic, elastic or semi-elastic, depending on its ability to adapt to different available bandwidths in the network. Inelastic flows (such as voice) have a fixed bandwidth requirement whereas elastic flows (such as web browsing or file transfers) can adapt to any level of assigned bandwidth. We use the term semi-elastic to refer to flows (such as streaming video with layered encoding) that have a minimum rate requirement but can also adapt to a higher bandwidth if it is available. Each elastic or semi-elastic flow f has a concave utility function $U_f(x)$ that represents the "benefit to the system" that is obtained when flow f is assigned bandwidth x. In our experimental results we shall often use the utility function $U_f(x) = \log x$. This choice of function is motivated by the paper [3] which showed that (up to some saturation point), human satisfaction with web browsing is proportional to the logarithm of the assigned bandwidth.

We assume throughout this work that the bandwidth requirements for inelastic flows and semi-elastic flows are feasible at all times. If this is not the case then an admission control scheme is required. The integration of wGPD with admission control is a topic for future work. In addition, many inelastic flows have end-to-end delay requirements. We assume that these are supported by giving each inelastic flow a minimum bandwidth requirement that is slightly higher than its injection rate. Explicit incorporation of end-to-end delay bounds into wGPD is left for future work.

Other simplifying assumptions that we make in this work are that transmission powers are fixed and routes are specified by some external protocol (e.g. OLSR or AODV). The extension of our approach to encompass power control and adaptive routing is deferred to future work.

The capacity of the network is defined by the *schedulable region*. This consists of the set of transmission rates that are achievable without violating any of the system assumptions. More formally, let L be the set of ordered tuples (i, j, r_{ij}) that represent node i transmitting to node j at rate r_{ij}. We say that a subset $S \subseteq L$ is *schedulable* (or, is within the schedulable region) at time t if for all $(i, j, r_{ij}) \in S$ node i can send data to node j at rate r_{ij} and these transmissions occur simultaneously. (This naturally implies that any subset of a schedulable set is also schedulable.) The capacity region of the system is then defined to be the set of flow rates that can be supported by the network by using only schedulable subsets of transmissions at any given time and without traffic queues in the network "blowing up" over time. Note that the capacity region is typically not known to any element in the system. Moreover it can change over time as nodes move.

With the above definitions in place, our goal becomes to solve the following optimization problem at all times:

$$\max \qquad \sum_f U_f(x_f)$$

subject to

minimum rate requirements are met for inelastic and semielastic flows

injections rates lie in system capacity region.

C. Description of the GPD algorithm

We now describe the theoretical Greedy Primal-Dual algorithm for combined congestion control and scheduling that was introduced in [24] and was shown to solve the above optimization problem. The focus of our paper is to demonstrate how this algorithm can be transformed into practical algorithms for wireless ad-hoc networks. We consider the GPD approach in conjunction with the key data-structure that we refer to as a per-destination queue (PDQ). (See Figure 1.) The per-destination queue for destination d at node i, denoted Q^i_d, stores all the packets at node i that have address d as their destination. Let q^i_d be the amount of data in this queue. Let $n(i, d)$ be the next hop for destination-d bound traffic after it leaves node i. Note that since we are focusing on unicast flows in which routes are specified, this notion is well-defined. Each PDQ has an associated concept of urgency weight. The urgency weight for queue Q^i_d is denoted by w^i_d and is defined by $w^i_d = [q^i_d - q^{n(i,d)}_d] r_{i,n(i,d)}$, i.e. the urgency weight is set to be the difference of the PDQ size at the current node minus the associated PDQ size at the downstream node, multiplied by the channel rate between these nodes. (This urgency weight is essentially the scheduling weight used by the Max-Weight algorithm of Tassiulas-Ephremides [26], [27] and its definition is why scheduling schemes that are based on it are often known as differential backlog scheduling schemes.)

In the theoretical definition of the GPD algorithm time is slotted. The scheduling component is as follows. In each time slot we schedule the transmissions from the schedulable set $S \subseteq L$ that maximizes $\sum_{(i,d)\in S} w^i_d$. The congestion control component of GPD is also simple. We use x_f to denote an exponentially filtered average of the injection rate into flow f. If flow f is an elastic flow with source and destination nodes s_f and d_f then in each time step it injects a packet if and only if $U'_f(x_f) - \beta q^{s_f}_{d_f} > 0$, where $\beta > 0$ is some (small) parameter and $U'_f(\cdot)$ is the first derivative of $U_f(\cdot)$. In the case that flow f is semi-elastic (and hence has a minimum rate requirement R^{min}_f) the congestion control component injects a packet whenever $g(\beta K_f) + U'(x_f) - \beta q^{s_f}_{d_f} > 0$, where $g(\cdot)$ is some (sharply) increasing function and K_f is a *token counter* that keeps track of whether or not flow f is meeting its minimum rate requirement. In particular, K_f receives tokens at rate R^{min}_f at all times. Whenever a packet of size ℓ_p is injected into flow f, K_f is decremented by an amount ℓ_p (but is never allowed to drop to below zero). A similar scheme was discussed in [4] for the provision of minimum rate guarantees in cellular networks.

