A Cross-layer Design for End-to-End On-Demand Bandwidth Allocation in Infrastructure Wireless Mesh Networks

Hong Lu, Steve Liu, Anxiao(Andrew) Jiang
Department of Computer Science
Texas A&M University
College Station, TX, 77840
{hongl,ajiang, liu}@cs.tamu.edu

Abstract

Wireless mesh networks have gained significant academic and industry attentions in the recent years. Supporting quality of service in wireless mesh networks is an important and challenging task which involves both medium access control and network layer design. In this paper, we investigate the problem of end-to-end on-demand bandwidth allocation in infrastructure wireless mesh networks. We formulate it as a combinatorial optimization problem, and prove that it is NP-hard. We present a polynomial time 2-approximation algorithm, MCRS(minimum consumption routing and scheduling), based on the concepts of consumption level *for routing and* bottom set *for scheduling. Comprehensive simulation results show that MCRS achieves better performance than traditional methods based on minimum hop routing.*

1 Introduction

WMN(Wireless mesh network), as a promising solution for building broadband home, community, enterprise, and even metropolitan networks, has gained significant academic and commercial attention in recent years. A WMN consists of two types of nodes: mesh clients and mesh routers. Mesh clients(PDA, cell phone, laptop) access WMN through mesh routers. Mesh routers interconnect with each other to form the network backbone. The network formed by mesh routers, called IWMN(infrastructure wireless mesh network), is the subject of study of this paper. It is uniquely characterized by: (a) multi-hop topology, as opposed to the single-hop topology of cellular networks or wireless LANs, (b) ad hoc networking, that is, the ability of self-forming, configuring and healing, (c) minimum mobility and reliable power supply: routers are typically fixed on building roofs or roadside poles. Many of the promised WMN applications such as video-on-demand, security surveillance, online gaming, etc depend on QoS support for end-to-end flows. Compared with other multi-hop ad hoc wireless networks such as MANET(mobile ad hoc networks) and WSN(wireless sensor networks), better support of QoS can be achieved in IWMN since mobility and power efficiency are not its main concerns.[1]

In this paper, we study the following on-demand bandwidth allocation problem: when a user's end-to-end connection request with a specific bandwidth requirement arrives, a QoS scheme allocates available network resource to satisfy this request while optimizing its bandwidth utilization such that more future requests can be satisfied. The major challenge of designing efficient bandwidth allocation schemes in IWMN is that it involves the cooperation of both MAC and network layer functions. Here, MAC is responsible for *scheduling* transmissions, that is, determining the specific time for each transmission to occur. Network layer is in charge of *routing*, that is, choosing the sequence of wireless links that will eventually carry the data. Traditionally, QoS aware network layer design adopts minimum hop routing and its variations, such as shortest widest path routing[20] and widest shortest path routing[3]. These routing algorithms are not suitable for wireless networks because they are designed for networks where links are physically isolated. Wireless links, however, are not independent from each other, and allocating bandwidth on one link does affect the available bandwidth of others. As a result, bandwidth allocation in IWMN involves both routing and schedul-

[1]Work reported in this paper is supported by NSF grants CNS-0530210, DUE-0516825 and an ARO grant W911NF-07-1-0178 . The contents of this paper do not necessarily reflect the position or policies of the U.S. government.

0-7695-2981-X/07 $25.00 © 2007 IEEE
DOI 10.1109/WASA.2007.9

ing that have to be solved in an integrated cross-layer framework.

