Delay Reduction and Scheduling in Multihop Wireless Networks

Goutami Garnepudi¹, Ch. Ramesh Kumar²

¹Pursuing M.Tech(CS) from Nalanda Institute of Engineering & Technology, Siddharth Nagar, Sattenapalli, Guntur., Affiliated to JNTUK, Kakinada, A.P., India.

²Asst. Professor, Department of Computer Science Engineering, Nalanda Institute of Engineering & Technology, Siddharth Nagar, Sattenapalli, Guntur., Affiliated to JNTUK, Kakinada, A.P., India.

gowtami.garnepudi@gmail.com

Abstract - In this paper, we analyze the delay performance of a multi-hop wireless network in which the routes between source-destination pairs are fixed. However, its delay performance may be quite poor even when the traffic load is not close to network capacity due to the following reason, each node has to maintain a separate queue for each commodity in the network, and only one queue is served at a time. We develop a new queue grouping technique to handle the complex correlations of the service process resulting from the multi-hop nature of the flows and their mutual sharing of the wireless medium. We propose a dual optimization based approach through back-pressure algorithm which the rate control problem and the scheduling problem can be decomposed.

Keywords – routes, traffic load, node, wireless medium.

I. Introduction

Resource allocation in wireless networks is complicated due to the shared nature of wireless medium. One particular allocation algorithm called the back-pressure algorithm which encompasses several layers of the protocol stack from MAC to routing was proposed by Tassiulas and Ephremides, in their seminar paper [1]. The back-pressure algorithm was shown to be throughput-optimal, i.e., it can support any arrival rate vector which is supportable by any other resource allocation algorithm. Recently, it was shown that the back-pressure algorithm can be combined with congestion control to fairly allocate resources among competing users in a wireless network [2], thus providing a complete resource allocation solution from the transport layer to the MAC layer. While such a combined algorithm can be used to perform a large variety of resource allocation tasks, in this paper, we will concentrate on its application to scheduling and routing.

Even though the back-pressure algorithm delivers maximum throughput by adapting itself to network conditions, there are several issues that have to be addressed before it can be widely deployed in practice. As stated in the original paper [1], the back-pressure algorithm requires centralized information and computation, and its computational complexity is too prohibitive for practice. Much progress has been made recently in easing the computational complexity and deriving decentralized heuristics. We refer the interested reader to [3], [4] and references within for some recent results along these lines. Besides complexity and decentralization issues which have received much attention recently, the back-pressure algorithm can also have poor delay performance. To understand that, we consider two different network scenarios: one in which the back-pressure algorithm is used to adaptively select a route for each packet, and the other in which a flow’s route is chosen upon arrival by some standard multi-hop wireless network routing algorithm such as DSR or AODV and the back-pressure algorithm is simply used to schedule packets. We refer to the first case as adaptive-routing and the second case as fixed-routing, respectively.

We first discuss networks with fixed routing. The back-pressure algorithm assigns a weight to each flow on each link. The weight is equal to the flow’s queue backlog at the transmitter of the link minus the flow’s queue backlog at the receiver. The weight of a link is equal to the maximum weight of any flow that
uses the link. The back-pressure algorithm then selects a schedule which maximizes the sum of the weights of the links included in the schedule. Under such an algorithm, for a link to be scheduled, its weight should be slightly larger than zero. Now, let us consider a flow that traverses $K$ links, and use an informal argument to show why it is very intuitive that the flow’s total queue accumulation along its route should grow quadratically with the route length. The queue length at the destination for this flow is equal to zero.

II. The System Model

We consider a wireless network with $N$ nodes. Let $L$ denote the set of node pairs $(i,j)$ such that transmission from node $i$ to node $j$ is allowed. Due to the shared nature of the wireless media, the data rate $P_{ij}$ of a link $(i,j)$ depends not only on the power $P_i$ assigned to the link, but also on the interference due to the power assignments on other links. Let $\mathcal{P} = [P_i: (i,j) \in L]$ denote the power assignments and let $\mathcal{R} = [R_{ij}: (i,j) \in L]$ denote the data rates. We assume that $R = u(\mathcal{P})$, i.e., the data rates are completely determined by the global power assignment. (Channel variation, e.g., due to fading, is not considered.) The function $u(.)$ is called the rate-power function of the system. There may be constraints on the feasible power assignment. For example, if each node has a total power constraint $P_{\text{max}}$, then $P_{ij} \leq P_{\text{max}}$. Let $\mathcal{R}$ denote the set of feasible power assignments, and let $\mathcal{R} = u(\mathcal{P})$. Let $\mathcal{F}_g$. We assume that $\text{Co}(\mathcal{R})$, the convex hull of $\mathcal{R}$, is closed and bounded.

