BANDWIDTH GUARANTEED ROUTING FOR AD HOC NETWORKS WITH INTERFERENCE CONSIDERATION

by

Zhanfeng Jia, Rajarshi Gupta, Jean Walrand and Pravin Varaiya

Memorandum No. UCB/ERL M04/43

Fall 2004

BANDWIDTH GUARANTEED ROUTING FOR AD HOC NETWORKS WITH INTERFERENCE CONSIDERATION

by

Zhanfeng Jia, Rajarshi Gupta, Jean Walrand and
Pravin Varaiya

Memorandum No. UCB/ERL M04/43

Fall 2004

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Bandwidth Guaranteed Routing for Ad-Hoc Networks with Interference Consideration

Zhanfeng Jia, Rajarshi Gupta, Jean Walrand and Pravin Varaiya
University of California, Berkeley
{jia, guptar, wlr, varaiya}@eecs.berkeley.edu

Abstract— **The problem of computing bandwidth guaranteed paths for given flow requests in an ad-hoc network is complicated because neighboring links share the medium. To address this issue, we first separate the underlying scheduling problem from QoS routing with guaranteed bandwidth, by presenting clique based constraints as the feasibility conditions for flows. We then define the path width, and present the Shortest Widest Path (SWP) routing problem in an ad-hoc network context. We propose a family of distributed Ad-hoc Shortest Widest Path (ASWP) algorithms to address the SWP problem. Numerical simulations compare the performance of these algorithms amongst themselves, and also analyze gains achieved over prevalent shortest path algorithms.**

Keywords: Mathematical programming/optimization, Graph theory.

I. INTRODUCTION

In today's ad-hoc networks, routing is primarily concerned with connectivity. Present routing algorithms, proactive or on demand, source routing or table driven, typically characterize the network with a single metric such as hop count, and use shortest path algorithms to compute paths.

These shortest-path based routing algorithms are not adequate for applications with quality-of-service (QoS) requirements, such as bandwidth guarantees. In wired networks, bandwidth requirements are modelled by independent link capacities $c_{(i,j)}$. Traffic carried by link (i,j) must be less than or equal to $c_{(i,j)}$ but does not consume the bandwidth over other links. Routing algorithms with bandwidth consideration include the *Widest Shortest Path* (WSP) algorithm by Guerin et. al. [1] and the *Shortest Widest Path* (SWP) algorithm by Wang and Crowcroft [2].

Although the WSP and SWP algorithms are efficient in wired networks, they cannot be directly applied to the wireless networks. This is because the bandwidth model in wireless environment involves interference between neighboring links – the transmission on one link taking up capacity in other links in the vicinity. It leads to consideration of an interference model, as well as a scheduling problem.

The interference model is described as an undirected graph CG with respect to the network graph G. By definition in [3], each link in G is represented by a CG-node in CG, and a CG-link exists if the two links in G interfere with each other[1]. The generated CG is called the *conflict graph*. It has been previously referred to by different authors, and also called the contention graph [4], or the interference graph [5].

To route flows across multiple hops, we need to find sets of non-interfering links and schedule them carefully. This forms the scheduling problem. The flows are said to be *feasible* if and only if there exists a set of link schedules that allow the network to carry the flow traffic.

The involvement of scheduling in the routing problem not only increases computational complexity, but also confuses different layers of the network architecture. Moreover, in the scenarios we consider, the wireless nodes employ the 802.11 MAC protocol with pre-defined scheduling schemes that use RTS/CTS signaling to access the medium in a random way. Indeed, because of the lack of infrastructure in an ad-hoc network, it is not practical to implement any centralized scheduling schemes. Hence, separating the scheduling problem from the bandwidth model is desirable and necessary, and is an important basis of the routing algorithm proposed in this paper.

There are three main ideas in this paper. First, we separate the underlying scheduling problem and quantify the feasibility of flows. This becomes the feasibility conditions of the proposed QoS routing mechanism. Second, we present the mathematical abstraction of the *Shortest Widest Path* (SWP) problem that we want to solve. It allows us to compute the maximum amount of traffic that a flow can carry. Though similar to the SWP

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

[1]We use the terms *node* and *link* for the network graph G, and the terms *CG-node* and *CG-link* for the conflict graph CG. Be cautious with these terms since they are different from those in [3].

problem in wired networks, it is NP-complete. And third, we propose distributed *Ad-hoc Shortest Widest Path* (ASWP) algorithms that find paths close to the optimum. The algorithms contain a parameter k that is adjustable to balance between the optimality and complexity. Numerical simulations demonstrate the performance of the proposed algorithms and the effects of different k values. We also provide a guidance on selecting the k parameter.

The rest of the paper is organized as following. We begin in Section II by presenting the related work in the field. Section III presents the scheduling problem and the feasibility of flows. We formulate the SWP problem in Section IV. Sections V and VI present our distributed ASWP algorithm and study its performance. Finally we conclude in Section VII.

II. Related Work

The problem of providing QoS in an ad-hoc network has intrigued researchers for some time now, and several solutions have been suggested to address this. First, we must note that many researchers have looked at providing QoS at the MAC layer (e.g. [6]). While this line of research is quite valuable, the problem we propose to address is different – to guarantee quality for an entire flow from end to end.

The initial solutions addressing end-to-end quality 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 relied on links to adequately estimate the available bandwidth, and did not consider the interference between multiple hops of the same flow.

One way to effectively provide QoS guarantees for a flow is to settle on a time division multiplexing (TDM) scheme that chooses the exact time slots to be used by a flow along each link. Lin and Liu proposed this approach in [9], and Lin extended that work in [10]. The authors of [11] proposed a new way for sharing the ad-hoc network, that seeks to achieve fair and maximum allocation of the shared wireless channel bandwidth. These proposals make a key assumption: By overlaying a Code Division Multiplexing (CDM) scheme on top of the TDM infrastructure, they allow multiple local sessions to share the same slot. This is difficult to implement in a real network due to issues of code and time synchronization between the nodes.

More recently, Zhu and Corson in [12] and Salonidis and Tassiulas in [13] have proposed distributed algorithms to determine the exact schedule of slots for a flow through the network. These schedules take the interference into account and guarantees bandwidth for constant bit rate flows. However, there is still an implicit assumption that the ad-hoc nodes are synchronized to slot boundaries, at a fairly accurate time scale. [13] attempts to overcome the synchronization requirement, but at a fairly high cost, and only for restrictive topologies. Our work does not assume any TDM/CDM scheme available in the network, and works with only the bandwidth requirement on the flows.