The on-demand end-to-end bandwidth allocation problem considered in this paper is strongly tied to two related problems in multi-hop wireless networks: end-to-end available bandwidth and capacity region estimation. End-to-end available bandwidth estimation has been studied in [2, 12–15, 21]. Suppose certain amount of bandwidth has already been allocated to support existing requests and a new end-to-end flow request comes, the problem is to compute the amount of available bandwidth between the request's source and sink under the schedulability constraint that nodes cannot send and receive at the same time. [2] showed that this problem is NP-hard and [12–15, 21] proposed various heuristic solutions. They differ from our work in that their focus is to satisfy the need of the current request, while ours is to optimize bandwidth utilization so that more future requests can be supported. End-to-end capacity region estimation has been studied in [1, 4, 5, 7–9, 11]. The problem is to establish a given set of end-to-end flow requests within the network capacity region. In their settings, the set of requests are given as *a priori* while in ours, requests are assumed to arrive one by one in an unpredictable manner. Due to this difference, their linear programming based solution framework is not applicable to our on-demand settings.

In this paper, we present a combinatorial optimization formulation of the on-demand end-to-end bandwidth allocation problem in IWMN. We prove that it is NP-hard, and present a 2-approximation algorithm MCRS(minimum consumption routing and scheduling). Experiments show that MCRS significantly outperforms existing minimum hop based QoS schemes.

2 Network Model

We consider an IWMN formed by a group of homogeneous mesh routers with minimum mobility and reliable power supply. IWMN works on the basis of TDMA, where time is divided into equal length frames and each frame into equal length slots. Each router is equipped with a wireless transceiver. All the other routers that a router can directly communicate with(transmit to or receive from) successfully via its transceiver are called its neighbors. Communication between two non-neighboring routers is forwarded by one or more intermediate routers. There is a directed link from a router to each of its neighbors. All the wireless links between neighboring routers are assumed to have the same capacity. A transmission is a one hop communication that occurs at a particular link. As mentioned in the introduction, IWMN links are not independent from each other. Indeed, only transmissions satisfying certain *schedulability constraints* can successfully occur at the same time. The first type of schedulability constraint is what we call the *single transceiver constraint*: a set of transmissions can successfully occur at the same time only if no two transmissions share a common sender or receiver. This constraint is caused by the fact that each IWMN router is equipped with only one radio transceiver, and thus a node cannot transmit to two different receivers or receive from two different senders at the same time. The impact of single transceiver constraint to routing and scheduling is the focus of our study. The other type of schedulability constraint is caused by radio interference: a set of transmissions can successfully occur at the same time if the signal to interference ratio at the receiver of each transmission is above a given threshold. Here, interference comprises of ambient noise and the combined signal of all other transmissions. Although the effect of interference is an important concern in wireless networks, it is not the focus of our paper and will be omitted in the following discussion. This simplification is based on the fact that interference can be dramatically suppressed with technologies such as directional antenna[6] and adaptive transmission power control[17]. Furthermore, interference can also be eliminated by a well-studied preprocessing step called channel assignment[10, 18, 19], which assigns interfering links to orthogonal channels that are mutually isolated.

3 Problem Formulation

The IWMN under consideration is represented by a directed graph $G = (V, E)$, where V is the set of IWMN routers, and $E \subseteq V \times V$ is the set of links between them. Let $e = uv \in E$ be a link, u and v are respectively called the sender and receiver of e. Given a node v, $in(v)$ and $out(v)$ respectively denote the set of inbound and outbound links incident on v. For two nodes $s, d \in V$, an $s \rightarrow d$ path p is a sequence of adjacent links connecting s and d. If $e = uv$ is a link of path p, we say $u \in p$, $v \in p$, and $e \in p$. Note that we do not assume disk graph based network model in our discussion.

The set of slots in a frame is denoted by $T = \{1, 2, \ldots, |T|\}$. The k-th slot of link e is referred to as t_e^k. Each t_e^k is an atomic reservable bandwidth unit. Link e_1 conflicts with e_2, or $e_1 \nparallel e_2$, if they share a common sender or receiver. Notice that, $e_1 \nparallel e_2$ if $e_1 = e_2$ by this definition. Slot $t_{e_1}^{k_1}$ conflicts with $t_{e_2}^{k_2}$, if $k_1 = k_2$ and $e_1 \nparallel e_2$. Each individual slot can be in one of the

following three states: *allocated*, *occupied*, or *free*. An allocated slot is one that is reserved to carry a unit of flow traffic. An occupied slot is one that an allocated slot conflicts with, hence cannot be used for data transmission. A free slot is one that is neither allocated nor occupied. The available bandwidth of a link is the set of free slots on that link. A slot is *consumed* if it is either allocated or occupied.

