QUALITY OF SERVICE FOR FLOWS IN AD-HOC NETWORKS

by

Eric Chi, Antonis Dimakis, Rajarshi Gupta, Linhai He, Zhanfeng Jia, John Musacchio, Wilson So, Teresa Tung and Jean Walrand

Memorandum No. UCB/ERL M04/44

1 November 2004

QUALITY OF SERVICE FOR FLOWS IN AD-HOC NETWORKS

by

Eric Chi, Antonis Dimakis, Rajarshi Gupta, Linhai He, Zhanfeng Jia,
John Musacchio, Wilson So, Teresa Tung and Jean Walrand

Memorandum No. UCB/ERL M04/44

1 November 2004

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Quality of Service for Flows in Ad-Hoc Networks

Eric Chi, Antonis Dimakis, Rajarshi Gupta, Linhai He, Zhanfeng Jia, John Musacchio,
Wilson So, Teresa Tung, and Jean Walrand
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
smartnets@uclink.berkeley.edu

Abstract— **This paper summarizes the SmartNets research group's work on QoS in ad-hoc networks. It describes the major lessons learned, the remaining open questions, and further research directions. Our work focuses on four major areas: Media Access Control (MAC) protocols, models for capacity estimation, routing protocols, and clustering. Our work in MAC protocols includes work on a novel "impatient" backoff algorithm that enhances fairness between contending nodes, as well as work on a multi channel MAC protocol that improves throughput. We are motivated to study capacity estimation models because many QoS routing algorithms require some methodology for determining how much capacity is available on the different paths of an ad-hoc network. This work has led to a clique based algorithm for residual capacity estimation that can be implemented in a distributed fashion, and that requires a manageable amount of computation. Our work in QoS routing includes routing schemes that maintain network state information, use that state as an input to our clique capacity model, and then use the capacity estimates to guide admission and routing decisions. We also look at more practical QoS schemes that make use of information from both real-time measurements, as well as from maintained state information. The scalability of QoS routing schemes can be improved by dynamically clustering the ad-hoc network into clusters, and having each node maintain detailed state information about its local cluster only, while perhaps keeping some limited global state information. Our work in clustering includes the development of a novel clustering algorithm that adapts to node mobility.**

Keywords: Ad-Hoc Networks, Quality of Service, Routing, Scheduling, Clustering, Interference.

I. Introduction

The wide prevalence of networked wireless devices are moving us towards an era of ubiquitous wireless ad-hoc networks. We can expect in the coming years to see applications like voice and video to be carried over wireless networks. This makes it important to design algorithms to offer Quality of Service (QoS) over ad-hoc networks.

The SmartNets research group has approached the problem from the viewpoints of modelling, algorithms and simulation. We have come up with novel solutions to many related problems, but have also encountered pitfalls. Above all, we learnt several lessons along the way in dealing with ad-hoc networks.

In ad-hoc networks, the term 'QoS' is in itself a tricky concept, since the varying nature of the underlying network makes it difficult to provide any form of guaranteed quality. Our primary focus was towards constant bit rate (CBR) flows like audio and video. We looked at bandwidth requirements and attempted to provide solutions that strive to satisfy those requirements.

This white paper is an attempt to accumulate our various results and to present the overall lessons that we have learnt from this exercise. We realized that QoS in ad-hoc networks is a difficult goal to achieve and that it requires suitable co-operation between various layers of the ad-hoc protocols. Further, there is a wide range of applications and network conditions – implying that a single solution is unlikely to satisfy these varied requirements.

We do not propose a *single* global architecture that is supposed to solve all the problems! Instead, we suggest likely solutions to many of the related problems. In fact, we propose competing solutions to some of the open problems (e.g. QoS routing for multi-hop flows), and discuss situations when each of these would be required.

We also discuss some of the outstanding issues relevant to this problem – areas that have not been the focus of our efforts. We hope that this will encourage others to tackle these problems, many of which lie begging for interesting approaches towards their solution.

The paper is organized as follows. In Section II, we provide an overview of the various ideas we studied, and their usefulness and caveats. In the next four sections (Sections III-VI), we present our schemes and results. In order to properly acknowledge previous work, we have split the related works section into several subsections in each one of the topics covered. Finally we present our conclusions on this topic (Section VII) and a listing of some of the open research issues (Section VIII).

II. Overview

In wired networks bandwidth requirements are per-link constraints – the traffic carried by a link must be less than or equal to its capacity. On the other hand, traffic on an wireless link *interferes* with neighboring links, significantly reducing the capacity of ad-hoc networks. Providing bandwidth-guaranteed service in ad-hoc networks therefore requires consideration of the underlying interference model.

As in [11] and [18], we make use of a "conflict graph" to model the interference relationships between the different links of a network. Every link in the connectivity graph $G = (V, E)$ is represented by a node in the conflict graph $CG =$

This work was supported by the Defense Advanced Research Project Agency under Grant N66001-00-C-8062.

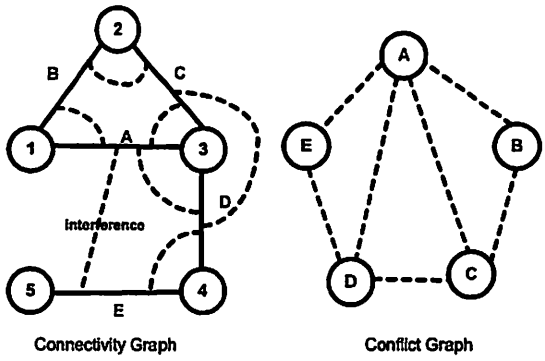

Fig. 1: Connectivity and Conflict graphs

(V^C, E^C). Two nodes in CG are connected by an edge if the nodes correspond to links in G that interfere. In Fig. 1, we show an example of a connectivity graph in which the interference between links is marked using dotted lines. The corresponding conflict graph is shown on the right.

We studied various aspects of providing QoS in ad-hoc networks, looking at the different layers of protocols and how they interact. An overview of the areas that we studied is shown in Figure 2. In particular we looked at media access control and scheduling mechanisms, algorithms for evaluating the current capacity in the network, and routing algorithms.

We first looked at how the nodes in an ad-hoc network are organized. The nodes can be in a flat network topology or in some form of hierarchical topology. Due to the localized nature of ad-hoc networks, it seems logical to conform nodes into clusters that limit the propagation of state information and simplify routing mechanisms. Clustering has the most impact on the choice of routing algorithms, because clustering enables hierarchical routing. This is why we choose to group clustering under the routing heading in Figure 2. However, by simply limiting the amount of inter-cluster knowledge, we do not eliminate the interaction (e.g. interference) between clusters – the effect of this interaction needs to be accounted for in other layers of the protocol besides routing.

The next area of focus is the available capacity in the ad-hoc network. This is another major heading in Figure 2. Given a set of flows in the network, with their bandwidth requirements, we want to determine whether the network can support these flows given the conflict graph. Further, we want to evaluate the remaining capacity in the network that may be used by future flows. We propose to perform this evaluation in a distributed and localized fashion using constraints based on clique structures in the conflict graph.