A similar bandwidth based approach is suggested by Yang and Kravets [14], who consider the interference between neighboring links, as also the interference between multiple hops on the same flow. The available bandwidth on a link is determined as a function of the neighboring link bandwidths. The route requests are flooded through the network to determine a feasible path that satisfies the bandwidth constraints.

The AQOR protocol proposed by Xue and Ganz [15] also maintain neighbor information to incorporate interference, and broadcast the route request. By utilizing the neighborhood bandwidth utilization of the new flow, feasible paths are detected; the final choice is made at the destination.

Of course, there is a vast body of literature on QoS routing in wired networks. It is imperative to mention these since the ideas presented in this paper are in fact adapted from the wired domain. However, due to space constraints, it is impossible to do justice to describing the related work in this area. So we limit the references here to the two major QoS algorithms referred to already in [1] and [2].

III. Feasibility of Flows

Consider the feasibility problem of a flow in an ad-hoc network $G = (V, E)$. The flow is represented by a vector $\mathbf{x} = (x_{(i,j)} : (i,j) \in E)$ where $x_{(i,j)}$ is the amount of traffic need to be carried over link (i,j). The question is whether flow $\mathbf{x}$ is feasible or not.

We introduce some notation before going on.Denote $l = (i,j) \in E$ to be a link in the graph $G = (V, E)$. Let $CG = (E, I)$ be the conflict graph with respect to G. As the definition in the previous section, $(l_1, l_2) \in I$ is an CG-link in CG if and only if the two links (CG-nodes) l_1 and l_2 interfere with each other, meaning that l_1 and l_2 cannot be active at the same time.

We make the following assumptions about the wireless nodes in the ad-hoc networks. First, every wireless node uses the same power to transmit signals. Secondly, interference happens not only at the receiver of a link, but also at the sender. This is true when the senders expect acknowledgement from the receivers for each successfully transmitted packet. These assumptions are natural

for 802.11 nodes, and imply *symmetric interference* that underlies the undirected conflict graph, i.e., if link (i,j) interferes with link (k,l), (i,j) is also interfered by (k,l) and so must keep silent when (k,l) is active.

The conflict graph describes how links share the medium. Qualitatively, the *cliques* (a complete subgraph) of the conflict graph denote which links are actually sharing the medium, and the *independent sets* (a set of nodes that are not connected with each other) describe which links can be active at the same time. Although it is known [16] that the problems of finding cliques and independent sets are both NP-complete, [3] provides a simple heuristic to compute cliques in the case of ad-hoc networks.

Let $q \subset E$ be a *maximal clique*[2], and Q be the set of all maximal cliques. q can be represented as a row vector $\mathbf{q} = (q_l : l \in E)$. For each link $l \in E$, q_l is 1 if $l \in q$ and 0 if $l \notin q$. The collection of all the row vectors $\mathbf{q}$'s forms a $|Q| \times |E|$ matrix of 0/1, and is called the clique matrix $\mathbf{Q}$. This is nothing but the usual definition of a clique-node incidence matrix [17] for CG.

A. Clique Constraints

Assume that the access of the medium is divided into a number of T time slots for every one-second interval. Let $Z_l \subset \{1,...,T\}$ denote the set of active time slots over link l. Let $z_{l,t} \in \{0,1\}$ indicate whether slot t belongs to Z_l, and z_l be the size of set Z_l. Since only one link can be active in a clique at a time,

$$\sum_{l \in E} q_l z_{l,t} \leq 1, \forall q \in Q, 1 \leq t \leq T. \tag{1}$$

The number of active time slots z_l is decided by x_l, the amount of traffic carried by link l. Define integer vector $\mathbf{z} = (z_l : l \in E)$ where

$$z_l = \frac{T x_l}{C}, \tag{2}$$

and C is the capacity of the medium. Inequality (1) and equation (2) together define the scheduling problem of flow $\mathbf{x}$. Flow $\mathbf{x}$ is feasible if and only if there exists a set of schedules $\{Z_l, l \in E\}$ that solve the scheduling problem.

The scheduling problem of flow $\mathbf{x}$ over the graph G is equivalent to the *weighted coloring* problem over the conflict graph CG. The coloring problem is widely studied in graph theory – it assigns colors to nodes of a graph such that any pair of connected nodes have different colors. The flow vector $\mathbf{x}$ weights the CG-nodes by assigning $z_l = Tx_l/C$ colors to CG-node (link) l. We use the *chromatic number* $\chi(CG, \mathbf{z})$ to denote the least number of colors needed in the weighted coloring problem. Obviously, $\chi(CG, \mathbf{z}) \leq T$ is the necessary and sufficient condition such that the scheduling problem is solvable, or equivalently, that $\mathbf{x}$ is feasible.

The results presented in [18] help to separate the scheduling problem from the feasibility conditions. Define $\omega(CG, \mathbf{z})$ to be the maximum value of $\sum_{l \in q} z_l$ over all the cliques of CG. $\omega(CG, \mathbf{z})$ is actually the *clique number*[3] of a graph $CG_{\mathbf{z}}$ associated with the pair $(CG, \mathbf{z})$, obtained by replacing each CG-node l in CG by a clique of size z_l. The authors of [18] introduce the *imperfection ratio* $\text{imp}(G)$ for a graph G, and show that

$$\text{imp}(G) = \sup_{\mathbf{z}} \left\{ \frac{\chi(G, \mathbf{z})}{\omega(G, \mathbf{z})} : \mathbf{z} \in \mathbb{N}^{|E|} \right\}. \tag{3}$$

Euqation (3) implies

$$\chi(G, \mathbf{z}) \leq \text{imp}(G)\omega(G, \mathbf{z}). \tag{4}$$

We can now introduce the *clique constraints* as the relaxed but sufficient conditions to the feasibility problem of a new flow $\mathbf{x}$. In Proposition 1, the capacity of the medium is scaled down to ensure the sufficiency. The scaling factor is defined as

$$\alpha = \frac{1}{\text{imp}(CG)}. \tag{5}$$

Notice that when the new flow $\mathbf{x}$ is introduced, there could be some flows that have already been installed in the network. However, the scheduling problem only concerns the aggregate of all these flows.

Proposition 1: Suppose there are a number of flows $\mathcal{F}$ that have been installed in the network. Let $\mathbf{x}_f$ be the flow vectors for each installed flow $f \in \mathcal{F}$. A new flow $\mathbf{x}$ is feasible if the following clique constraints are satisfied,

$$\mathbf{qx} = \sum_{l \in E} q_l x_l \leq c_q, \; q \in Q, \tag{6}$$

or

$$\mathbf{Qx} \leq \mathbf{c}, \tag{7}$$

where $c_q = \alpha C - \sum_{f \in \mathcal{F}} \mathbf{q}\mathbf{x}_f$ is the residual capacity in clique q, $\mathbf{Q}$ is the clique matrix, $\mathbf{c} = (c_q : q \in Q)$, and α is the scaling factor.