III. The Shadow Algorithm

In this section, we consider networks with fixed routing, and propose architecture to reduce delays and reduce the number of queues maintained at each node. The main idea is to use a fictitious queueing system called the shadow queueing system to perform flow control and resource allocation in the network while using only a single physical FIFO queue for each outgoing link (also known as per-neighbor queueing) at each node. The idea of shadow queues was introduced in, but the main goal there was to extend the network utility maximization framework for wireless networks to include multicast flows. However, one of the main points of this work is to show that shadow queues can be useful even in networks with unicast flows only for the purpose of delay reduction. Further, the idea of using per-neighbor queueing and establishing its stability is new here.

A. Description

The traditional back-pressure algorithm requires the queue length of every flow that passes through a node to perform resource allocation. The main idea of the shadow algorithm is to decouple the storage of this information from the queueing data structure required to store packets at each node. The details of the shadow algorithm are described as follows.

Queues and Counters: At each node, instead of keeping a separate queue for each flow as in the backpressure algorithm, a FIFO (first-come first-served) queue is maintained for each outgoing link. This FIFO queue stores packets for all flows going through the corresponding link. When a node receives a packet, it looks at the packet’s header: if the node is not the final destination of that packet, it will send the packet to the FIFO queue of the next-hop link; otherwise, it will deliver the packet to the upper layer. We let $P_{nm}[t]$ denote the length of the queue maintained at link $(n,m)$ and at the beginning of time slot $t$.

We recall that shadow queues are just counters. The action of “transmitting shadow packets” is simply the action of updating the counters’ values. In other words, “transmitting” $k$ shadow packets from $Q_f n$ to $Q_f m$ means that we subtract $k$ from $Q_f n$ and add $k$ to $Q_f m$. From the above description, it should be clear that the shadow packets can be interpreted as permits which allow a link to transmit. Unlike the traditional back-pressure algorithm, the permits are associated with just a link rather than with a link and a flow.

Congestion control at the source: At time slot $t$, the source of flow $f$ computes the rate at which it injects packets into the ingress shadow queue as follows:

$$x_f[t] = \min \left\{ U_f \left( \frac{\bar{Q}_f[h_f][t]}{M} \right), \beta \right\},$$

where $\beta$ is an upper-bound of the arrival rates, and $M$ is a positive parameter. The source also generates real traffic at rate $x_f[t]$ where $\beta$ is a positive number less than 1. If $x_f$ and $\beta$ are not integers, the actual number of shadow and real packets generated can be random variables with these expected values. Since the shadow packets are permits that allow real-packet transmission, from basic queueing theory, it follows that the actual packet arrival rate must be slightly smaller than the shadow packet arrival rate to ensure the stability of real queues. The parameter $\beta$ is chosen...
to be less than 1 for this purpose. As we will see later in simulations, the queue backlog in the network would be smaller for smaller values of $\beta$.

The above description of the shadow algorithm applies to elastic traffic. For inelastic traffic, the same shadow algorithm can be used without congestion control. To ensure stability of the real queues, if the real arrival rate of an inelastic flow is $\lambda f$, the shadow arrival rate for this flow must be larger than $\lambda f$. For example, if we wish to make the shadow arrival rate larger than the real arrival rate by a factor of $(1 + \epsilon)$, it can be accomplished as follows: for every real packet arrival, generate a shadow packet. Generate an additional shadow packet for each real packet with probability $\epsilon$. This procedure ensures that the shadow arrival rate will be $(1 + \epsilon)$ times the real arrival rate. For the algorithm to be stable, the set of arrival rates $\{\lambda f (1 + \epsilon)\}$ must lie in the interior of capacity region.