The main theoretical result of [24] is that as β approaches zero, the flow rates produced by the GPD algorithm approach the optimal solution of the above optimization problem.

In the remainder of the paper we discuss some of the implementation details that arise when implementing the GPD algorithm in a real wireless system. We call the resulting algorithm wGPD for wireless-GPD. We then present a number of simulation results that compare the performance of wGPD with the standard 802.11 protocols operating in conjunction with TCP congestion control.

The main issues that need to be addressed when converting the theoretical GPD algorithm into a practical wireless algorithm are:

- The urgency weight calculation requires that each node knows the PDQ sizes at its neighbors. We need a mechanism whereby the nodes exchange PDQ information among themselves.
- The congestion control mechanism of GPD specifies how the flows regulate their injections into the network in such a way that the utility maximization problem is solved. However, in standard congestion control protocols such as TCP, the congestion control is combined with the reliability mechanism in which packets that are lost or time out are retransmitted. We require that the congestion control algorithm of GPD works with a similar reliability mechanism so that we can obtain end-to-end reliability for flows.
- We need to perform the scheduling component of GPD in a distributed fashion. In particular, we need to resolve contention among the nodes in a distributed manner. This is of course a problem for any decentralized wireless system. However, we need to do this in such a way that we maintain (to the extent possible) the optimality properties of GPD.

Incremental design: One other feature that we would like our wireless protocols to satisfy is that they reuse as many features as possible from existing wireless protocols. There are many protocols in existence for wireless networks that are proven to be robust and we try to use as many of these features as possible while still being consistent with the GPD framework. For example, we are able to reuse the notions of Request-to-Send/Clear-to-Send (RTS/CTS) and the backoff mechanism from the 802.11 contention resolution protocols. For the congestion control, we are able to reuse the sequencing and acknowledgment mechanisms from TCP. However, we stress that we do not reuse the congestion control algorithm of TCP.

Scalability issues: Since we use the PDQ data structure, each node has to potentially maintain a separate queue for every possible destination in the network. We believe that this is feasible for mobile ad-hoc networks of moderate size (say up to ~ 100 nodes). In future work we plan to explore efficient queue aggregation methods for larger networks which would allow us to drastically reduce the number of queues while preserving most of the benefits of the "pure" PDQ structure studied in this paper.

D. Related work

As already mentioned, there has been a lot of prior work that considers scheduling or congestion control in wireless networks in isolation. Early work on scheduling includes the well-known papers of Tassiulas and Ephremides [26], [27] who showed that scheduling schemes based on queue sizes can maintain queue stability for inelastic traffic whenever this is feasible. (See also Awerbuch and Leighton [6].) More recent research has looked at scheduling issues in single-hop cellular-type networks. One difference between cellular networks and ad-hoc networks is that the simpler nature of the topology means that it is often easier to schedule according to detailed channel feedback statistics. One area of interest has been providing "Proportional Fair" bandwidth allocations for elastic traffic among a set of mobile users. This corresponds to maximizing the aggregate logarithm of the assigned flow rates. An algorithm for achieving this is known as "Proportional Fair" and was first presented in [28]. Papers that derive theoretical properties of the Proportional Fair algorithm include [25], [16], [1]. An extension of Proportional Fair that applies to semi-elastic traffic with minimum rate requirements was presented in [4]. Other papers that discuss the provision of Quality-of-Service in wireless data networks include [19].

The literature on the theoretical analysis of congestion control schemes is also large. As discussed earlier, Kelly, Maulloo and Tan [15] initiated a body of work that analyzes congestion control algorithms such as TCP from the perspective of primal-dual algorithms that aim to solve an associated network utility maximization problem. This work was extended in numerous papers (e.g. [20], [21], [30], [18]). Specific extensions that considered the capacity region of wireless networks include [9]. One feature of much of this work is that the flows are assumed to be permanently present in the network. More recent papers (e.g. [10], [17] and references therein) look at a model in which flows with a finite amount of data to transmit arrive in and depart from the system. In this case the aim is to keep the number of active flows stochastically bounded whenever possible. Another topic that has recently gained increasing interest is methods for achieving optimal network utility when network coding is permitted. (See e.g. [8].)

As we have already mentioned, most of the prior work on network utility maximization does not explicitly take scheduling dynamics into account. The goal was simply to calculate a set of flow rates that lie in the capacity region without directly worrying about how to schedule the data so as to achieve these rates. As shown in [5] the fact that a set of rates is achievable does not necessarily mean that they are actually achieved if the wrong scheduling is used. This issue was addressed by a set of recent papers [24], [11], [23], [18] that provide methods for joint congestion control and scheduling with the goal of optimizing aggregate network utility. The paper [24] introduced the GPD algorithm that is the motivation for the wGPD protocols that we present in this paper.

One problem with many of the theoretical scheduling algorithms in wireless ad-hoc networks is that in their pure form they require some central control to determine which set of nodes should transmit at each time step. Recent work has therefore looked at distributed schemes for resolving contention among nodes so that we can remain close to the boundaries of the capacity region and also take network utilities into account. Algorithms of this type are presented in [14], [29], [13]. In particular, the results of [13] will motivate our techniques for contention resolution that we present in Section IV.