The state S of an IWMN G is a triple (A, O, F) where function $A : E \rightarrow T$ specifies the set of allocated slots on each link. The sets of occupied and free slots are specified by functions O and F respectively. A state S is conflict free if no two allocated slots conflict with each other. A, O, and F are obviously related. Given G and $\nparallel$, O and F can be derived from A. Therefore, only A is the essential component of a conflict free state. The total number of free slots,

$$\sum_{e \in E} |F(e)|,$$

characterizes the available network bandwidth that can be allocated to support users' requests. Let $s \overset{b}{\rightarrow} d$ denote a flow request between source s and sink d asking for b free slots. An $s \overset{1}{\rightarrow} d$ request is called a unit request.

An $s \overset{b}{\rightarrow} d$ flow *arrangement* A_p is specified by an $s \rightarrow d$ path p, and b time slots to be allocated on each link e of p. The set of b slots to be allocated on link $e \in p$ is denoted by $A_p(e)$. A flow arrangement A_p is *feasible* with regard to state $S = (A, O, F)$ if $A_p(e) \subseteq F(e)$ for every link e of p. Notice that, if S is conflict free and A_p is feasible, then the new network state $S' = (A', O', F')$ after the allocation of A_p is also conflict free. Here, A' is defined by $A'(e) = A(e) \cup A_p(e)$ for every $e \in p$, and $A'(e) = A(e)$ for every $e \notin p$. Recall that, O' and F' can be easily derived from A'. Unless explicitly declared, only feasible arrangements are considered in the rest of this paper.

Given state S, the consumption set $C(S, t_e^k)$ of a free slot t_e^k is defined to be $\{t_{e'}^k | t_{e'}^k \in F(e'), e \nparallel e'\}$, that is, the set of free slots that t_e^k conflicts with. The consumption set of a flow arrangement A_p, denoted by $C(S, A_p)$ is defined as $\bigcup_{e \in p} C(S, A_p(e))$, where $C(S, A_p(e)) = \bigcup_{t_e^k \in A_p(e)} C(S, t_e^k)$ is the set of free slots to be consumed by allocation of $A_p(e)$. Note that, $t_e^k \in C(S, t_e^k)$ and $A_p(e) \subseteq C(S, A_p(e))$ by the above definition. When S is understood, $C(S, A_p(e))$ and $C(S, A_p)$ will be abbreviated as $C(A_p(e))$ and $C(A_p)$. With the above definitions and notations, we formulate the end-to-end on-demand bandwidth allocation problem under consideration as follows.

Problem 1 (EEOBA) *Given network G, its state S and flow request $s \overset{b}{\rightarrow} d$, find a feasible flow arrangment A_p if there is any, such that $|C(S, A_p)|$, the number of slots to be consumed by A_p, is minimized.*

Theorem 1 *Problem EEOBA is NP-hard.*

The proof of Theorem 1, as well as Theorem 2 in the next section, is omited due to space limit. Interested readers are refered to [16].

4 Minimum Consumption Routing and Scheduling

4.1 An Illustrative Example

Before getting into the detailed discussion of MCRS, we first use a toy network in Fig 1 to illustrate the basic idea of the routing sub-algorithm of MCRS. To simplify the discussion, we assume each frame has only one slot, so that establishing a unit flow connection only involves routing.