Unfortunately, the theoretical capacity estimation framework suffers from two practical problems. First, it assumes complete and accurate knowledge of all interference neighbors of a node – this may be difficult to achieve in practice. Second, it only proves the *existence* of a feasible schedule to support the flows in the network, but does not provide the means to achieve it. In fact, distributed scheduling mechanisms like 802.11b are seen to be quite far from the optimally feasible schedule.

However, even with less than complete knowledge, the capacity estimation framework is useful in estimating the available capacity in various parts of the network. This can in turn be utilized to determine QoS routes through the network. In this context, we study several ideas that incorporate ad-hoc constraints in QoS routing.

Routing is the next major heading area of Figure 2. In a strictly hierarchical topology, we propose a cluster-based routing algorithm that divides the routing between inter and intra-cluster segments, thereby benefiting from limited routing tables. Within each cluster, a distributed ad-hoc shortest widest path routing is seen to provide enhanced performance. In ad-hoc networks, it pays to consider multiple distributed algorithms to select the optimal paths – and Interference-aware QoS Routing (IQRouting) chooses one out of several candidate paths. Finally, in the absence of a joint routing-scheduling mechanism, we realize that measurement is the best indicator of actual scheduling policies. Thereby, we incorporate measurements into our routing algorithms, in order to properly reflect the current status of the network.

Scheduling/Media Access control is the last major focus area illustrated in Figure 2. We consider scheduling schemes that are executed independently of routing. While a joint routing-scheduling mechanism would in fact be more optimal, MAC chips typically work in a wide range of devices and do not have the flexibility that would be required for such a mechanism.

So the routing schemes are only best guesses at the QoS route since only a suitable scheduling can truly guarantee quality through the network. In this regard, we consider distributed scheduling mechanisms like iterative longest-queue first, that might achieve stability in an ad-hoc network. We also observe that 802.11 MAC is unfair towards nodes with many neighbors. To address this issue, we propose a novel back-off mechanism that improves fairness in an ad-hoc network spanning multiple interference domains. Finally, we extend our scope of research to multi-channel MACs and discuss how multi-channel contention resolution can lead to improved throughput.

While we have gained insights into several aspects of QoS in ad-hoc networks, the problem space is large enough to warrant much further research. In Section VIII, we attempt to provide a sample of some of these open issues.

III. Clustering

In wired networks, introducing a hierarchy allows for the simplification of routing tables and limits the propagation of changes. Wireless networks experience similar benefits; furthermore, in wireless networks adding structure helps decouple the effects of interference, thereby simplifying the accounting of resources.

Ideally, we would like to assume that constraints in separate clusters are independent: whether a cluster can accommodate a flow is independent of the neighboring clusters. This approximation makes how we cluster important. Specifically, we would like nodes in the same cluster to share common constraints, while minimizing the correlation of constraints across clusters.

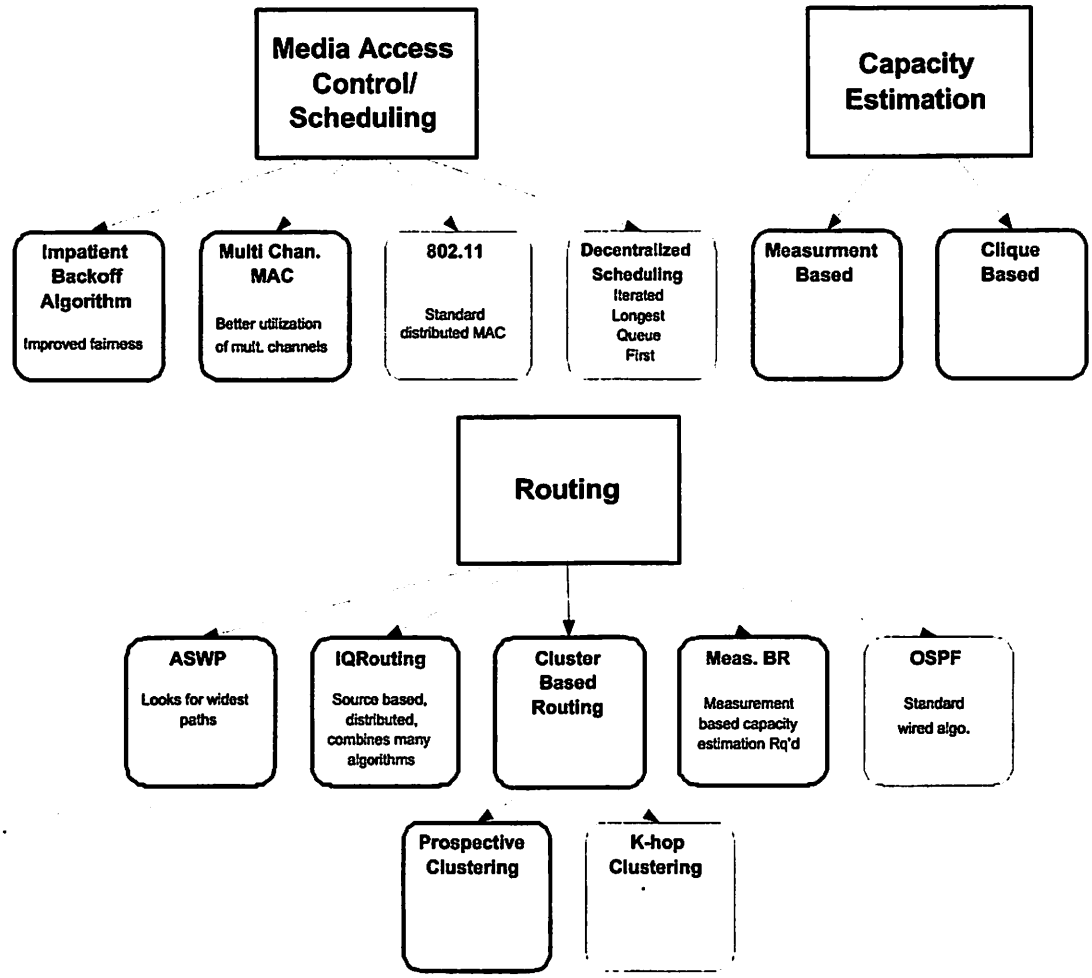

Fig. 2: Overview of Contributions. The major focus areas of our work are shown in large rectangles. Individual algorithms are shown in the smaller rectangles, with a thinner outline for standard algorithms that we study and a thicker outline for novel algorithms.

A. Clustering Algorithms

Numerous clustering algorithms for mobile ad-hoc wireless networks exist in the literature. The most common are variants of K-hop clustering, where every node within a cluster is at most K-hops away from a given node.

Distributed algorithms exist for $K = 1$ ([45]) and $K = 2$ ([44] [47] [48]). These algorithms elect cluster leaders based on some metric (lowest node ID, highest degree of connectivity) and form the cluster by adding neighbors of those leaders. For larger K, [46] requires more centralization of information and computation. These algorithms find a good clustering given a network.

We strive to find the optimal clustering when the network is stable; in this context "stable" meaning that links are not breaking frequently. Any K-hop cluster's constraint must be met by every node within the cluster, however this may not be the bottleneck constraint. In an ideal clustering, we wish to expose the bottleneck constraints as the cluster's constraints. We should choose leader nodes in dense areas and draw cluster boundaries at sparse areas of the network. Under mobility, we use a good clustering but try to refine this clustering when the network is stable.