Proof: Let $\mathbf{z}$ be the total number of time slots needed at each link,

$$z_l = \frac{T}{C}\left(x_l + \sum_{f \in \mathcal{F}} x_{f,l}\right), \tag{8}$$

[2]A maximal clique is a clique such that it is not contained in any other clique.

[3]The cliqnc number is the size of the largest clique of the graph.

where $x_{f,l}$ is the element of $\mathbf{x}_f$ at link l. Applying the clique constraints (6), we have

$$\sum_{l\in E} q_l z_l = \frac{T}{C}\sum_{l\in E}\Big(x_l + \sum_{f\in\mathcal{F}} x_{f,l}\Big) \le \alpha T,\ q \in Q. \quad (9)$$

According to (4),

$$\chi(CG,\mathbf{z}) \le \text{imp}(CG)\omega(CG,\mathbf{z}). \quad (10)$$

Recall the definition of the clique number $\omega(CG,\mathbf{z}) = \max_q(\sum_{l\in q} z_l)$. Then,

$$\chi(CG,\mathbf{z}) \le \text{imp}(CG)\alpha T = T, \quad (11)$$

implying that flow $\mathbf{x}$ is feasible. ■

Thus, the mathematical formulation of the routing problem is converted to finding a flow vector $\mathbf{x}$ that satisfies the connectivity constraints (described in Sec. IV) and the clique constraints (6), which form a convex polytope. The residual capacities of each clique can be easily computed by

$$c_q = \alpha C - \sum_{f\in\mathcal{F}} \mathbf{q}\mathbf{x}_f. \quad (12)$$

B. Bounds on the Scaling Factor

Some bounds on imp(G) are given in [18] for graphs with particular characteristics. For instance, a *unit disk graph* is one that can be embedded in the plane such that two CG-nodes are connected if and only if the Euclidean distance between them is at most 1 [19]. If the nodes of an ad-hoc network are placed on the ground of a free space with no obstacles in between, the associated conflict graph is a unit disk graph. If G is a unit disk graph, then

$$\text{imp}(G) \le \frac{1+\sqrt{3}/2}{\sqrt{3}/2} \approx 2.155. \quad (13)$$

For such networks, we can simply select the scaling factor to be $\alpha = \frac{1}{2.155} \approx 0.46$.

C. Clique Constraints in Reality

To study the effect of the clique constraints as the feasibitlity conditions in a real ad-hoc situation [20], we perform simulations using OPNET [21], which implements detailed packet level simulation models of channels, interference, as well as 802.11 MAC and ad-hoc routing protocols. We generate a random ad-hoc network using a MATLAB [22] program, placing 50 nodes in a $2.5km \times 2.5km$ area. Then, we feed the locations of the nodes into OPNET. Transmission range of the nodes is set to $500m$, and the interference range is $1km$. These numbers roughly correspond to a battalion of tanks in a battlefield, with powerful radios. We set up video flows between five pairs of nodes, and alter the rates in order to change the load of the network.

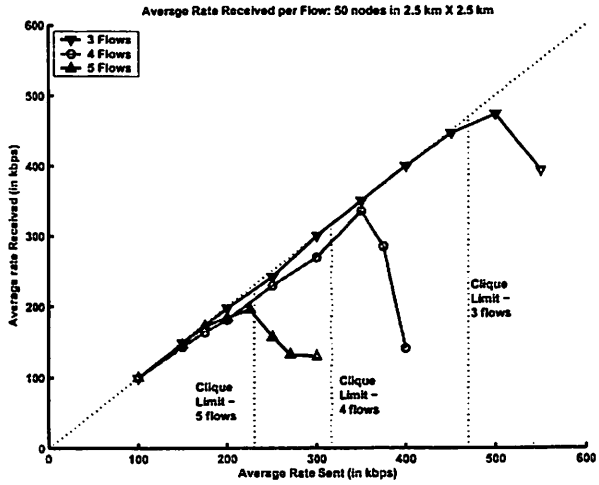

Fig. 1: Comparing real ad-hoc network with theoretical model

The results are shown in Figure 1. By looking at the paths and interpolating between various video rates, we can determine the rate at which the spare capacity somewhere in the network, as predicted by the scaled clique constraints, becomes 0. We determine this limit for simulation scenarios with three, four and five flows respectively, and plot these using the dotted vertical bars.

The curves simply plot the received rates vs. the sent rates for the simulations scenarios involving three, four and five flows. As seen from the figure, the flows receive almost all their traffic until the predicted limit. In each case, the flows experience a sharp loss of quality soon after the theoretical limit is crossed.

The clique constraints only indicates the existence of feasible schedules, but do not offer any way of calculating it. We do not attempt to determine this global schedule, since it would be impossible to impose anyway. Consequently, the OPNET simulations simply model the standard 802.11b behavior. We see that the random access MAC protocol is able to find the schedules for flows that satisfy the clique constraints. Our capacity constraints match the behavior observed in the simulation probably because 0.46 is a good estimate of the combined effects of typical case graph imperfection and the inefficiencies of the distributed scheduling scheme in 802.11b.

IV. Ad-Hoc Routing with Bandwidth Guarantee

The routing problem is to map a flow request (s,d,bw) to a flow vector $\mathbf{x}$ by computing a feasible path

$p = (s, i, j, ..., k, d)$, where $s \in V$ is the source node, $d \in V$ is the destination node, and bw is the bandwidth requirement. The flow vector $\mathbf{x} = \mathbf{x}(p, bw)$ is given by

$$x_l = \begin{cases} bw, & \text{if } l \in \{(s,i),(i,j),...,(k,d)\} \\ 0, & \text{for all other } l \in E. \end{cases} \quad (14)$$

Define path width $width(p)$ to be the largest bw such that $\mathbf{x}(p, bw)$ is feasible. Thus, p is a feasible path for flow request (s, d, bw) if $width(p) \geq bw$.