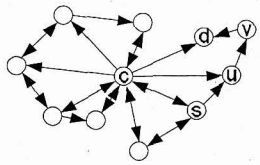
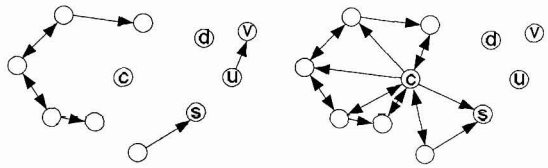


Figure 1. A simple network

Initially, all links are available. Suppose a unit flow between s and d is to be established. Two obvious routing choices are: *scd* and *suvd*, of which *scd* is the one with minimum number of hops. If *scd* is chosen, then link *sc* and *cd* will be allocated, and all the links in $out(s)$, $in(c)$, $out(c)$ and $in(d)$ are to be occupied. The residual network after the allocation of *sc* and *cd* is shown in Fig 2(a), where both *sc* and *cd*, as well as the links conflicting with them are left out. Similarly, the residual network after the allocation of *suvd* is shown in Fig 2(b). Clearly, network (a) is "less connected" than (b), and hence has a poorer chance to support future requests.

This simple example shows, unlike in wired networks, choosing the path with minimum hops, *scd* in this case, does not necessarily lead to minimum bandwidth consumption in wireless network. A better strategy would be to avoid choosing node with high degree, such as node c in Fig 1. Here, the degree of a node is a dynamic quantity, which keeps changing as flow requests are processed. For example, the degrees of c



(a) path *scd* is chosen (b) path *suvd* is chosen

Figure 2. Residual networks

in Fig 1 and Fig 2 are different. The same observation can also be made from the perspective of links: choosing a link $e = uv$ would result in $|out(u) \cup in(v)|$ links to be consumed, thus one would like to avoid choosing links with high $|out(u) \cup in(v)|$ value in order to reduce bandwidth consumption.

The above observation is based on the assumption that each frame has only one slot. Algorithm MCRS in the next subsection further develops this idea for the general case where a frame has more than one allocatable slots.

4.2 Minimum Consumption Routing and Scheduling

Algorithm MCRS relies on the following two concepts, assuming network G and state $S = (A, O, F)$ are given.

Definition 1 *The b-th* bottom set *of a link e, denoted by $B_b(S, e)$, is the set of b free slots of e with minimum consumption sets if $|F(e)| \geq b$, or Φ if $|F(e)| < b$.*

Definition 2 *The* consumption level *of a free slot t_e^k, denoted by $c(S, t_e^k)$, is the size of its consumption set $|C(S, t_e^k)|$. The b-th consumption level of link e, denoted by $c_b(S, e)$, is defined as $|C(S, B_b(e))|$ if $|F(e)| \geq b$, or ∞ if $|F(e)| < b$. In other words, the b-th consumption level of link $e = uv$ is the minimum number of slots that will be consumed to support a $u \xrightarrow{b} v$ flow.*

When S is understood, $B_b(S, e)$, $c(S, t_e^k)$, and $c_b(S, e)$ will be abbreviated as $B_b(e)$, $c(t_e^k)$, and $c_b(e)$. With these definitions, we present our routing and scheduling algorithm, MCRS, as shown in Algorithm 1.

Basically, MCRS uses the b-th consumption level for routing(step 1, 2) and b-th bottom set for scheduling(step 4, 5). This method of routing and scheduling will hereafter be called MCR(minimum consumption routing) and MCS(minimum consumption scheduling) respectively. The flow arrangement A_p is obviously feasible since $B_b(e) \subseteq F(e)$, thus the new network state after the allocation of A_p is conflict free.