In our clustering algorithm, each node maintains two types of clustering: prospective and actual. The following distributed algorithm computes the prospective clustering. A leader node is associated with each prospective cluster. Initially, every node starts as its own prospective cluster and as the leader of that cluster.

This algorithm works for any objective that can be locally determined. For example, suppose our objective is to form clusters near size n while minimizing the resulting number of inter-cluster links. We choose the target cluster size n to reflect the expected size of the natural clusters. Or if one does not have such prior knowledge, we choose n to be the square root of the total number of nodes in order to minimize the total size of the routing tables.

At fixed intervals, all nodes broadcast a vector of information, (node ID, cluster ID, path to the leader, number of nodes in cluster), to their one-hop neighbors and send a hello message to the leader node of its cluster. By counting the hello messages, the leader determines the current size of the cluster.

A node updates this size from a neighbor in the same cluster with the shortest path to the leader. If no such neighbor exists, the field is set to infinity.

Nodes dynamically update their cluster ID based on their neighbors' broadcast. After waiting for a random period of time, each node decides whether to update its cluster ID with probability P_{switch}. P_{switch} is lower when the number of nodes in a cluster is near n. The updated cluster ID of a node is a random variable that takes on values from the set of cluster IDs of neighboring nodes and a new cluster ID. The value is chosen in a way that favors the more popular IDs and better cluster sizes.

The prospective clustering may vary frequently and take on suboptimal clustering configurations. However this behavior is necessary to discover better clustering configurations. We define an actual clustering that is updated from the prospective clustering at a slower rate. The actual clustering is more stable and is used by the channel allocation and the routing modules.

B. Cluster Based Routing

Mobility causes changes to cluster membership and cluster locations. Clusters localize information so within a cluster routing tables adapt quickly to the changes. However between clusters, information regarding changes requires time to propagate. Furthermore, updating changes in far-away clusters may be unnecessary. Fisheye ideas in [49] suggest updating local clusters more frequently than far-away clusters.

Under fisheye updates, each cluster has a view of the cluster-level topology of the network. This view contains more accurate information about nearby clusters. Hence we suggest that inter-cluster routing operates on a per cluster basis. A cluster computes a route using its view of the network of clusters. Using the computed route, the cluster forwards the route request to the next hop cluster. This process repeats until the destination cluster is reached or there are no paths available. In even that there is no path available, the request backtracks and tries again. Recomputing routes at each cluster leverages the current information close to the cluster.

IV. Capacity Estimation

A. Related Work

Many researchers have looked at modelling the capacity of ad-hoc networks. In [25], Gupta and Kumar study how the capacity of an ad-hoc network scales asymptotically with the number of nodes in the network. The authors show that the maximum capacity is $\Theta(1/\sqrt{n})$ per node, assuming optimal node placement in a disk of unit area. They also show that under random node placement, the capacity is $\Theta(1/\sqrt{nlogn})$. Other researchers have extended the work of [25] by considering the effects of node mobility [26], and throughput-delay trade-offs [27].

Li, Blake, et. al. analyze the capacity of specific network topologies, and run packet level simulations of both the specific topologies and of random graphs [28]. Luo, Lu, and Bhargavan [11] as well as Jain et. al. [18] study the problem of finding rate constraints on a set of flows in an ad-hoc network, modelling what would be possible if there were a global scheduler. Both works model the interference between links in an ad-hoc network using conflict graphs and find capacity constraints by finding the independent sets of the conflict graph. In [29], Kodialam and Nandagopal model the routing and scheduling of flows in an ad-hoc network as a graph edge coloring problem, and find necessary and sufficient conditions for the achievability of a rate vector. However, this model only considers conflicts between links incident at the same node, and does not take into account interference due to all other neighboring links.

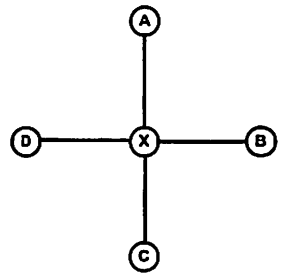

Fig. 3: Star graph

B. Row Constraints

We begin by describing one set of sufficient conditions for a set of flow rates to be feasible. We represent a set of flow rates as the column vector $\mathcal{F}$ of size n, where n is the number of links in the network and $\mathcal{F}_i$ is the flow rate assigned to link i. We also make use of a conflict graph incidence matrix $\mathcal{M}$ defined as follows:

$$\mathcal{M}_{jk} = \begin{cases} 1 & \text{if links } j \text{ and } k \text{ are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

Theorem 1: [17] A set of flow rate assignments $\mathcal{F}$ has a feasible schedule if

$$\mathcal{M}\mathcal{F} \leq \mathcal{C}. \tag{1}$$

where $\mathcal{C}$ is a column vector of size n with all entries equal to the channel capacity C. We refer to expression (1) as the "row constraints" because each row of the conflict graph matrix $\mathcal{M}$ is used to construct a linear inequality on the values of $\mathcal{F}_1$, $\mathcal{F}_2$,... , $\mathcal{F}_n$.

The row constraints may be evaluated in a localized and distributed manner by evaluating

$$\mathcal{M}^i\mathcal{F}^i \leq \mathcal{C}^i, \forall\, i, \tag{2}$$

where $\mathcal{F}^i$ and $\mathcal{C}^i$ are the flow vector and capacity vector representing only the neighbors of link i. Each CG-node only needs local information from all its neighbors to check the validity of these constraints.

While the row constraints are sufficient, they are not necessary. For example, the row constraints in Fig. 3 imply that

$$\mathcal{F}_A + \mathcal{F}_B + \mathcal{F}_C + \mathcal{F}_D + \mathcal{F}_X \leq C. \tag{3}$$

However, expression (3) is much stronger than what is necessary to be feasible. For example, one could achieve the rates $\mathcal{F}_A = \mathcal{F}_B = \mathcal{F}_C = \mathcal{F}_D = C$ if CG-node X is

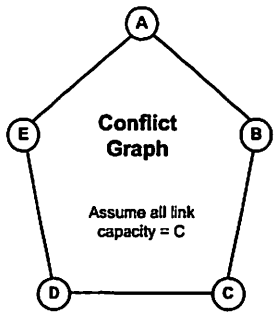

Fig. 4: A pentagon conflict graph

switched off. Yet, if we used expression (3) as our guide (with $\mathcal{F}_X = 0$), we would have to set all the rates to $C/4$. In theory, the row constrained solution could be arbitrarily far from optimal. Hence we are motivated to develop a different set of constraints using cliques. We describe the method in the next section.

C. Clique Constraints

We begin with a few definitions well-known in graph theory. An induced subgraph that is a complete graph is called a *clique*. A *maximal clique* of a graph is a clique such that it is not contained in any other clique. The links in a clique cannot transmit together since they conflict. Accordingly, the sum of the rates of the links in each maximal clique cannot exceed the capacity of the channel; these conditions are the *clique constraints*.