A. Shortest Widest Path Problem

Consider the widest path problem, i.e., to find the path p from s to d that maximizes $width(p)$. Then, for any flow request (s, d, bw), we can simply compare bw with the width of the widest path and we will know whether to admit the request or to reject it. The mathematical formulation of the widest path is as following.

$$\max \quad bw \quad (15)$$

$$\text{s.t.} \quad \sum_{(i,j)} x_{(i,j)} - \sum_{(j,k)} x_{(j,k)} = \begin{cases} -bw, & j = s \\ bw, & j = d \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

$$\mathbf{Qx} \leq \mathbf{c} \quad (17)$$

$$x_{(i,j)} \in \{0, bw\}, \forall (i,j) \in E. \quad (18)$$

The solution to problem (15) guarantees that for any flow request, if there exists a feasible path to accommodate it, we are able to admit it and assign a path to it. However, some concerns still exist. First, the path computed by (15) may contain loops. Indeed, path width is determined by the bottleneck cliques. Loops at the under-utilized part of the network may not affect the width at all. Secondly, it is always good to use as little network resources as possible, especially in the ad-hoc network where nearby links share the medium. These concerns lead us to select the shortest path among the widest paths, forming the *Shortest Widest Path* (SWP) problem. Shortest widest paths have the loop-free property, similar to the property of the shortest widest paths in wired networks [2].

Proposition 2: The solutions to the SWP problem are loop-free.

Proof: The key of the proof lies in the fact that when flow $\mathbf{x}' \leq \mathbf{x}$, $\mathbf{x}$ is feasible implies that $\mathbf{x}'$ is also feasible. To see this, suppose $\{Z_l, l \in E\}$ is a feasible schedule for flow $\mathbf{x}$. Since $x'_l \leq x_l$ for all $l \in E$, let Z'_l be any subset of Z_l of size Tx'_l/C, we have

$$\sum_{l \in E} q_l z'_{l,t} \leq \sum_{l \in E} q_l z_{l,t} \leq 1, \forall q \in Q, 1 \leq t \leq T. \quad (19)$$

Therefore, $\{Z'_l, l \in E\}$ is a set of schedules for flow $\mathbf{x}'$, and $\mathbf{x}'$ is feasible.

Suppose path $p = (s, ..., i, j, ..., k, i, ..., d)$ is a solution to the SWP problem containing loop $(i, j, ..., k, i)$. Let $p' = (s, ..., i, ..., d)$ be the shortened path by removing the loop. By definition of the shortest widest path,

$$width(p) \leq width(p'). \quad (20)$$

However, since for any bw, $\mathbf{x}'(p', bw) \leq \mathbf{x}(p, bw)$, thus,

$$width(p') \leq width(p). \quad (21)$$

Combining (20) and (21) together implies

$$width(p') = width(p). \quad (22)$$

Therefore, p' is also a widest path. But p' is shorter than p, contradicting the hypothesis that p is a solution to the SWP problem. ∎

Notice that the solution to the SWP problem being less than or equal to bw doesn't necessarily mean that the flow request cannot be accommodated. The scaled clique constraints (17) are the sufficient conditions to the feasibility, not the necessary conditions. In fact, the *unscaled* clique constraints are necessary for a feasible schedule to exist [23].

B. Widest Shortest Path Problem

The *Widest Shortest Path* (WSP) problem is also an abstract of the routing problem with bandwidth consideration. It wants to find the widest path from the set of the shortest paths. As the name suggests, path length has the priority to path width.

Path width and length are two metrics that address different requirement issues. From flows' point of view, path width is related to the bandwidth requirement, while path length is related to the delay requirement. The path length is also concerned with the amount of network resources used by a flow. Since the purpose of this paper is to provide bandwidth guaranteed paths for flow requests, the precedence is placed on the width metric. Therefore, the SWP problem is the mathematical abstraction that we want to solve.

V. Ad-hoc Shortest Widest Path Algorithms

Our goal is to design a distributed algorithm towards the SWP problem that minimizes the exchanged information and overhead. Look at problem (15). The second line (16) is the connectivity constraints, containing $|N|$ equations that represent the network topology. The third line (17) is the clique constraints, containing $|Q|$ inequalities over the maximal cliques of the conflict graph CG. A centralized algorithm requires all these information.

One can either collect the conflict graph by link state protocols and compute the maximal cliques at each node, or compute the maximal cliques distributedly and exchange the clique information. In both cases, overhead will be large.

It is however possible to compute the maximal cliques in a distributed manner. Notice that interference will not occur if two links are far from each other. Assume that the distance threshold is d. For any link $l \in E$, only the links within a disk of radius d are required to build a subgraph and compute the maximal cliques that l belongs to. For the ad-hoc networks whose conflict graph are unit disk graphs, Gupta and Walrand [3] proposed a polynomial-time approximation algorithm to approximate the maximal cliques in a heuristic way.

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. Here ω is the interference range of the CG-nodes. Each position of the scanning disk generates a clique. The generated set of cliques is reduced to result in the set of maximal cliques around the link.

A. The Basic Heuristic

The proposed *Ad-hoc Shortest Widest Path* (ASWP) algorithm follows the Bellman-Ford architecture. Each node maintains a table of the best paths to all the destination nodes in the network. Each row of the table forms a *record*, denoted by a quaternity $r_d = (d, p, width(p), len(p))$ that contains the destination node id d, the complete path p from the self to the destination, the path width, and the path length in terms of hop-count. The algorithm performs up to $|N| - 1$ rounds of relaxation operations over all links. These relaxation operations update (or relax) the width and length of the appropriate records in the table.

A complete description of the ASWP algorithm is as follows:

Step 1: For each node $i \in N$, initialize the records $r_{i,d} = (d, \emptyset, 0, 0)$ for all $d \neq i$, initialize $r_{i,i} = (i, i, \infty, 0)$ and mark it *active*.

Step 2: For each *active* record $r_{j,d}$ of node j, send the *update massage* $r_{j,d}$ to every incoming neighbor i such that $(i, j) \in E$ is a link of the network.

Step 3: Upon receiving the *update message* $r_{j,d}$, call the *relaxation operation* at node i, as

1) Let $\tilde{p}_{i,d} = (i, p_{j,d}) = (i, j, ..., d)$ be the extended path;
2) Compute $width(\tilde{p}_{i,d})$ and $len(\tilde{p}_{i,d})$;
3) Compare with the record $r_{i,d} = (d, p_{i,d}, width(p_{i,d}), len(p_{i,d}))$. If

$$width(\tilde{p}_{i,d}) > width(p_{i,d}),$$

or

$$width(\tilde{p}_{i,d}) = width(p_{i,d})$$

and $\quad len(\tilde{p}_{i,d}) < len(p_{i,d}),$

the *relaxation operation* succeeds and the record $r_{i,d}$ is updated to be

$$r_{i,d} = (d, \tilde{p}_{i,d}, width(\tilde{p}_{i,d}), len(\tilde{p}_{i,d}));$$

4) Mark the record $r_{i,d}$ *active* if it is updated.