II. Overview of 802.11 scheduling

The wGPD algorithm will reuse as many components as possible from the standard 802.11 protocol. For this reason we briefly give an overview of how scheduling is performed in 802.11.

In the most basic form of 802.11, each node has a single FIFO queue. The next packet that a node transmits is always the packet at the head of the FIFO queue. Contention between interfering nodes is resolved in the following manner. Each node has a value known as its current contention window. Whenever a transmission needs to be made the node initiates a backoff counter to a random value from within the contention window. It then starts to decrement this backoff counter, pausing it whenever the channel goes busy. When the counter reaches zero the node transmits the packet. If the transmission is unsuccessful the contention window is doubled and the procedure is repeated.

One well-studied problem that can occur in ad-hoc networks is the "hidden node" problem in which we have two transmissions that interfere at one of the intended receivers, even though the two transmitters cannot hear each other. This problem can be ameliorated by using a (Request-to-Send/Clear-to-Send) handshake before each transmission. In this case, before the main data transmission takes place, the transmitting node first broadcasts a small RTS packet and then the receiving node replies with a broadcast CTS packet. If any other nodes hear this conversation they freeze their own access attempts for the length of the transmission. This signaling makes it much less likely that a hidden node will interfere with the intended receiver. In addition any collisions that do occur are more likely to involve small RTS/CTS packets than large data packets.

We are now able to describe the wGPD algorithm in detail. In Section III we discuss the maintenance of the PDQs and in Section IV we describe the scheduling and contention resolution protocol. In Section V we describe the signaling that is required for the exchange of PDQ information and in Section VI we present the congestion control protocol.

III. PDQ MAINTENANCE

The queuing structure of wGPD is the same as that for the theoretical GPD algorithm described earlier. At each node i and for each destination d there is a *Per-Destination Queue* (PDQ) that we denote by Q_d^i. (See Figure I-A.) When data with final destination d arrives at node i, it is placed into that queue. When it is transmitted from the node it is removed from the queue. We use q_d^i to denote the size of this queue. We also assume that each queue has an associated *urgency weight* w_d^i. The urgency weights are the inputs to the scheduling component of wGPD and are defined in exactly the same manner as for the theoretical GPD algorithm, i.e. $w_d^i = [q_d^i - q_d^{n(i,d)}]r_{i,n(i,d)}$, where $n(i,d)$ is the next hop after node i on the route to destination d and $r_{i,n(i,d)}$ is the channel rate between node i and node $n(i,d)$. Note that the definition of these weights requires signaling among neighboring nodes. We discuss how this signaling is performed in Section V.

IV. SCHEDULING AND CONTENTION RESOLUTION

As we have just described, the scheduling and contention resolution decisions are based on the urgency weights associated with the PDQs. In wGPD the scheduling decisions are made in two stages. First, there is an intra-node scheduling procedure in which each node decides which packet it will transmit whenever it is next allowed to make a transmission. Next, there is the contention resolution (or inter-node) scheduling phase in which transmissions that interfere compete among themselves to determine who should transmit next.

The intra-node scheduling phase is extremely simple. Whenever a node has completed a transmission it determines the PDQ Q_d^i for which w_d^i is maximum (and is positive). It then removes a packet from that PDQ and joins in a contention resolution competition that aims to determine which among a competing set of transmissions should next be allowed to transmit.

For inter-node scheduling, it is clear from the definition of the ideal algorithm in Section I-C that its exact implementation would require the solution of a max-weight independent set problem which is not only NP-hard but also hard to approximate. Our inter-node contention resolution protocol is based on the following heuristic. Each node determines the urgency weights of all transmissions with which it will potentially interfere. If it has the maximum urgency weight among those transmissions then it decides to transmit. This heuristic is very natual in itself, and is also motivated, in particular, by the random-access based schemes proposed in [12], [13]. The RC-MAC scheme [12] applies to wireless LAN settings (where all transmissions interfere with each other), and essentially tries to ensure that only the node(s) with the highest urgency weight(s) attempt channel access, thus reducing overall contention. The paper [13] adresses general topology networks; our heuristic can be viewed as the limiting case of the scheme proposed in [13], if we replace urgency weights w_d^i, as defined in this paper, with $f(w_d^i)$, where $f(\cdot)$ is a "steeply" increasing function.

We implement the priorities via a simple modification to the 802.11 contention resolution protocol defined in Section II. This modification is based on the urgency weights and is defined as follows. If a node believes that it has the largest urgency weight in its interfering neighborhood (based on all the urgency weight information it has received), it starts with a small initial value for the contention window (e.g. 32 "slots"), otherwise it starts with a much larger initial value for the contention window (e.g. 128 "slots"). This has the consequence that any node that has the largest urgency weight in its interfering neighborhood is significantly more likely to access the channel during the contention resolution process.

V. SIGNALING

We now describe how the queue states and the urgency weights are exchanged between neighboring nodes. We propose two alternative methods for doing this, depending on whether or not the Request-to-Send/Clear-to-Send (RTS/CTS) protocol is used. The main reason that RTS/CTS is not always used is that for small packets it adds a certain amount of overhead. Moreover, some of the hidden node problems that RTS/CTS was designed to address can also be prevented via an enhanced carrier sensing mechanism in which nodes can hear interference at a range greater than their transmission range.