Assume that each CG-node (i.e., link in the connectivity graph) is aware of all the maximal cliques that it belongs to. This information may be described by an incidence matrix $\mathcal{Q}^i$, where

$$\mathcal{Q}^i_{jk} = \begin{cases} 1, & \text{if links } i \text{ and } k \in \text{clique } j \\ 0, & \text{if links } i \text{ or } k \notin \text{clique } j \end{cases}$$

Since a network must satisfy the capacity constraints for all cliques, we can write the clique constraints in a matrix form. As in Section IV-B, $\mathcal{F}$ designates the flow vector and $\mathcal{C}$ the capacity vector. Hence we have,

$$\mathcal{Q}\,\mathcal{F} \leq \mathcal{C}. \tag{4}$$

The clique constraints provide a *necessary* condition for a realizable schedule to exist. One might hope that these constraints would also be sufficient. Unfortunately, that is only true for a special sub-class of graphs called *Perfect Graphs* [34]. Perfect graphs are those whose chromatic number and clique number (size of the largest clique) are equal for all induced subgraphs.

As noted in [18], the simplest example of an imperfect graph where the clique constraints are insufficient is the conflict graph shaped like a pentagon, as seen in Fig. 4. Although the clique constraints suggest a valid flow allocation of $0.5C$ on each link, in reality only $0.4C$ on each link is achievable since at most two out of the five CG-nodes may be active simultaneously.

By modelling a link by its mid-point (Section II), and assuming the interference range is the same for all links, we can approximate the resulting CG by a *unit disk graph* (UDG) [20]. By definition, a unit disk graph is such that there is an edge between two nodes if and only if their Euclidean distance is at most 1 (or a constant value ω). We then use properties of UDG to prove the following theorem.

Theorem 2: [17] When the conflict graph is modelled as a UDG, a set of flow rate assignments $\mathcal{F}$ has a feasible schedule if

$$\mathcal{Q}\mathcal{F} \leq \mathcal{C} \times 0.46. \tag{5}$$

It is worth noting that the clique constraints may be evaluated in a distributed fashion by checking

$$\mathcal{Q}^i\mathcal{F}^i \leq \mathcal{C}^i \times 0.46, \forall\, i, \tag{6}$$

where $\mathcal{F}^i$ and $\mathcal{C}^i$ are the flow vector and capacity vector as known to link i.

A link typically belongs to many maximal cliques in the network. Then, link i's available bandwidth Γ^i is defined as the minimum slack over all clique constraints that this link belongs to, i.e.

$$\Gamma^i = \min_{\forall j \ s.t. \ i \in Q_j} \{0.46 \times C - \sum_{k \in Q_j} \mathcal{F}_k\}, \tag{7}$$

where Q_j is the j^{th} clique in the network, and C is the capacity on each link.

D. Clique Constraints in Non-UDG Networks

Typically, interference regions in a real ad-hoc network are not shaped like perfect disks, due to the presence of obstructions. In such situations, the underlying CG may not be a UDG. One solution to this situation is to construct what we call a "virtual" conflict graph, where links that lie within an interference range of each other are always modelled as conflicting, even if in reality an obstacle prevents the links from interfering each other. Using the virtual-CG we may state and prove the following theorem.

Theorem 3: [17] A set of flow rate assignments $\mathcal{F}$ has a feasible schedule if

$$\mathcal{Q}_V\mathcal{F} \leq \mathcal{C} \times 0.46. \tag{8}$$

where $\mathcal{Q}_V$ is the clique incidence matrix of the virtual-CG, as defined above. This is valid even if the "true" CG is not UDG.

At other times, the interference region may not be shaped like a perfect disk, but be uneven near the edges, due to fading effects. Such an interference region may be modelled as being bounded between two disks. Two CG-nodes cannot interfere if their distance exceeds 1, and two CG-nodes will always interfere if their distance is less than x where $x \leq 1$. For such networks, we have the following result.

Theorem 4: [17] When the interference range ω in a CG varies between x and 1 for $x \in (0.5, 1)$, a set of flow rate assignments $\mathcal{F}$ has a feasible schedule if

$$\mathcal{Q}\mathcal{F} \leq \mathcal{C} \times \frac{z}{1+z} \text{ where } z = \sqrt{x^2 - (\frac{1}{2})^2}. \tag{9}$$

E. Computing Cliques

For our purposes, we would like to compute maximal cliques in a computationally simple, distributed and localized manner. General algorithms to generate cliques in a graph (e.g. [31], [32], [33]) are centralized in nature. Also, these are exponential algorithms since the number of maximal cliques in a graph (even in a UDG) is exponential. So we aim for a polynomial approximation algorithm.

The essence of the approximation is to use 'super-maximal' cliques. When the number of cliques grows large, the approximation algorithm generates the *union* of several nearby cliques as the super-maximal clique. By using approximate cliques, the constraints are only strengthened further as multiple individual constraints are replaced by their union. Thus, the sufficiency of the approximated clique constraints implies the sufficiency of the actual set of clique constraints.

The heuristic algorithm described in [16] distributedly approximates all maximal cliques that a CG-node belongs to. It recognizes two key features about the geographic nature of ad-hoc networks. First, two CG-nodes that are part of a clique must be within an interference radius ω of each other. Second, if a group of CG-nodes form a clique, then the maximum distance between any pair of them must be ω.

The heuristic approximation uses a small disk of diameter ω (i.e. radius = $\omega/2$) to scan a larger disk of radius ω around a CG-node. Each position of the scanning disk generates a clique. The generated set of cliques is then shrunk to generate the approximate set of maximal cliques around the link.

V. Routing

Many prevalent QoS routing algorithms are based on shortest path techniques [1]. However, shortest paths often pass through the middle of the network – so these links are the first to run out of capacity. In an ad-hoc network where each path interferes with multiple links in its vicinity, the effect is greatly amplified. Consequently, QoS routing algorithms using shortest path tend to have low admission ratios.

QoS routing can also be achieved by computing the widest paths for flow requests, following the shortest widest path [2] or widest shortest path [3] algorithms from the wired domain. However, these distributed techniques encounter a problem in ad-hoc networks, due to the failure of *Bellman's Principle of Optimality* [4] that states the following: If an optimal path from node X to node Y passes through Z, it must also be the optimal path from X to Z and from Z to Y. A failure of this principle implies that distributed algorithms cannot find optimal paths in ad-hoc networks.

Fig. 1(a) illustrates an ad-hoc network where we attempt to find the widest paths and encounter a failure of the principle of optimality. We show the links in the ad-hoc network using solid lines and indicate the interference between links using dotted lines. Let the channel capacity be C. The widest path (with capacity C) from node 1 to node 3 is 1-3. But the maximum capacity $C/2$ from node 1 to node 5 is achieved by path 1-2-3-4-5. The shorter path 1-3-4-5 can at most achieve $C/3$ since its three links A, D and E all interfere with each other.

Moreover, notice that widest-path routing balances the traffic loads by taking longer paths, thereby taking up more network resources. In the long run, these paths can end up blocking future flow requests. Research on Ad-hoc Shortest Widest Path (ASWP) algorithm [5] demonstrates that long term admission ratio doesn't always gain from taking the more optimal (e.g. wider), but longer paths. Efficient QoS routing algorithms need to balance the path width and path length.