Step 4: The algorithms stops if no record is marked *active*. Otherwise, return to Step 2.

The key step in the ASWP algorithm is to compute the width and length of the extended path $\tilde{p}_{i,d} = (i, p_{j,d})$ in the relaxation operation. The length is simple, as $len(\tilde{p}_{i,d}) = len(p_{j,d}) + 1$. The width, however, needs more computation. It is the largest bw such that $\mathbf{x}(\tilde{p}_{i,d}, bw)$ satisfies the clique constraints (17).

Remember that $width(\tilde{p}_{i,d})$ is computed at node i, a node that owns the information of the maximal cliques $Q_{(i,j)} = \{q : q \ni (i, j),\ q \in Q\}$ that link (i, j) belongs to. We show that this information is enough for node i to compute $width(\tilde{p}_{i,d})$. Let $\bar{\mathbf{x}} = \mathbf{x}(p, 1)$ be the unit bandwidth flow carried by path p. Thus,

$$width(p) = \min_{q \in Q} \frac{c_q}{\mathbf{q}\bar{\mathbf{x}}}. \tag{23}$$

Since $\tilde{p}_{i,d}$ is one-hop extension of $p_{j,d}$,

$$\bar{\tilde{\mathbf{x}}}_{i,d} = \bar{\mathbf{x}}_{j,d} + e_{(i,j)}, \tag{24}$$

where $e_{(i,j)}$ is a vector with only one 1 at link (i, j). Therefore,

$$\begin{aligned} width(\tilde{p}_{i,d}) &= \min_{q \in Q} \frac{c_q}{\mathbf{q}\bar{\tilde{\mathbf{x}}}_{i,d}} \\ &= \min\Big(\min_{q \in Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\tilde{\mathbf{x}}}_{i,d}}, \min_{q \in Q \setminus Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\mathbf{x}}_{j,d}} \Big) \\ &= \min\Big(\min_{q \in Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\tilde{\mathbf{x}}}_{i,d}}, \min_{q \in Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\mathbf{x}}_{j,d}}, \\ &\qquad \min_{q \in Q \setminus Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\mathbf{x}}_{j,d}} \Big) \\ &= \min\Big(\min_{q \in Q_{(i,j)}} \frac{c_q}{\mathbf{q}\bar{\tilde{\mathbf{x}}}_{i,d}}, width(p_{j,d}) \Big) \end{aligned} \tag{25}$$

The implication is, when a path extends, the bottleneck clique either remains unchanged or becomes one of the maximal cliques that the extending link belongs to. This is an important property that makes the distributed algorithm possible.

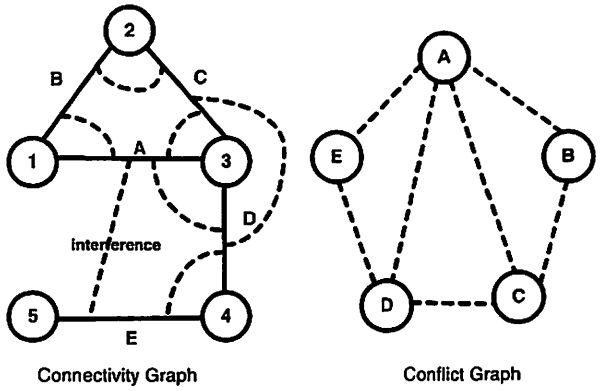

Fig. 2: QoS Routing with interference does not follow shortest path paradigm

B. kASWP Extension

The ASWP algorithm is distributed and polynomial. Unfortunately, it is inherently an approximate algorithm because of the failure of the *Principle of Optimality*. Bellman's Principle of Optimality [24] states: 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. Interference in an ad-hoc network does not conform to this paradigm. Specifically, a partial part of a shortest widest path may not be optimal itself.

Figure 2 illustrates an example in an ad-hoc network where we attempt to find the widest paths. The channel capacity is denoted by C, and the figure presents both the connectivity graph and the conflict graph. The interference between the links in the connectivity graph are marked using dotted lines – which become the edges of the conflict graph.

Clearly, the widest path from node 1 to node 3 is 1-3. But note the widest path from node 1 to node 5. The maximum capacity of $C/2$ 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 conflict with each other.

The Bellman-Ford architecture of ASWP allows each node to keep one record of the most promising path to a destination. However, since a sub-optimal path may extend to be optimal, it is efficient to keep multiple records of the most promising paths for each destination. This leads to a kASWP algorithm where each node maintains a table of up to $k|N|$ records. The kASWP algorithm increases the possibility of finding the optimal solution with large k value. On the other hand, kASWP will send up to k times more update messages than ASWP. So a small k is desirable. Simulations in the next section uses values of k as 2, 4, and ∞ to demonstrate the trade-off between the optimality and complexity.

VI. Numerical Simulations

Qualitatively, we would like to show how the proposed algorithms improve the path width for source/destination (s/d) pairs so that the flows can carry more traffic, and the network can carry more flows. To achieve this goal, we turn to simulations conducted over the 50-node network described in section III-C. We implement five algorithms in MATLAB to compute paths between s/d pairs. The first three are the proposed ASWP algorithms, including the basic ASWP algorithm and the extended 2ASWP and 4ASWP algorithms, where the numerical digits (2 and 4) indicate how many records each node has to keep. The fourth algorithm is an ideal SWP algorithm that computes the optimum of the SWP problem precisely by solving the mixed integer programs given in Sec. IV-A. This provides an ideal to compare against. Notice that the ideal SWP is an extreme version of kASWP where $k = \infty$, while ASWP is another extreme with $k = 1$. Therefore, the first four algorithms can be all viewed as kASWP with $k = 1$, 2, 4 and ∞ respectively. They are together referred to as the "ASWPs algorithms". Finally, we implement the Bellman-Ford shortest path algorithm, since shortest paths are often used in ad-hoc QoS routing (Sec. II) and are useful for performance comparison.

A. Static Simulations

1) Path Width: We compare the path width made available on the network by the various algorithms that we consider. We increase the load on the network by placing constant rate flows one-by-one into the network. The sources and destinations are randomly selected.

The load is represented by the *average clique utilization*, which is the average ratio of the used capacities over all cliques. Notice that the used capacities vary a lot from clique to clique, depending on how the flows are placed into the network. Therefore, the average clique utilization gives only a rough measure of the load of the network.