A. RTS/CTS case

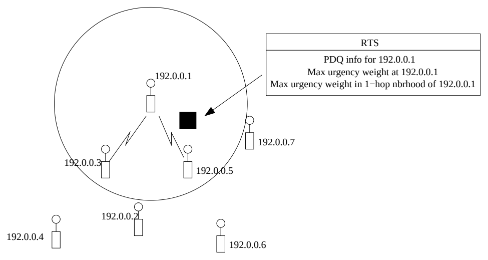


Fig. 2. An RTS transmission showing the attached PDQ and urgency weight information.

The main result of the RTS/CTS process is that nodes that are two-hop neighbors of each other do not interfere. Hence, if we are to regulate the contention process via the urgency weights, we need urgency weight information to propagate to two-hop neighbors. In addition, we need queue information to propagate to one-hop neighbors so that the nodes can calculate the urgency weights. The RTS/CTS packets offer an elegant

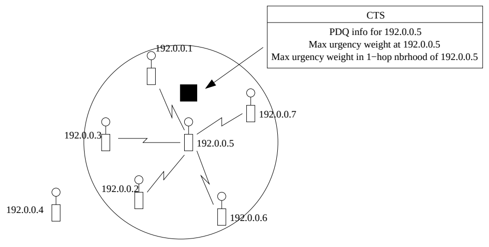


Fig. 3. The corresponding CTS transmission.

way to do this. First, when sending an RTS, the node adds a field that lists its PDQ sizes. It also includes the value of the maximum urgency weight among its queues and an estimate of the maximum urgency in its one-hop neighborhood, which is obtained by the procedure described below. Each node also does this when sending a CTS packet. When a node j receives this information from a neighbor i it runs through all its PDQs and for each destination d it looks to see if node i is the next hop on its path to destination d. If it is, then it sets $w_d^j = [q_d^j - q_d^i]r_{ji}$, where r_{ji} is the current transmission rate between node j and node i. (This rate depends on the transmit power of node j and the channel conditions and interference between node j and node i.) Further, it updates its estimates of the maximum urgency in its 1-hop and 2-hop neighborhoods, denoted by T^j and V^j, respectively, as follows:

$$\begin{aligned}\hat{T}^j &= \max\left\{\hat{T}^j, w^i\right\}\\ \hat{V}^j &= \max\left\{\hat{V}^j, \hat{T}^i\right\},\end{aligned}$$

based on the urgency weight information, w^i and $\hat{T}^i$, contained in node i's transmission. It is clear that $\hat{V}^j$ provides a good estimate of node j's 2-hop max urgency , V^j, as long as the urgency weights do not change drastically over one round of transmissions. Node j then uses $\hat{V}^j$ to decide on its backoff window as described in Section IV.

B. Non RTS/CTS case

There are some cases in which RTS/CTS is not appropriate to use. In particular, for small packets the amount of overhead required by RTS/CTS can be excessive. In this case contention resolution takes place with respect to the data packets only: all the information which would be included into a RTS is included into the data packet itself, and the MAC layer ACK packets carry the information that would otherwise be placed in the CTS packets.

VI. Congestion Control

In this section we outline the methods by which we implement the transport-layer congestion control component of wGPD. In particular, we show how decisions regarding whether or not to inject a packet can be made by examining the size of the PDQ at the source node. We define two types of congestion control, an "unreliable version" that fits within the standard UDP protocol and can be used for flows that can tolerate loss, and a "reliable version" that fits within the TCP protocol and ensures that all data is eventually delivered to the destination. We remark that the entire notion of transport-layer congestion control only makes sense for elastic traffic (or semi-elastic traffic) since for inelastic traffic we are expected to inject all the data that arrives. Hence we assume that we have a utility function for each traffic flow for which congestion control is being applied.

Unreliable version: In the unreliable version of the congestion control protocol, we decide whether or not to inject a packet whenever it arrives from the application layer. The decision is made as follows. For each flow f we maintain an average rate x_f that is an exponentially-filtered average of the amount of data admitted to the flow. The time constant for this filter is some small parameter β, i.e. x_f is multiplied by a factor $1-\beta$ in each time step and is increased by $\beta\ell_p$ whenever a packet of size ℓ_p is injected into flow f.

Suppose that flow f has source s_f and destination d_f. Decisions regarding whether to inject a packet into flow f are made by s_f, and are based on the utility function for flow f and the size of the PDQ $Q_{d_f}^{s_f}$. In particular, for elastic flows a packet is injected as long as $U'(x_f) - \beta q_{d_f}^{s_f} > 0$, where $U_f'(\cdot)$ is the first derivative of $U_f(\cdot)$. In other words, a packet is injected as long as the incremental benefit of the injection is more than a scaled version of the PDQ size at the source node.

If flow f is semi-elastic (and hence has a minimum rate requirement R_f^{min}) we use the same modification that was described for the theoretical GPD algorithm in Section I-C. In this case, a packet is injected as long as $g(\beta K_f) + U'(x_f) - \beta q_{d_f}^{s_f} > 0$, where $g(\cdot)$ is some (sharply) increasing function (e.g. e^x) and K_f is a *token counter* that keeps track of whether or not flow f is meeting its minimum rate requirement. In particular, K_f receives tokens at rate R_f^{min} at all times. Whenever a packet of size ℓ_p is injected into flow f, K_f is decremented by an amount ℓ_p (but is never allowed to drop below zero).