A. Related Work

The problem of QoS routing in ad-hoc networks has intrigued researchers for some time now, and several solutions have been suggested to guarantee end-to-end quality for flows. The initial solutions considered the bandwidth on an ad-hoc link individually and attempted to find paths that satisfied the quality requirements (e.g. [7] and [8]). Such solutions did not consider the interference between neighboring links, or between multiple hops of the same flow.

One way to effectively provide QoS guarantees for a flow is to settle on a TDM/CDM scheme that chooses the exact time slots to be used by a flow along each link. This approach was proposed in [9], and extended in [10]. The authors of [11] also proposed a way to achieve fair and maximum allocation of the shared wireless channel bandwidth. More recently, [12] and [13] have proposed other algorithms to determine the exact schedule of slots for a flow through the network.

Our work does not assume any TDM/CDM scheme available in the network and considers only the bandwidth requirement on the flows. A similar bandwidth based approach is suggested in [14], which considers the interference between neighboring links, as also the interference between multiple hops on the same flow. The AQOR protocol [15] also maintains neighbor information to incorporate interference and broadcasts the route request. By utilizing the neighborhood bandwidth information for the new flow, feasible paths are detected; the final choice is made at the destination.

B. ASWP

The goal of ASWP is to compute the widest path between the given source and destination nodes. If there are multiple paths with the same largest width, the algorithm chooses the shortest one among them. A flow request is admitted if the path width is larger than the bandwidth requirement. The definition of path width is based on the clique constraints. Specifically, the path width is the minimum residual capacity over all the cliques that the path passes through. Path length is measured by hop-count.

ASWP is designed to be a distributed algorithm. Each node maintains a table containing records for all the destination nodes in the network. A record includes the destination node, the complete path to that node, the path width, and the path length. The record is sent to a neighbor node in an update message, extending the recorded path by one hop. Upon receiving the update message, the neighbor node evaluates the width and length of the extended path and compares it with the metrics in its records. If the new path is better, the node replaces the record and sends a new update message.

The cliques are computed distributedly according to the method in Section IV-E. As a result, each node obtains the information of all the local cliques, i.e., the cliques that the local links belong to.

The key step is to compute the path width of the extended path. Fortunately this can be done with local cliques information, assuring the distributed nature of ASWP. The idea is as follows. For a one-hop extension of the path, the bottleneck clique either remains unchanged or is a clique that the extending link belongs to. In the former case, we know the path width from the record in the update message. In the latter case, we can compute the path width from the clique constraints locally since only local cliques are involved in the constraints.

The ASWP algorithm is polynomial, but finds only approximate solution because of the failure of Principle of Optimality which shows that a sub-optimal path may extend to be optimal, as we saw at the beginning of this section. Consequently, it is preferable to record more then the most promising path for each destination. This leads to a k-ASWP algorithm where each node the records of the k widest paths with ties broken in favor of the shortest. k-ASWP increases the likelihood of finding the optimal solution at the cost of more update messages and computations. We suggest small k values of 2 or 4 to trade off the optimality and complexity.

C. IQRouting

The essence of Interference-aware QoS Routing (IQRouting) lies in recognizing that a single distributed algorithm may not find the best path, as described above. Consequently, we compute several "candidate paths" and choose the best amongst them. This is admittedly a heuristic, but simulations demonstrate significant improvements in admission ratio (up to 30%) over other routing algorithms.

A flow request includes the source, destination, and bandwidth requirement. IQRouting is executed at the source node based on the local view of the network topology and available bandwidth. The candidate paths are the best paths computed by a number of source-routing algorithms. All of these algorithms are polynomial and lightweight, hence the total computational complexity is manageable.

IQRouting uses the following candidate paths:

- Widest Shortest Path (WSP). Choose the shortest path. If multiple choices, pick the widest path. Same as [3].
- WSP Complement (WSPC). Remove all links used by WSP, and select the WSP in the remaining graph.
- Shortest Feasible Path (SFP). Assume an interference model as the feasibility criterion and choose the shortest feasible path.
- OSPF-like Weighted Path Cost (OSPF): Akin to the OSPF [21] algorithm, assign a cost to each link and choose the path with the least total cost.
- Shortest Widest Path (SWP). Look for the widest path available. If multiple choices, pick the shortest.

Some of the above paths may be the same, in which case we have fewer than five candidates.

Once the candidate paths are chosen, we send out a flow setup packet along each path, the entire path being included in the setup packet. Admission control is carried out distributedly at each hop in the path, by evaluating the clique constraints.

To check if the flow is admissible, each node on the path π checks its updated clique constraints, by including the new flow. Let Γ^i_j be link i's slack on clique Q_j. Then, for each link $i \in \pi$, we need to ensure

$$\sum_{k \in Q_j \cap \pi} \beta \leq \Gamma^i_j, \ \forall j \ s.t. \ i \in Q_j. \tag{10}$$

In addition to admission control, the nodes on the path are also required to carry out a distributed comparison of the path metric. For each link i on the path, the node needs to evaluate $\Gamma^i_{new} = \min_{j \ s.t \ i \in Q_j} \left(\Gamma^i_j - \sum_{k \in Q_j \cap \pi} \beta \right)$. This new value of Γ^i is used to compare amongst the various candidate paths being probed.

We propose *two* flavors of IQRouting – IQR-Width (IQRW) and IQR-Cost (IQRC), based on the metric of comparison. IQR-Width chooses the candidate path with the largest width, where width of path π is given by $w_\pi = \min_{i \in \pi} \Gamma^i$. And IQR-Cost chooses the candidate path with the minimum cost, where cost of path π is given by $c_\pi = \sum_{i \in \pi} (\epsilon + (C - \Gamma^i))$. Both these metrics can be evaluated distributedly, and accumulated as the probe packet proceeds hop by hop along the path.

When multiple probe packets reach the destination, the destination chooses the path which has the best cumulative metric and sends a path confirmation back to the source.

D. Measurement-based Routing

One method to account for available capacity uses clique constraints as described earlier. This method relies on computations of cliques and exchanges of link state messages. However correct computations and exchanges of messages may be impossible if the set of interfering nodes is not connected. In Measurement-based routing we offer an alternative method of estimating network resources based on measurements. Furthermore, in the absence of link state messages to update reservations, there is more spare capacity for traffic.

We use measurements as follows: Each node uses local measurements to obtain an estimate of available capacity in that node's interference domain. Nodes measure the idle time defined to be the time when the node is not transmitting or receiving and the channel is not noisy. Available capacity for the node's links is then lower bounded by the link speed scaled by the fraction of available idle time.

1) Fine-Scaled: One possibility is to use the measurements made per node as an estimate of the available residual capacity at a per link basis. The estimates work in conjunction with any routing algorithm that requires available capacity.

2) Coarse-Scaled: Another possibility is to use the measurements on a clustered network. We partition the network into non-overlapping clusters of nodes that interfere with some leader node. Within a cluster, the leader (e.g. a centrally located node) takes measurements to obtain a lower bound on the available capacity of the cluster. Using any node within the cluster affects the available capacity of the leader. The

leader's estimates of available capacity represents the cluster's available capacity, since it is necessary to satisfy the leader's constraint when sending traffic to any node in the cluster.

Observe that the difference between routes through the same interference domain is negligible since both are limited by a shared capacity constraint. Using shortest path routing at an intra-cluster level is desirable to limit the number of transmissions. At the inter-cluster level, routing chooses amongst paths through less correlated interference domains. Inter-cluster routing occurs with any routing strategy on the network of clusters using representative measurements.