We also differentiate the simulations with respect to the distance of s/d pairs. We use the hop counts of the shortest path between the pair to represent the distance. Intuitively, distant pairs seem to be more *improvable*, meaning that the widest path is wider than the shortest one. To see this in an opposite way, think of an one-hop pair: the shortest path is clearly the widest.

The first set of simulations compare the width of given s/d pairs as the network load increases. Figure 3 shows the results for a pair of nodes (chosen randomly) that are 7 hops distant. In the Y-axis, we plot the width of the best

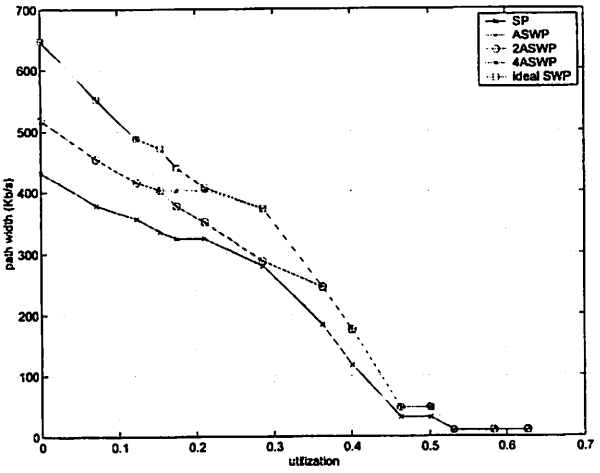

Fig. 3: Path width for a distant s/d pair (7 hops away).

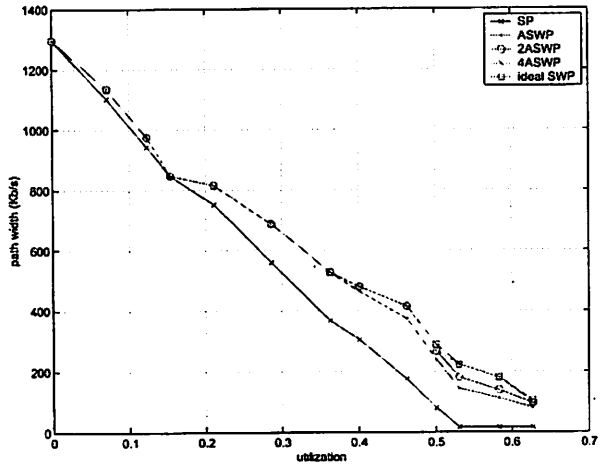

Fig. 4: Path width for a close s/d pair (2 hops away).

path between this pair of nodes – as computed by each of the algorithms. We plot this against rising utilization in the network. As seen in the figure, the ASWP algorithm finds paths that are significantly wider than the shortest paths; 2ASWP returns the same solutions as ASWP; and 4ASWP improves the path width very close to the optimum. For instance, when network utilization is 0.12, the path width of the shortest path is 357 Kb/s; ASWP/2ASWP improve the value to be 416 Kb/s; And 4ASWP finds the optimal 488 Kb/s.

When the load of the network increases, the improvements shrink. Indeed, when the utilization is over 0.53, there is no improvement at all. This can be explained by the fact that the network is so congested that none of the algorithms can find good paths. At this point, the widest available path is down to a few Kb/s.

The results are somewhat different for nearby s/d pairs. Figure 4 shows the same simulations for another randomly chosen s/d pair that are only 2 hops distant. In this figure, as the load of the network increases, all the ASWPs algorithms continue to find wider paths. Consider the points where utilization is 0.53: We observe that the shortest path can carry only 15 Kb/s; At the same situation, path width is 143 Kb/s by ASWP, 181 Kb/s by 2ASWP, and 222 Kb/s by 4ASWP. To see this, remember that the residual capacities vary significantly from clique to clique, especially when the load is heavy. For distant s/d pairs, there are always some congested cliques becoming the bottleneck, since paths traverse many cliques. But for close s/d pairs, it is sometimes possible to find paths that avoid the congested cliques. These paths can therefore be wider than the shortest ones.

2) Improvement over all s/d pairs: Not all s/d pairs have paths that are wider than their shortest paths. The next set of simulations differentiate the s/d pairs with respect to the hop distance, and evaluate the improvement over *all* s/d pairs.

The simulations run at two utilization levels: an empty network with no flows installed, and a medium-loaded network with the utilization being 0.32. The load of the network is achieved by placing random flows in the network, as we did in the Sec. VI-A.1. Note that the randomly placed flows do not load the network uniformly, different parts are loaded to different levels. The simulations compare the ASWPs algorithms with the simple shortest path, and compute the width improvement from the width of the shortest path.

First, we consider the percentage of s/d pairs whose path widths experience an improvement. The simulations consider 2,450 s/d pairs of the 50-node network (total number of s/d pairs $= |N| \times (|N| - 1)$). Among the total pairs, 255 (10.4%) pairs have improved width when the network is empty. When the network is partially (0.32) loaded, 869 (35.5%) pairs are improvable. For the rest of the pairs that are not improvable, the shortest paths are also the widest.

As mentioned, the improvements depend on the distance between the s/d pairs. Figure 5 shows the results in the empty network. The upper figure shows the ratio of the improvable s/d pairs versus the hop distance of each pair. We see that for close s/d pairs whose distance is 3 hops or less, there is no improvement at all. The ratio increases when the distance increases. For instance, the farthest s/d pairs of 10 hops away (there are 8 such pairs) are all improvable, though ASWP, 2ASWP and 4ASWP only manage to better half of them.

This trend is also seen in the lower figure, which

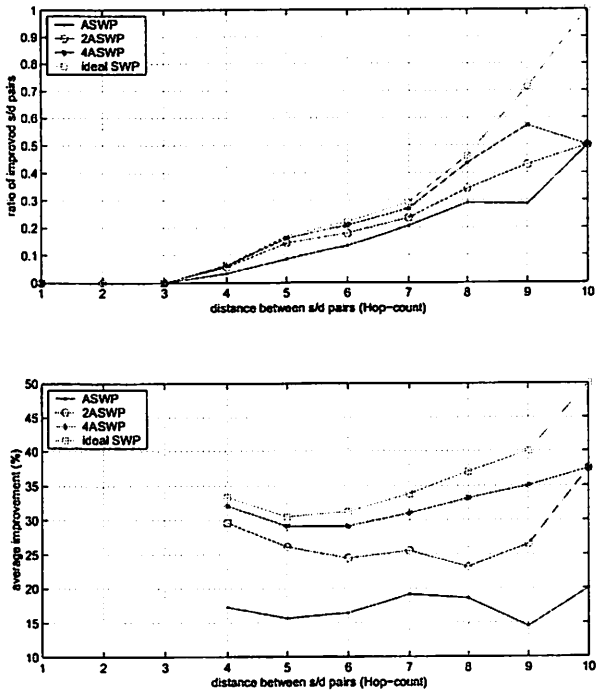

Fig. 5: The upper figure shows the ratio of the improved s/d pairs; the lower figure shows the average improvement. The results differentiate with respect to the hop distance. The load is low (empty, utilization = 0).