Algorithm 1 Minimum Consumption Routing and Scheduling

Input: network graph $G = (V, E)$, state $S = (A, O, F)$, flow request $s \xrightarrow{b} d$

Output: a feasible $s \xrightarrow{b} d$ flow arrangement A_p if there is one

1: For each $e \in E$, compute the b-th consumption level $c_b(e)$
2: Use $c_b(e)$ as the cost of $e \in E$ to find a shortest $s \rightarrow d$ path p by Bellman-Ford's algorithm
3: If p does not exist or the cost of p is ∞, then return "failed", else
4: For each link $e \in p$, compute the b-th bottom set $B_b(e)$
5: Let $A_p(e) = B_b(e)$, and return A_p

The running time of algorithm MCRS is dominated by step 1, 2 and 4. Step 1 computes the b-th consumption level for each link $e \in E$, which can be done in time $O(b|E||T|)$ in the following two substeps. In the first substep, we compute the consumption level of every free slot as follows. For a free slot t_e^k, its consumption level $c(t_e^k) = c_{in}(t_e^k) + c_{out}(t_e^k) + 1$. Here, $c_{in}(t_e^k)$ and $c_{out}(t_e^k)$ are the number of free slots $t_{e'}^k$, $e \neq e'$, such that e and e' share the same receiver and sender respectively. The number "1" corresponds to slot t_e^k itself. Notice that, slots t_e^k and $t_{e'}^k$ have the same c_{in} value if e and e' share the same receiver. Based on this fact, the c_{in} value for all the free slots can be computed in time $O(|E||T|)$. Similarly, the c_{out} value and hence the consumption level for all the free slots can also be computed in time $O(|E||T|)$. Then in the second substep, we spend another $O(b|T|)$ time for each link $e \in E$, that is, $O(b|E||T|)$ for all the links, to find the b slots with minimum consumption levels for each link and compute the b-th consumption level. Therefore, step 1 takes time $O(b|E||T|)$ in total. Step 2 runs Bellman-Ford's shortest path algorithm which takes time $O(|E||V|)$. Step 4 computes the minimum consumption schedule for every link of the path found by step 2. By an analysis similar to that of step 1, step 4 takes time $O(b|E||T|)$. In summary, Algorithm MCRS takes time $O((|V| + b|T|) \cdot |E|)$.

The NP-hardness of EEOBA shows that an optimal arrangement cannot be found by an polynomial time algorithm, assuming P is not equal to NP. Nevertheless, algorithm MCRS provides a solution whose cost is guaranteed to be no more than twice the cost of an optimal solution.

Theorem 2 *MCRS is a 2-approximation algorithm.*

4.3 Stateless Variations

As one of the two components of MCRS, MCS is amenable to efficient distributed implementation. The main task of MCS is to compute $B_b(e)$ for every link $e = uv$ of the path p chosen by MCR. To do this, we can simply let u collect the state of links in $out(u) \cup in(v)$, and compute $B_b(e)$. In this way, the computation of MCS can be performed concurrently using only local information.

MCR, based on Bellman-Ford's algorithm, on the other hand, is relatively harder to implement in an efficient way. A common method of implementing Bellman-Ford's algorithm is to precompute and cache routing decisions in a table. This method may not be suitable for MCR because it requires up-to-date network state information, which may change frequently as flow requests arrive and leave.

To address the above issue, we propose the following stateless version of MCR, called MCR$^-$. Specifically, we use $|out(u) \cup in(v)|$ for every $e = uv \in E$ instead of $c_b(e)$ as the cost function for routing. This idea of MCR$^-$ has actually already been illustrated in the example in Section 4.1. The intuition is essentially the same as that of MCR, that is, to avoid consuming "highly connected" nodes as much as possible. Note that, MCR$^-$ only requires the network topology as input, that is, it needs not to be aware of the current network state. Therefore, routing decisions can be precomputed efficiently by distributed Bellman-Ford's algorithm. Stateless algorithm does not provide any upper bound on the amount of bandwidth to be consumed. The actual performance of stateless algorithm will be experimentally studied in Section 5.

5 Simulation Evaluation

To evaluate the performance of our scheme, we perform a series of simulations on an IWMN with 200 routers randomly distributed in a 500×500 rectangular plane area. For each router, we assign a random(Gaussian) transmission range with mean and variance set to 100 and 50. A router can communicate unidirectionally with any router within its transmission range. The number of slots per frame is set to be 50. Note, although this simulation setting implies a disk graph based network model, MCRS is designed for general graph models. We take this simulation design mainly because it is a widely adopted set up, and we use it as a benchmark to test performance.