3) Refinements via Trial Flows: We use trial flows to refine our estimates of available capacity. Assume that routes are assigned at the source. Consider constant bit rate flows that require rate R. If the flow achieves its desired rate, then we know that there was enough capacity along each cluster. If the flow cannot achieve its desired rate instead it achieves some fraction ρ, then we know that the capacity available to flows from this source is less than R and is at most $\rho * R$.

For higher-priority flows, the source can recompute routes after eliminating clusters that do not have enough spare capacity. Allowing higher priority flows multiple probes of the network increases the chance of finding a path that can accommodate the desired rate. For increasing the probability of acceptance we trade-off longer paths that utilize more network resources.

VI. MAC Scheduling

The set of wireless nodes that are allowed to transmit concurrently without interfering is determined by the operation of a MAC protocol, e.g., 802.11. If the network is to carry a large amount of bandwidth, nodes need to coordinate their transmissions across the network in an optimal manner. This might not be feasible nor desirable in an ad-hoc network because of the large size and temporality of state information involved. Instead, practical MAC protocols have nodes contend for transmission slots on a need basis in a distributed fashion.

In our work on scheduling we attempt answering the following:

1) How much throughput is lost by distributed MAC protocols as compared to the throughput achieved by optimal scheduling protocols?
2) Contention resolution of current MAC protocols, as 802.11, can treat nodes unfairly based on the interference pattern. Can we incorporate fairness?
3) In a large network, contention resolution might inadvertently affect performance. Can we mitigate contention using a multi-channel architecture?

A. Related Work

Throughput optimality of backoff algorithms for contention resolution, in the case where all nodes share the same channel, has been established in [35]: this says that in this case, no throughput is lost by distributed scheduling. In the case of non-trivial interference patterns no such result is available.

In [36] a (throughput) optimal scheduling policy is proposed that depends only on the size of transmission queues inside the network. The algorithm though, requires centralized computation of schedules. In [37] the authors study a class of optimal algorithms for the downlink scheduling problem. Although in the latter case there is no contention, the scheduling part of the problem is the same as the one in the ad-hoc case.

B. Distributed Scheduling

In this work we study the throughput achieved by a simple algorithm that serves as an *idealized* example of a distributed MAC. It is idealized because it ignores the contention problem, i.e., assumes that problem is solved, and focuses on the scheduling problem instead. This simplification is not done only because of model tractability reasons but because we are interested in the throughput loss pertaining to decentralization itself.

In particular, the schedule on each slot is determined as follows: first, the node with the longest transmission queue is included. All nodes interfering with that node are excluded, and the longest one is included from the remaining set of nodes, etc. We keep adding nodes until no further inclusion is possible. This iterative schedule construction cannot be done on-line of course, but it corresponds to the case where transmissions are prioritized according to the size of queues: a node gets higher priority if it has more packets in its queue than it's neighbors. It is important to note that the priority of a node depends only on local information and hence can be implemented distributedly. (For example, MAC backoff timers could use small intervals when the queue size is high.)

This algorithm -iterated Longest Queue First (iLQF)- was proposed in [38] in the context of input-queued switches with virtual outputs as an approximation to the algorithms in [38], [36], but the question of its stability has been left open.

In [39] we have identified sufficient conditions on the conflict graph that imply throughput optimality. In fact, trees (and other graphs too) satisfy that condition. This suggests that this algorithm should perform well in sparsely connected networks; any throughput loss will be due to contention resolution, not scheduling.

On the other hand, we have simulation evidence that optimality does not hold for some graphs that do not satisfy the graph condition above. In Fig. 5 we depict the evolution of queue sizes for a network of 8 nodes forming a ring, i.e., each node interferes with exactly two of its neighbors. The packet arrival process for each node was independent and identically idstributed across time and nodes, and at a rate of 0.498 packets per transmission epoch. Note that the capacity for this network is 0.5 packets per epoch if a periodic (hence not distributed) schedule is used.

This example of instability suggests that even under perfect contention resolution, some loss of throughput is possible because of the myopic nature of distributed MAC protocols not being able to coordinate efficiently over the whole network. Admittedly, these investigations are more interesting because they require the development of new analytical tools than for their practical impact since networks are never expected to operate that close to their maximum throughput. Nevertheless, we believe that gaining a deeper understanding of the stability of MAC protocols is worth the effort.

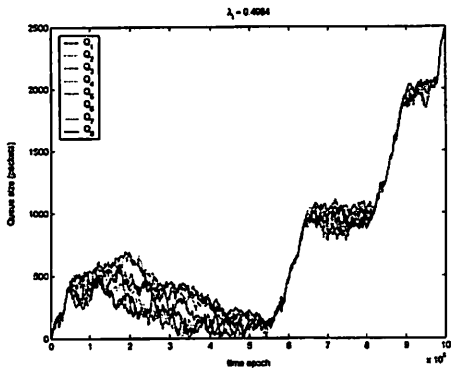

Fig. 5: Example of instability of iLQF for load λ_i less than capacity (0.5 packets/epoch). The queue sizes Q_i are depicted for a network whose conflict graph is an 8 vertex cycle.

C. Fair Scheduling

Traditional ad-hoc MAC protocols (e.g. 802.11b) follow a backoff mechanism whereby nodes try to capture the channel after waiting for a random backoff delay. Upon a collision, nodes are required to increase the average random backoff delay and try again i.e. try less aggressively. When the ad-hoc network spans multiple interference domains, this scheduling, motivated by trying to keep the network stable, is biased towards outer nodes in the network and against nodes in the middle. This phenomenon has been observed in [28] and [17]. This is particularly bad since multi-hop routes tend to use the middle of the network more often than the outer routes.

We propose a novel backoff mechanism that attempts to improve the fairness in a distributed MAC algorithm running across multiple interference domains. This increased fairness is achieved by encouraging nodes to *decrease* their average backoff delay upon collision – thereby becoming *more* aggressive in capturing the next slot. In turn, nodes are required to increase their backoff upon successful transmission. The danger of becoming an unstable system because of frequent collisions is handled by resetting *all* the average backoff delays when one of them becomes too small.

We analyze a Markov model of the protocol to show that our backoff mechanism achieves fairness in simple topologies. We demonstrate the stability of the backoff scheme under certain conditions by constructing a suitable Lyapunov function of the Markov states. We also evaluate the throughput tradeoff that we pay in order to achieve fairness.

However, the analysis can not take into account collisions that happen due to propagation delays and imperfect knowledge of interference. We have therefore built a simulation model to simulate the performance of the backoff scheme in an arbitrary topology and to compare it against traditional 802.11-like exponential backoff mechanisms. Simulation results demonstrate that in a random network, this scheme is able to maintain a level of mean throughput comparable to exponential backoff – but achieves significantly better fairness.

We use the simulation model to study the effect of a realistic reset mechanism which propagates the reset information hop by hop. We are also evaluating the variations caused by the values of certain design parameters.