Alternatively, the shadow algorithm for inelastic traffic can be implemented slightly differently if we are willing to tolerate packet loss: fix the shadow arrival rate for each flow and regulate the arrival rate of real packets to be a fraction $\beta$ of the shadow arrival rate. For example, if the rate of shadow arrivals in a time slot is $\lambda f$, then one can inject real packets according to a Poisson distribution of mean $\beta \lambda f$. The real packets could be stored in a queue at its ingress node, and drained at this rate to inject into the network. If the mean real arrival rate is larger than $\beta$ times the mean shadow arrival rate, then the real packet buffer at the edge of the network will overflow leading to packet loss. Packet loss is unavoidable for inelastic flows unless the arrival rate is less than the capacity that the network is willing to allocate to the flow. The shadow arrival rate in this case should be thought of as the network-designated capacity for a flow.

**IV. The Solution**

In this section, we will develop a framework for solving the joint rate control and scheduling problem.

**A. The Route-Independent Case**

We first study the route-independent case using a duality approach. We can assign a Lagrange multiplier $q_d$ (i.e., $d$) for each constraint in (2), construct the Lagrangian and obtain the dual objective function (see [15] for the detail). Let $q_d = 0$ The dual objective function is of the following form:

$$D(\tilde{q}) = \sum B_s(q^d_f) + V(\tilde{q}),$$

$$B_s(q) = \max_{x_s \leq M_s} U_s(x_s) - x_s q,$$

$$V(\tilde{q}) = \max_{r=\sigma(P), P \in \Pi} \sum_{(i,j) \in L} r_{ij} \max(q^d_i - q^d_j).$$

The dual approach thus results in an elegant decomposition of the original problem. Given $\tilde{q}$, we have decomposed the problem into the rate control problem and the scheduling problem. The Lagrange multiplier $q_d$ can be interpreted as the implicit cost at node $i$ for destination node $d$. Each user $s$ solves its own utility maximization problem $\bar{B}_s$ independently as if the “price” for user $s$ is $q_d$. The scheduling problem $V(\tilde{q})$ is precisely the the DRPC policy in [3]. The dual problem of (1) is then

$$\min_{\tilde{q} \geq 0} D(\tilde{q}),$$

$$\frac{\partial D}{\partial q^d_i} = \left[ \sum_{j: (i,j) \in L} r^d_{ij} - \sum_{j: (j,i) \in L} r^d_{ij} - \sum_{x_i = 0} x_i \right].$$

**B. The Route-Dependent Case**

The route-independent formulation in Section III-A is convenient for systems with a small number of destination nodes. For example, for traffic from wireless terminals to the (single) base-station, each node only needs to maintain one queue. No per-flow information needs to be maintained. If the number of destinations is large, each node then needs to maintain many queues, each of which corresponds to one destination node. As we will see next, the route-dependent formulation is more convenient in such scenarios. With the route-dependent formulation, each link again only needs to maintain one queue (or implicit cost) and no per-flow information needs to be maintained. We introduce an auxiliary variable $c_{ij} \geq 0$ for each link $(i, j) \in L$, and rewrite the primal problem (1) as:

$$\max_{|x_{\sigma v} | \geq 0} \sum_{s,v} U_s \left( \sum_{v} x_{sv} \right),$$

subject to $\sum_{s,v} H^v_{ij} x_{sv} \leq c_{ij}$ for all $(i, j) \in L$ and $|c_{ij}| \in C(\mathbb{R}), \sum_{v} x_{sv} \leq M_s$.

The route-dependent case can then be treated analogously to the route-independent case. We can
associate a Lagrange multiplier $q_{ij}$ for each constraint, and obtain the following iterative solution. We will refer to this solution as the link-centric solution, since the Lagrange multiplier is associated with each link.

V. Conclusion

In this paper, we have presented a framework for joint rate-control and scheduling in multihop wireless networks. We proposed a dual approach through which the rate control problem and the scheduling problem can be decomposed. Our solution not only fully utilizes the capacity of the network, but also ensures fairness and good quality of service to the users. We demonstrate via both analytical and numerical results that the proposed mechanism can effectively reduce the queue length and the packet delay inside the network. The most computationally expensive part of the solution is to find the schedule that maximizes the total weighted link capacity at each iteration. For future work, we plan to study simple heuristics that can approximate the optimal schedule. We are particularly interested in heuristic solutions that are easy to implement in a distributed fashion. We will also study how the rate allocation will be affected by the use of these heuristics.

References