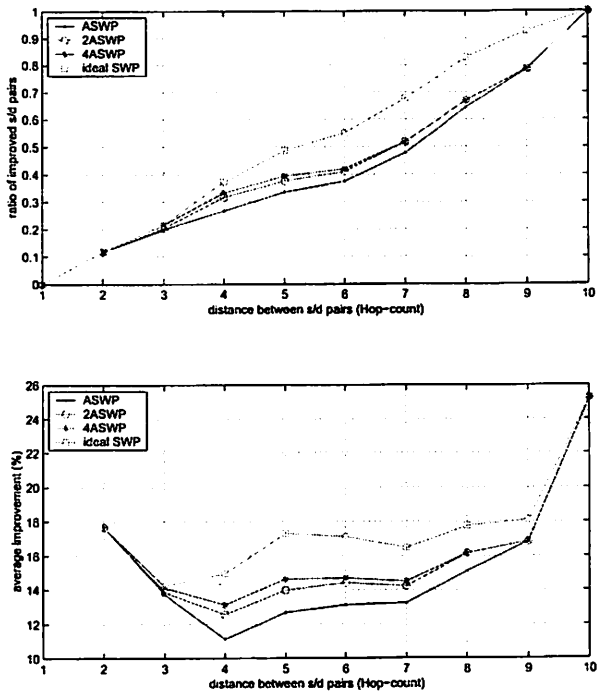

Fig. 6: The upper figure shows the ratio of the improved s/d pairs; the lower figure shows the average improvement. The results differentiate with respect to the hop distance. The load is medium (utilization = 0.32).

presents the percentage improvement in the path width achieved by using the ASWPs algorithms. The values are percentage improvement over the width of the shortest paths, and are averaged over all the improvable pairs (no improvement in 1, 2 or 3-hop paths).

We notice that an increment of the k parameter augments the improvement. For instance, when the distance is 6 hops away, the average improvement at $k = 1$ (ASWP) is 16.4%. This value increases to be 24.4%, 29.1% and 31.2% as k changes to 2, 4 and ∞ (Figure 5, lower part). In general, the path width achieved by 2ASWP lies at about the middle between the path width by ASWP and the ideal. As expected, 4ASWP performs even better than 2ASWP, sometimes achieving the optimal solution.

It is difficult to predict whether a specific s/d pair is improvable. The answer depends not only on the network topology, but also on how the background flows are placed in the network. Indeed, the simulations show that in the medium loaded network, there are more improvable pairs than in the empty network. The results are shown in Figure 6. Comparing with the empty network case, the improvable ratio is 0.12 for 2-hops s/d pairs, and is 0.22 for 3-hop s/d pairs. For s/d pairs of 6 and 7 hop away, the improvable ratio is more than twice that of the empty network case.

What we learn from Figure 6 is similar to the previous figure. First, distant s/d pairs tend to be more improvable than close pairs. Second, the width of paths chosen by 2ASWP is larger than ASWP, and the width of paths by 4ASWP is even larger than 2ASWP.

B. Dynamic Simulations

The simulations in the previous section compare the performance by running different algorithms on a network with a specific value of load. In this section, we want to compare how the algorithms behave when flows are placed into the network dynamically, according to the path found by these algorithms. We use the proposed algorithms to route an entire sequence of flow requests, and compare the performance generated by different algorithms over the same sequence of requests.

1) Admission ratio: We choose five s/d pairs in the network, and generate fixed rate flow requests of 4 Kb/s. The requests come in at a rate of 0.32 flows per second, and are assigned uniformly to one of the five s/d pairs. If a flow is admitted, it will last a duration that is uniformly

distance	SP	ASWP	2ASWP	4ASWP
2 hops	99.4	100	100	100
4 hops	47.9	54.8	54.8	54.7
7 hops	31.8	44.1	43.4	43.9
mixed	66.5	71.4	71.0	70.9

TABLE I: Admission ratio of SP and the ASWPs algorithms in the dynamic simulations. The five s/d pairs are randomly chosen with distance consideration.

distributed between 400 and 2800 seconds. Thus on average the demand of the network is $0.32 \times \frac{400+2800}{2} \times 4 = 2048$ Kb/s, i.e., 2 Mb/s. We route the flows with SP, ASWP, 2ASWP and 4ASWP algorithms. Some flows are admitted and installed accordingly, the others are rejected because the paths are not wide enough. Notice that as the flows are installed, the network is utilized differently, since the ASWPs algorithms may find paths that are longer than the shortest paths.

The five s/d pairs are chosen randomly with distance consideration. Specifically, in the first row of Table I, the five s/d pairs are randomly chosen from all the s/d pairs that are 2 hops distant. Similarly for the second and third rows, the s/d pairs are chosen with 4 hops and 7 hops distances respectively. In the last row, the distances between the s/d pairs are mixed, with 2, 3, 5, 6, and 7 hops each.

We run the simulation over 10,000 flow requests. The purpose is to compare the admission ratio of these algorithms in long run. Table I presents the results: with the ASWPs algorithms we consistently admit more flow requests than using shortest paths. The improvement varies for different s/d pair sets, and is up to 12.3% for the 7-hops pairs. Indeed, the improvements are more significant when fewer flows can be admitted into the network – a feature that augurs well for utilizing these algorithms in congested scenarios. The results demonstrate that the ASWPs algorithms are good at finding paths for flow requests between fixed s/d pairs.

We also observe that 2ASWP and 4ASWP are not necessarily better than ASWP in the long run. As listed in Table I, 4ASWP admits 0.5% less flows than ASWP in the last row, while 2ASWP admits 0.7% less flows than ASWP in the 7-hops case. The implication is that pursuing the widest paths may not gain in the long run. If the chosen paths are too long, they consume more network resources, and affect future requests. Some sub-optimal paths selected by ASWP may in fact be better for the longer term. In short, it is useful to find wider paths than the shortest, but we must be cautious to adopt the extremely long paths.