D. Multi-Channel MAC

Traditionally, nodes in an ad-hoc network use the same frequency band (or channel) even if multiple frequency bands are available. For example, 802.11b radios, being frequency agile, can use any 1 of the 3 non-overlapping bands (i.e., channels 1, 6, or 11). However, all the nodes within the same ad-hoc network must use the same channel. This requirement guarantees that nodes can always communicate with nearby neighbors. However, the unused bands are wasted.

One way to use more spectrum to design a multi-channel MAC that coordinates different pairs of senders and receivers to transfer data in parallel on different channels. Another way is to build wireless radios that are capable of modulating the entire available bandwidth. Below, we compare these two approaches. Fundamentally, however, multi-channel MAC protocols cannot increase network capacity by breaking down available spectrum into narrower non-overlapping frequency bands if we assume the transmit power spectral density (i.e. watt per Hz) is fixed.

Dividing up the available spectrum into bands has its practical advantages. First, nearby but disjoint networks can choose different channels so that interference management is greatly simplified. Moreover, it is challenging to engineer inexpensive radios that can modulate a very wide band. In a situation where a frequency-agile radio cannot modulate the entire available spectrum, it is beneficial to use a multi-channel MAC to allow parallel data transfers on different channels.

Some multi-channel MAC protocols assume nodes can simultaneously receive from more than one channel at the same time. These assumptions are suitable for certain hardware, but not for commercial 802.11 radios which can only transmit or receive on one channel at a time. For these radios, there are two traditional approaches to designing a multi-channel MAC. The first approach (used in [41] and [42]) is to make every node not engaged in transmission or reception hop synchronously through all the channels using a well-known sequence. A sender with outgoing data sends a Request-to-Send (RTS) packet to the receiver on this common hopping sequence. Upon receiving the packet, the receiver returns a Clear-to-Send packet (CTS) and the pair stays on the same channel to transfer data. When the transfer is complete, the pair return to the common hopping sequence to which all idle nodes are listening. The second approach (used in [40]) is to divide time into equal periods. At the beginning of each period, every node returns to a common control channel to negotiate the channels to be used for data transfer for the rest of the period.

The approach we are investigating differs from the approaches mentioned above in that contention happens on all channel simultaneously to avoid a bottleneck in resolving contention on a single channel. SSCH [43] uses a similar approach. Currently, we assume that all the nodes are synchronized. This assumption should be relaxed in the future. In our algorithm, each node uses the same pseudo-random number generator function but different seeds to generate independent

channel hopping sequences. Each node broadcasts its seed and its current index into its sequence in every packet it sends. As a result, when neighbors overhear any packet from this node, they can immediately predict its future hopping sequence. Nodes broadcasts periodically so that their neighbors can eventually discover them.

In the simplest case, if a node i has a packet destined for j, it waits until the first slot in which they happen to hop to the same channel. With a probability that is inversely proportional to the estimated number of nodes on the same channel, the sender sends an RTS message to node j. If i decides not to transmit an RTS during this time slot, it attempts to transmit in the next time slot when they meet again. If two or more nodes transmit in the same slot on the same channel, their packets collide and the slot is wasted. If the receiver receives an RTS message correctly, it returns a CTS message. Both nodes remain on the same channel until they have finished transmitting all packets to each other. In practice, there should be a limit as to how long they can stay together. They then return to their original hopping sequence so that other nodes can find them.

A small, but important improvement to this basic algorithm is to allow sender nodes waiting to meet their receivers a chance to change their channel temporarily to transmit an RTS to their receivers immediately with non-zero probability. This strategy turns out to improve the channel access delay without adversely affecting the total network throughput. Preliminary simulation results suggests that this approach achieves stable queue-lengths even at very high load (80%) under uniform traffic patterns.

VII. Conclusions

The design of ad-hoc network protocols is significantly more challenging than in the wired domain because of the interference between wireless links. This is especially true if one wishes to guarantee quality of service to flows in an ad-hoc network. A flow on one link can have its quality disrupted by an increase of flow on a neighboring link for example. One approach to the problem is to keep track of all the dependencies between links, using a capacity model like the clique-based model we have developed, and admit new flows only when the capacity model says that the new flow will not disrupt the previously admitted flows. However, capacity models are based on idealized assumptions about the network that are often not true in practice. Thus, such a network might admit flows that seem to be feasible according to the capacity model, but are not feasible in practice. A completely opposite approach is to do no modelling of capacity and no bookkeeping of previously admitted flows. While this approach is much simpler and perfectly acceptable for some applications, other applications like voice and video that require some minimal QoS from the network to function correctly are likely to be disrupted frequently if the network does nothing to protect them. For QoS applications we advocate an approach somewhere in-between these two extremes. We also feel that measurements of metrics like the channel idle time are important to corroborate capacity predictions of the capacity model and to make corrections in situations where the capacity model is inaccurate.

Routing is another key area of ad-hoc network design that is significantly harder than in the wired domain. The failure of the Principle of Optimality means that it is impossible in many situations to compute the optimal routes with a Bellman-Ford-like distributed algorithm. Because of this fact, we advocate routing schemes like ASWP and IQrouting that use heuristics derived from traditional wired routing algorithms in order to find good paths in a distributed fashion.

Routing, capacity estimation, and admission control all require knowledge about the state of the network. The amount of information that must be known is even greater in an ad-hoc network than in a wired network because of the inter-dependencies between links. One way of managing these interdependencies is to impose a hierarchy on the network by dividing the network into clusters. This works best when the network has regions densely populated with nodes separated by regions with a sparse number of nodes, so that clusters can be drawn in a way that limits the interaction between them. We study a novel clustering algorithm that works in a distributed manner and explore its use in the context of a cluster-based routing algorithm.

Scheduling and Media Access Control are other key design areas. In theory, the most efficient schedule would be based on all of the information in the network and would be computed jointly with the routing. However this is not practical. Instead, we look at MAC protocols that have nodes contend for the transmission medium in a distributed fashion. In particular we study a novel MAC mechanism that may improve fairness by having nodes become more aggressive after a collision, rather than becoming less aggressive as would happen in a traditional MAC like 802.11. We also study a Multi-Channel MAC protocol that improves efficiency by making better use of multiple channels. Finally we study a distributed scheduling algorithm called iLQF and show sufficient conditions for it to achieve full throughput.

VIII. Open Issues and Future Works

Though we have made progress on understanding how to provide QoS in ad-hoc networks, there is much more to be done. One fundamental issue is to define precisely what QoS means in ad-hoc networks. In wired networks, QoS usually means that the network guarantees something like a minimum bandwidth or maximum delay to a flow. In contrast, it is not possible to guarantee with certainty bandwidth or delay to flows in an ad-hoc network when nodes are mobile and links break and re-join in an unpredictable way. However, it is possible for the network to do the best it can to protect flows in foreseeable circumstances. Defining this notion more precisely might be worthwhile.

We also feel that more work is needed in developing algorithms to cope with mobility. Re-routing schemes that try to find new routes when links go down are a key part of coping with mobility. One re-routing mechanism we like is known as Ad-hoc QoS on-demand Routing [15]. However we feel that there is much more that can be done to better cope with mobility.

ACKNOWLEDGEMENT

The authors would like to thank David Jaffe of Cisco Systems, and Berkeley Professors Venkat Ananthram, David Tse, and Pravin Varaiya, for their help and guidance during this project. We also would like to thank Jun Shu for his work on the Smart Network Toolkit, and Bill Hodge for his help in running simulations.