Reliable version: In the reliable version of the protocol, we need sequence numbers and acknowledgments in order to keep track of which bytes are received at the destination. In keeping with the philosophy of making the minimal number of changes to existing protocols, we incorporate the wGPD congestion control protocol within an implementation of TCP which allows us to reuse the TCP sequence number / acknowledgment mechanism. We make two fundamental changes to the TCP congestion control mechanism however. First, we set the TCP congestion window so that it is always equal to the maximum received window size. In this way we essentially disable the effect of the congestion window. Second, whenever TCP makes a decision regarding whether or not to send additional data, instead of using the the TCP congestion window we use the wGPD criterion, namely whether or not $U'(x_f) - \beta q_d^{s_f} > 0$ (or $g(K_f)U'(x_f) - \beta q_{d_f}^{s_f} \geq 0$).

We reuse a number of other components from TCP, in particular the timeout and retransmission mechanism. Whenever TCP suffers a timeout and needs to retransmit, we place the data that has timed out in a retransmission queue. The decision regarding when to inject such data again relies on the wGPD criterion. However, the fact that the timeouts and retransmissions operate in essentially the same way as in TCP means that we can reuse many of the enhancements to TCP that are known in the literature, such as Selective Acknowledgment (SACK) [22].

VII. Simulations

In this section we use OPNET simulation to compare the performance of wGPD with standard protocols such as the 802.11 MAC and the TCP congestion control algorithm. Our results show that:

- The wGPD protocols produce flow rates that are a good solution to the utility maximization problem defined in Section I-B. In particular, the flow rates are close to the optimal rates even in the case that the nodes are mobile.
- For inelastic flows, the wGPD algorithm is able to smooth out bursts of traffic. This keeps queues stable and minimizes packet loss.
- The standard 802.11 protocols, since they do not take queue lengths or the user utilities into account, do not have any of the above properties. In particular, they produce flow rates that do not match the optimum solution of the network utility maximization problem. (In addition, the flow rates tend to oscillate more than the flow rates produced by wGPD.) Moreover, even for inelastic traffic, there are situations where the 802.11 protocols do not keep the queues bounded, even when this is feasible.

We perform our experiments using the OPNET simulation program. We assume that all packets are of size 1024bytes and we use a channel rate of 1Mbps. Although this is lower than many current 802.11 systems, a small rate allows us to see the effects of saturation with lower injection rates and hence the simulations run more quickly. We have verified that at a qualitative level, all our results are unchanged as we move to higher bit rates.

A. Elastic traffic

We begin with a simulation in which all flows are elastic. This means that each flow has a logarithmic utility function that always rewards an additional increase in bandwidth. As mentioned in Section I-B, the logarithm function for applications such as web browsing is motivated by experiments performed with human subjects [3].

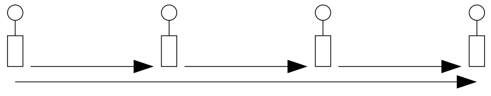


Fig. 4. The simple linear array used in Experiment 1.

Experiment 1: Our first topology is depicted in Figure 4. We have a simple linear array consisting of four nodes each separated by the interference radius. There is one flow that passes through all three hops from left to right. There are three other flows that each pass through one hop only. Since

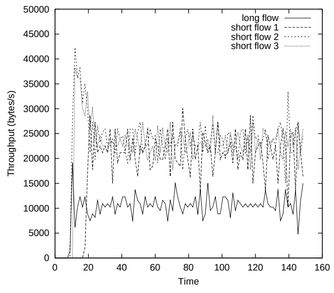


Fig. 5. The flow rates allocated by wGPD in Experiment 1.

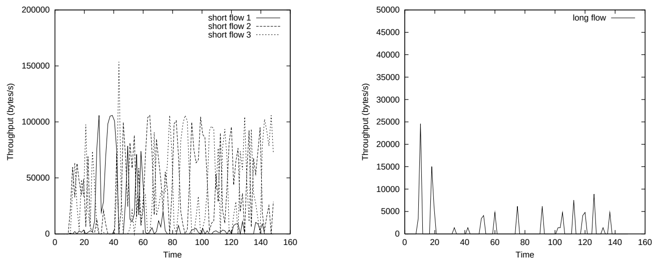


Fig. 6. The flow rates allocated by 802.11+TCP in Experiment 1. (Left) Short flows. (Right) Long flow.

neighboring nodes cannot transmit or receive simultaneously, only one transmission can take place at a time. Since the utility functions are logarithmic each flow consumes the same fraction of network resources under the optimal rate allocation. Since the long flow involves three hops this implies that the bandwidth assigned to the long flow should be one-third of the bandwidth assigned to each of the short flows. A plot of the application level throughputs under wGPD is shown in Figure 5. We see that under wGPD the rate allocated to the long flow is close to this optimal allocation. A close examination of the plot shows that the long flow receives slightly more than one-third of the bandwidth assigned to the long flows. The reason for this discrepancy is that for finite values of the parameter β used in the congestion control, the exponentially filtered values x_f constantly oscillate and so we do not converge to the exact optimum. As β approaches zero, the rate allocations match the optimum allocations more and more closely. However, we may not want to do this in practice since smaller values of β lead to larger PDQ sizes.