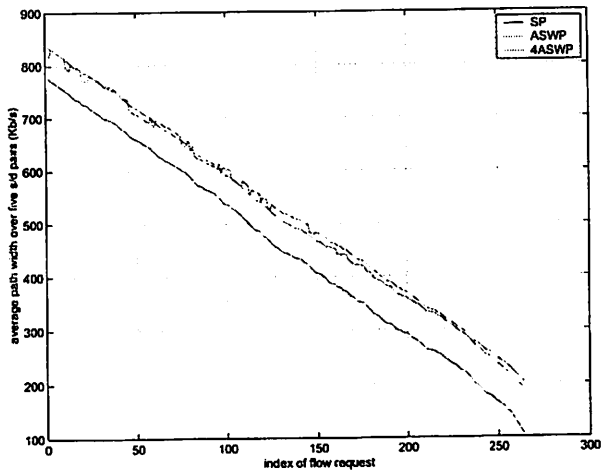

Fig. 7: Dynamic behavior of SP, ASWP and 4ASWP routing algorithms. The 4 Kb/s flows are generated between 5 s/d pairs (the mixed distance case as in the last row of Table I). The Y-axis shows the average path width of the 5 pairs; the X-axis is the indices of the flow requests.

2) Average path width: Similar results can be found by analyzing the path width. This time we install a sequence of flow requests into an empty network. There is no deletion of the installed flows. The five s/d pairs are the same ones as in the last row of Table I, with mixed distances. We plot in Figure 7 the average path width over the five s/d pairs. The X-axis is simply the indices of the flow requests. As each flow is installed, it changes the average path width available to the s/d pairs. We stop the simulation at flow #265, which is the first flow rejected by the SP algorithm.

Figure 7 shows that the path width decreases linearly as flows are admitted and installed. At any time, the ASWPs algorithms outperform SP by providing wider paths. The gap is about 60-70 Kb/s. Once again, 4ASWP performs a little worse than ASWP.

The average path width of the ASWPs algorithms are seen to *increase* sometimes – this is because the introduction of a new flow may cause the algorithms to look for a distant alternative path (not previously considered), which turns out to have more available bandwidth.

Notice that the dynamic simulations conducted in this section consider flow requests with small size (4 Kb/s) and long durations (hundreds of seconds). Therefore, the results expose the concerns that kASWP consumes too much network resources and affects future requests. Actually, the ASWPs algorithms, especially kASWP with large k value, are designed to find paths for large flow requests as long as they are feasible. We would expect

	SP	ASWP	2ASWP	4ASWP
# of update messages	12,900	14,293	16,146	22,032
running time (sec)	5.3	27.9	50.4	80.0

TABLE II: Time complexity for SP and ASWPs algorithms.

that if some of the flow requests are large (hundreds of Kb/s) and short-lived, the kASWP algorithm will further show its superiority.

C. Time Complexity

The time complexity of the proposed algorithms can be measured by counting how many times the relaxation operation (Sec. V-A) is performed. It is also the number of update messages that are sent between neighbor nodes. Besides this, the running time in seconds is also a straightforward measure of the complexity. However, it only serves as a rough estimate to the time complexity, because it depends on the programming language, code efficiency, and running environment. We list the measurements of a typical case in Table II. This is the case of the medium loaded network in Sec. VI-A.2. The algorithms are implemented and run in MATLAB 6.0 in a PC with 750 Mhz Pentium III.

We first focus on the ASWP algorithms and consider the impact of the k parameter. According to Table II, 2ASWP sends 1.1 times more update messages than ASWP, while 4ASWP sends 1.5 times more. These ratios are much less than the k values 2 and 4. However, the kASWP algorithms need to compare and sort the records during the relaxation operations. We thus expect larger running time than the number of update messages present. According to the table, 2ASWP takes 1.8 times longer time than ASWP, while 4ASWP is 2.9 times slower than ASWP. These numbers show that kASWP scales sub-linearly as k increases. Note that these numbers do not include the time of computing the maximal cliques (see [3] for more details).

Table II also compare the time complexity between the ASWPs algorithms and the Bellman-Ford shortest path algorithm. The running time of ASWP, 2ASWP and 4ASWP algorithms are about 5.3, 9.5 and 15.1 times slower than Bellman-Ford.

VII. Conclusion

We study the problem of computing bandwidth guaranteed paths for given flow requests in an ad-hoc network. The problem is complicated because neighboring links share the medium. To route a flow, one has to solve a feasible scheduling problem in addition to finding a route for the flow.

Three main ideas are presented in this paper. First, we separate the scheduling problem and quantify the feasibility of flows. By applying recent results from graph theory, we construct a set of clique constraints as the feasibility conditions of the proposed QoS routing mechanism. Second, we present the mathematical abstraction of the SWP problem that we want to solve. The idea behind the SWP problem is to find the maximum amount of traffic that a flow can carry along a single path. And third, we propose a distributed ASWP algorithm towards the SWP problem. It is further extended to a family of kASWP algorithms that achieve a performance close to the optimum.

The simulations over a 50-node network show the performance of the proposed algorithms. We differentiate the results based on the network load, as well as the hop distance between s/d pairs. The results show that for s/d pairs with large hop distance, the ASWPs algorithms improve the path width considerably. Moreover, the kASWP algorithm with larger k value is able to find paths closer to the optimum at the cost of longer running time.

The simulations offer us another curious insight – a comparison between the ASWP algorithm and the kASWP algorithm. We are aware that the kASWP algorithm keeps k times as much state as ASWP, and so takes more time and send more update messages. In a static situation, kASWP indeed outperforms ASWP and achieves results close to the optimum. Surprisingly though, when the algorithms are compared on the basis of their longer term performance, the ASWP results are nearly as good or better than those of kASWP. It suggests that ASWP probably offers the best trade-off to balance the optimality and long term performance. Choosing wider paths is certainly better, but the algorithm should be careful not to consume too much resources – for the sake of future demands.

References

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

[2] 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.

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

[4] H. Luo, S. Lu, and V. Bhargavan, "A New Model for Packet Scheduling in Multihop Wireless Networks", *Proceedings ACM Mobicom*, 2000.

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

[6] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for Ad Hoc Network," *Proceedings of The ACM Symposium on Mobile Ad Hoc Networking & Computing*, Long Beach, California, 2001.

[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] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", *W.H. Freeman and Company*, New York, 1979.

[17] M. Grotschel, L. Lovazs, A. Schrijver, "Geometric Algorithms and Combinatorial Optimization," *Springer*, 1993.

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

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

[20] R. Gupta, J. Musacchio and J. Walrand, "Sufficient Rate Constraints for QoS Flows in Ad-Hoc Networks", *Submitted to INFOCOM 2005. Available at http://www.eecs.berkeley.edu/~guptar/RGpublications.html.*

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

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

[23] 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.

[24] 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*