REFERENCES

[1] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[2] R. Guerin, A. Orda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", *IETF Internet Draft*, November 1996.

[3] Z. Wang and J. Crowcroft, "Quality of service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228-1234, Sept. 1996.

[4] J.Picone, "Bellman's Principle of Optimality," Lecture Notes, ECE 8463, Mississippi State University. Available at: *http://www.isip.msstate.edu/publications/courses/ece_8463/lectures/current/lecture_21/lecture_21_02.html*

[5] Z. Jia, R. Gupta, J. Walrand, and P. Varaiya, "Bandwidth Guaranteed Routing for Ad-Hoc Networks with Interference Consideration," UCB/ERL Technical Memorandum 04/43, 2004.

[6] A. Puri, "Optimizing Traffic Flow in Fixed Wireless Networks", *Proceedings WCNC*, 2002.

[7] E. M. Royer, C. Perkins, and S. R. Das, "Quality of Service for Ad-Hoc On-Demand Distance Vector Routing," *Internet Draft draft-ietf-manet-aodvqos-00.txt*, July 2000.

[8] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad-hoc networks," *IEEE Journal Selected Areas in Communication*, vol. 17 no. 8, pp. 14881505, Aug 1999.

[9] C. R. Lin and J.-S. Liu, "QoS Routing in Ad Hoc Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 14261438, Nov./Dec. 1999.

[10] C. R. Lin, "On-Demand QoS Routing in Multihop Mobile Networks," *Proceedings INFOCOM 2001*, Anchorage, Alaska.

[11] H. Luo, S. Lu, and V. Bhargavan, "A New Model for Packet Scheduling in Multihop Wireless Networks," *ACM Journal of Mobile Networks and Applications (MONET)* vol. 9, no. 3, June 2004.

[12] C. Zhu and M. S. Corson, "QoS Routing for Mobile Ad Hoc Networks," *Proceedings INFOCOM 2002*, New York.

[13] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," submitted for publication.

[14] Y. Yang and R. Kravets, "Contention-aware admission control for ad hoc networks," *UIUC Tech Report*, 2003.

[15] Q. Xue and A. Ganz, "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks," *Journal of Parallel Distributed Computing* vol. 63, pp. 154-165, 2003.

[16] R. Gupta and J. Walrand, "Approximating Maximal Cliques in Ad-Hoc Networks", *Proc. PIMRC 2004*, Barcelona, Spain, September 2004.

[17] R. Gupta, J. Musacchio, and J. Walrand, "Sufficient Rate Constraints for QoS Flows in Ad-Hoc Networks", UCB/ERL Technical Memorandum 04/42, 2004.

[18] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance," *Proceedings ACM Mobicom 2003*, San Diego, CA, USA, September 2003.

[19] S. Gerke and C. McDiarmid, "Graph Imperfection", *Journal of Combinatorial Theory, Series B*, vol. 83, pp.58-78, 2001.

[20] A. Graf, M. Stumpf, and G. Weisenfels, "On Coloring Unit Disk Graphs," *Algorithmica*, vol. 20 (1998), pp. 277-293.

[21] J. Moy, "OSPF Version 2," *RFC 1583*, March 1994.

[22] Matlab Simulation Environment, The Mathworks Inc. *http://www.mathworks.com*

[23] OPNET Modeller, OPNET Technologies Inc. *http://www.opnet.com.*

[24] J. Walrand and P. Varaiya, "High-Performance Communication Networks," *Morgan-Kaufman*, Second Edition, 2000.

[25] P. Gupta, and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910-917, 2000.

[26] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Ad-hoc Wireless Networks," *IEEE/ACM Transactions on Networking* vol. 10, no. 4, pp. 477-486, August 2002.

[27] A. El Gamal, J. Mammen, B. Prabhakar, and D. Shah, "Throughput-Delay Trade-off in Wireless Networks," *Proceedings IEEE INFOCOM*, Hong Kong, March 2004.

[28] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," *Proceedings ACM Mobicom 2001*, Rome, Italy, July 2001.

[29] M. Kodialam, and T. Nandagopal, "Characterizing the Achievable Rates in Multihop Wireless Networks," *Proceedings ACM Mobicom 2003*, San Diego, CA, September 2003.

[30] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, March 2000.

[31] F. Harary, and I. C. Ross, "A Procedure for Clique Detection Using the Group Matrix," *Sociometry*, vol. 20, pp. 205-215, 1957.

[32] J. G. Augustson, and J. Minker, "An Analysis of Some Graph Theoretical Cluster Techniques," *Journal of the ACM (JACM)*, vol. 17, no. 4, pp. 571-588, October 1970.

[33] C. Bron and J. Kerbosch, "Finding All Cliques in an Undirected Graph," *Communications of the ACM*, vol. 16, pp. 575–577, 1973.

[34] L. Lovasz, "A Characterization of Perfect Graphs," *Journal of Combinatorial Theory, Series B*, vol. 13, pp. 95-98, 1972.

[35] J. Hastad, T. Leighton, and B. Rogoff, "Analysis of Backoff Protocols for Multiple Access Channels," in *Proceedings of the nineteenth annual ACM Conference on Theory of Computing*, pp. 241–253, New York, 1987.

[36] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, Vol. 37, No. 12, pp. 1936-1949, December 1992.

[37] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar and P. Whiting, "Scheduling in a Queueing System with Asynchronously Varying Service Rates," *Probability in the Engineering and Informational Sciences*, Vol.18, pp. 191-217. 2004.

[38] N. McKeown, V. Anantharam and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in *Proceedings of IEEE Infocom '96*, Vol. 1, pp. 296-302, San Francisco, March 1996.

[39] A. Dimakis, "Stability and Instability of iterated Longest Queue First Scheduling", unpublished report, 2004.

[40] J. So and N. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver," in *Proceedings of the Fifth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004)*, pp. 222-333, Tokyo, Japan, May 2004.

[41] Z. Tang and J. Garcia-Luna-Aceves, "Hopreservation multiple access (HRMA) for multichannel packet radio networks", in *Proceedings of the IEEE IC3N'98, Seventh International Conference on Computer Communications and Networks*.

[42] A. Tzamaloukas and J. J. Garcia-Luna-Aceves", "Channel-Hopping Multiple Access", in *Proceedings of the ICC 2000* Vol.1, pp.415-419, 2000.

[43] P. Bahl and R. Chandra and J. Dunagan "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks", in *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[44] C. R. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Jour. Selected Areas in Communications*, pp. 1265-1275, Sept. 1997.

[45] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM/Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[46] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. "A cluster-based approach for routing in dynamic networks." *ACM SIGCOMM Computer Communication Review*, pages 49–65, April 1997.

[47] R.Sivakumar, B.Das, and V.Bharghavan, "Spine Routing in Ad Hoc Networks", *ACM/Baltzer Cluster Computing Journal*, 1998.

[48] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, Perth, Western Australia, June 1999, pp. 310-315.

[49] G. Pei, M. Gerla, and T.-W. Chen. "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks." *Proceedings of the IEEE International Conference on Communications*, pp 70-74, New Orleans, LA, June 2000.