The rate allocations under 802.11+TCP are significantly worse. On the left sie of Figure 6 we show the rates allocated to the short flows. We see that these rates are much less stable than the rates under wGPD. More seriously, the rate allocated to the long flow (as shown on the right side of Figure 6) is zero except for occasional small bursts. Hence, under a logarithmic utility model, the aggregate utility will be extremely poor. The reason this happens is that in the 802.11 random access control, each node always has equal opportunity to transmit when it has data. Therefore, in order for a packet to completely traverse a long path, it has to win multiple random access competitions. It is difficult for the long flow to do this at a high rate.

1) Experiment 2: Our next topology is depicted in Figure 7. We once again have a long flow that passes through a three-hop linear array. We also have two one-hop flows that pass start off outside the interference range of the long flow and then move inwards at time 60s so that all flows are mutually interfering. (The node movement is depicted with dotted lines.) The rate allocations under wGPD are depicted in Figure 8. We remark that before the short flows move, their rate allocation saturates in the sense that the transmission rate is equal to the rate at which data arrives from the application layer. We see that after the short flows move, the rate allocations quickly converge to the new optimal values. In contrast, under 802.11+TCP, after the short flows move at time 60s the short flows exhibit large oscillations (see Figure 9) and the long flow typically receives an extremely small rate allocation (with sporadic spikes).

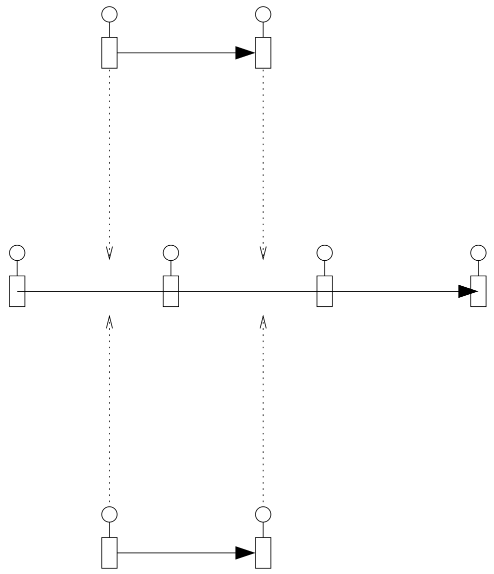


Fig. 7. The simple linear array used in Experiment 2.

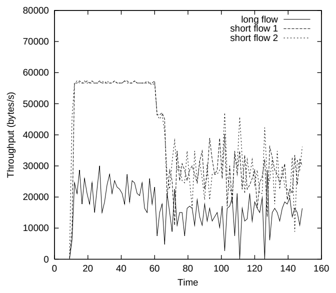


Fig. 8. The flow rates allocated by wGPD in Experiment 2.

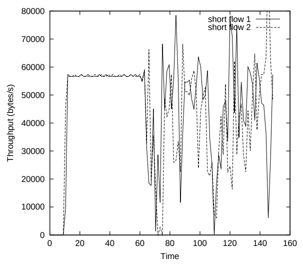
B. Inelastic traffic

In this section we compare the performance of wGPD versus standard protocols in the case that traffic is inelastic, i.e. the sources do not have any congestion control mechanism that can adjust their source rate. We present a situation where the scheduling component of wGPD is able to carry all of the injected traffic whereas the 802.11 scheduling mechanisms are unable to do so.

Experiment 3: This experiment is designed to highlight the contrast between the inherent instability of the 802.11 scheduling protocols and the better stability properties of wGPD. The example is motivated by an instability example for the FIFO scheduling protocol that was first presented in [2]. We use the topology shown in Figure 10. The traffic injections occur in two phases. We assume that at the the beginning of the example we have a small burst of packets arriving at node 0. We can think of these packets as coming from a flow (that we call flow 0) that has node 6 as its destination and that has already stopped injecting. In phase I we have two new flows, one from node 0 to node 11 and one from node 6 to node 12. Then in phase II these two flows cease and we have two additional flows that start injecting packets, one from node 2 to node 11, and one from node 5 to node 13.

The key feature of this example is that due to the interference between the flows, the initial burst of packets that we had at the start of the example gets propagated and amplified under the standard 802.11 scheduling protocols. This is because each burst of data is concentrated. When the data arrives at node 6, all of the bursts collide and cause a large buildup of data which leads to significant packet loss. In contrast, the “backpressure” mechanism of the wGPD scheduler (which comes from the fact that packets are never transmitted from a small PDQ to a large PDQ) means that GPD is able to smooth out bursts of packets so that no packet loss occurs.

The see this, we plot the queue process at nodes 0,1,2 and 6 (where most of the interesting behavior occurs) in Figure 11. We see that for standard 802.11 the buffer at node 6 reaches its limit which leads to significant packet loss. In contrast, the queues under the wGPD protocol remain significantly smaller and hence no packet loss occurs.



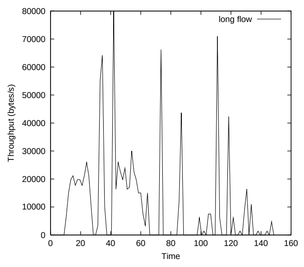


Fig. 9. The flow rates allocated by 802.11+TCP in Experiment 2. (Left) Short flows. (Right) Long flow.

C. Semi-elastic traffic

In this section we present a simple experiment whose aim is to show that the wGPD algorithm is able to support minimum rate guarantees for semi-elastic traffic. In contrast, the standard 802.11+TCP algorithm is oblivious to any such guarantees and hence does not always meet them, even when all of the minimum requirements are feasible.

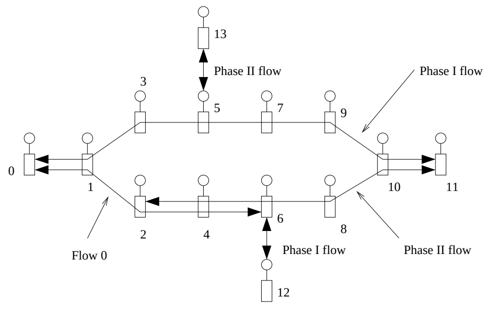


Fig. 10. The topology used in Experiment 3.

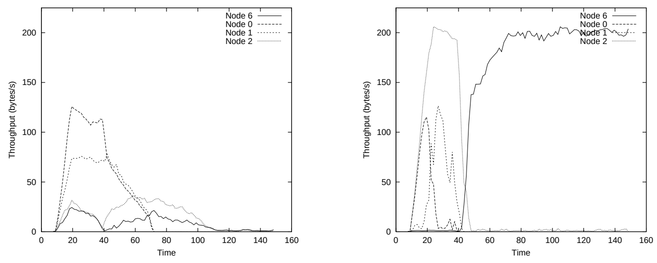


Fig. 11. The buffer process in Experiment 3. (Left) wGPD. (Right) Standard protocols.

Experiment 4: Experiment 4 uses an extremely simple topology with three single-hop flows eacch of which has a common source. One of the flows (flow 0) has a minimum rate guarantee of 40kbytes/s. The standard protocols are oblivious of this requirement and hence allocate rate equally among the flows. Figure 12 shows that the wGPD is able to raise the rate of this flow so that its minimum rate requirement is met.

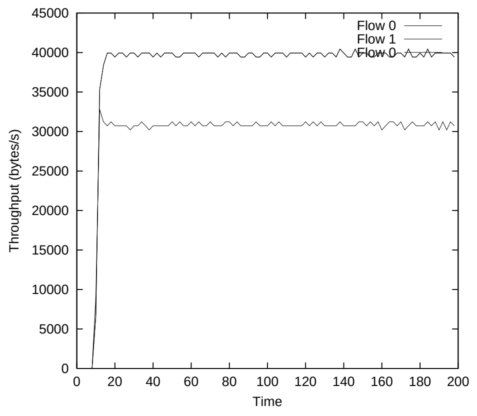


Fig. 12. The flow rates allocated by wGPD in Experiment 4.

D. Multicast

All of the experiments described up to now have been for topologies of modest size carrying unicast traffic. In Figure 13 we show the queue dynamics for a 70 node network carrying multicast traffic. We see that for this experiment wGPD keeps the queue sizes small whereas 802.11 creates clear instabilities.

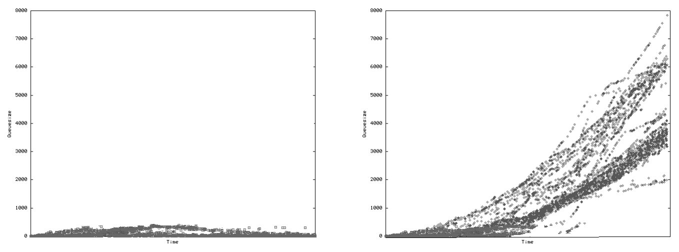


Fig. 13. Plot of queuesizes vs time for 70 node multicast experiment. (Left) wGPD. (Right) 802.11.

E. Other benefits of wGPD

We conclude this section by mentioning two other benefits of wGPD over standard protocols. First TCP typically does not reduce its sending rate until buffer overflows. This leads to large buffer occupancies that in wireless networks can lead to excessive packet delays. This effect was noticed by Chakravory et al. in [7]. In contrast, the wGPD protocols react directly to buffer occupancy at the source node of a flow. This enables wGPD to keep buffer occupancies smaller.

Another benefit occurs when a link breaks. Consider a link break at the end of a linear array. With standard 802.11, packets will continue to be sent to the node next to the broken link. This will cause that node's buffer to grow large and potentially drop packets. The backpressure mechanism of wGPD will spread the load among multiple queues, thereby reducing the probability of packet drops. For this reason we believe that wGPD has potential uses in "Disruption Tolerant Networks" (DTNs).

VIII. Conclusions and Future Work

In this paper we described the wGPD protocol for combined congestion control and scheduling in wireless ad-hoc networks and compared its performance with the standard 802.11+TCP protocols via simulation.

One immediate question is how applicable any new congestion control scheme is since TCP is so widely used in today's systems. This is one of the reasons why in this work we have focused on mobile ad-hoc networks. Although a largescale replacement of TCP would be essentially impossible in today's internet, we believe that some mobile ad-hoc networks (e.g. military networks) are sufficiently self-contained that a replacement of both the scheduling protocols and the congestion control algorithms may be feasible. In addition, the fact that spectrum is limited in many ad-hoc networks means that they would benefit most from the performance gains of wGPD.

There are a number of possible directions for extending wGPD. In particular, we are currently looking at numerous extensions including support for multicast traffic, power and rate control. We are also integrating wGPD with algorithms for admission control and routing. The theoretical GPD algorithm can be extended to handle routing in a very natural way, namely each packet is always forwarded to the neighbor with the smallest PDQ for a destination. However, this can lead to situations where different packets from a flow can follow different paths, thereby leading to out-of-order packet arrival at the destination. We are therefore examining methods through which PDQ information can be used to create more persistent routes.

Acknowledgments

This work is supported by the DARPA CBMANET program. The authors would like to thank Chris Ramming of DARPA and members of the CBMANET-sponsored MARCONI team (T. Abdelzaher, E. Belding-Royer, P. Bose, S. Boyd, A. Carrier, M. Chiang, M. Gersh, A. Goldsmith, P. R. Kumar, D. Li, M. Polyakov, V. Slavin, R. Srikant, N. Vaidya, M. Wittie and A. Zimdars) for many helpful discussions regarding scheduling and congestion control in mobile ad-hoc networks.

References

[1] R. Agrawal and V. Subramanian. Optimality of certain channel aware scheduling policies. In *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, October 2002.

[2] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, January 2001.

[3] M. Andrews, J. Cao, and J. McGowan. Measuring human satisfaction in data networks. In *Proceedings of IEEE INFOCOM '06*, Barcelona, Spain, April 2006.

[4] M. Andrews, L. Qian, and A. Stolyar. Optimal utility based multi-user throughput allocation subject to throughput constraints. In *Proceedings of IEEE INFOCOM '05*, 2005.

[5] M. Andrews and A. Slivkins. Oscillations with TCP-like flow control in networks of queues. In *Proceedings of IEEE INFOCOM '06*, 2006.

[6] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 459–468, 1993.

[7] R. Chakravorty, S. Katti, I. Pratt, and J. Crowcroft. Flow aggregation for enhanced TCP over wide area wireless. In *Proceedings of IEEE INFOCOM '03*, 2003.

[8] L. Chen, T. Ho, M. Chiang, S. Low, and J. Doyle. Optimization based rate control for multicast with network coding. In *Proceedings of IEEE INFOCOM '07*, 2007.

[9] M. Chiang. Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control. *IEEE Journal on Selected Areas in Communications*, 23(1):104–116, 2005.

[10] M. Chiang, D. Shah, and A. Tang. Stochastic stability under network utility maximization: General file size distribution. 2006.

[11] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proceedings of IEEE INFOCOM '05*, Miami, FL, March 2005.

[12] P. Gupta, Y. Sankarasubramaniam, and A. Stolyar. Random-access scheduling with service differentiation in wireless networks. In *Proceedings of INFOCOM '05*, Miami FL, March 2005.

[13] P. Gupta and A. Stolyar. Optimal throughput allocation in general random-access networks. In *Proceedings of CISS '06*, Princeton, NJ, March 2006.

[14] K. Kar, S. Sarkar, and L. Tassiulas. Achieving proportionally fair rates using local information in Aloha networks. *IEEE Trans. Autom. Control*, 49(10):1858–1862, 2004.

[15] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operations Research Society*, 49(3):237–252, 1998.

[16] H. Kushner and P. Whiting. Asymptotic properties of proportional-fair sharing algorithms. In *40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[17] X. Lin, N. Schroff, and R. Srikant. On the connection-level stability of congestion-controlled communication networks. 2006.

[18] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks. In *Proceedings of IEEE INFOCOM '05*, 2005.

[19] X. Liu, E. Chong, and N.B.Shroff. Opportunistic transmission scheduling with resource-sharing constraints in wireless networks. *IEEE Journal on Selected Areas in Communications*, 19(10), 2001.

[20] S. Low and D. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861 – 874, 1999.

[21] S. Low, L. Peterson, and L. Wang. Understanding Vegas: a duality model. *Journal of the ACM*, 49(2):207–235, 2002.

[22] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment options. RFC 2018, IETF, October 1996.

[23] M. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proceedings of IEEE INFOCOM '05*, Miami, FL, March 2005.

[24] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[25] A. Stolyar. On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation. *Operations Research*, 53:12 – 25, 2005.

[26] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936 – 1948, December 1992.

[27] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 30:466 – 478, 1993.

[28] D. Tse. Multiuser diversity in wireless networks. http://www.eecs.berkeley.edu/˜dtse/stanford416.ps.

[29] X. Wang and K. Kar. Distributed algorithms for max-min fair rate allocation in Aloha networks. In *Proceedings of the 42nd Annual Allerton Conference*, Urbana-Champaign, 2004.

[30] L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation in wireless networks. *IEEE Transactions on Communications*, 52(7):1136–1144, 2004.