To the authors' best knowledge, no other non-trivial QoS routing and scheduling algorithm has been proposed in the literature. Thus, in our experiments, we fix the scheduling algorithm to be MCS, and compare the performance of MCR and MHR, and their stateless versions, MCR$^-$ and MHR$^-$. Here, MHR is acronym for minimum hop routing, a widely used QoS routing algorithm for wired networks. Note, when we speak of the performance of one of the four routing algorithms, we are really talking about the performance of its combination with MCS. For MHR, we assign a cost of 1(one hop) to link e, $\forall e \in E$, if $|F(e)| \leq b$, or ∞ if $|F(e)| > b$. For MHR$^-$, we simply let the cost of each link to be 1 regardless of the current network state. For convenience, MCR and MCR$^-$ will be collectively called min-consumption algorithms, and MHR and MHR$^-$ min-hop algorithms.

5.1 Static experiment

In the first experiment, we load the IWMN with static requests. Once a static flow request is accepted and established, it stays in the network indefinitely, and the allocated bandwidth is not recycled. We load the IWMN with 5000 such unit requests one by one, each of which is assigned a random source and sink. We observe the bandwidth and delay performance, as well as load balancing performance for each algorithm.

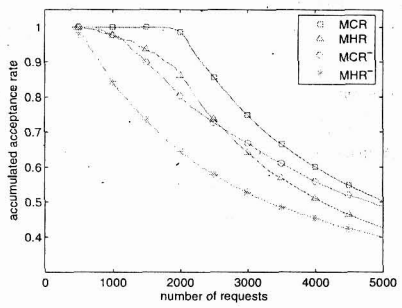


Figure 3. Accumulated acceptance rate

Fig 3 plots the accumulated acceptance rate, y, after x requests are loaded, $0 < x \leq 5000$, that is, $y = a(x)/x$, where $a(x)$ is the number of accepted requests out of the x loaded ones. As shown in this figure, MCR maintains a 100% acceptance rate up to 1537 requests, while this number for MHR, MCR$^-$, and MHR$^-$ is 406, 742, and 401 respectively. We take the point x up to which an algorithm maintains a 100% acceptance rate as a simple measurement of that algorithm's bandwidth performance. By this measurement, the bandwidth performance of MCR is more than 4 times of that of MHR in this experiment(in all our other experiments, the bandwidth performance

of MCR is consistently at least 3 times that of MHR). It is interesting to notice that the stateless algorithm MCR$^-$ exhibits better performance than the stateful algorithm MHR. In Fig 3, both y_{MCR} and y_{MHR} keep dropping monotonously when more and more requests are loaded, but y_{MCR} is consistently higher than y_{MHR}. In the end, MCR accepted 2511 out of 5000 requests, which is 17.9% more than MHR. y_{MCR^-} and y_{MHR^-} exhibit the same pattern. Interestingly, y_{MCR^-} falls below y_{MHR} at $x \approx 1000$, but it ends up to be higher than y_{MHR} when x approaches 5000.

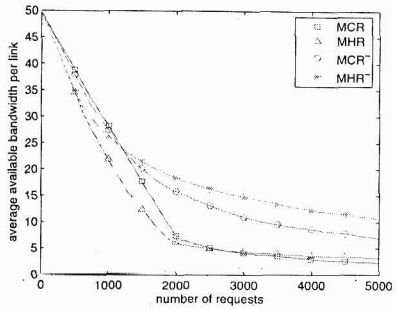


Figure 4. Available bandwidth

Fig 4 shows the average number of free slots per link, y, after x requests are loaded, $0 < x \leq 5000$. As expected, $y_{\text{MCR}} > y_{\text{MCR}^-} > y_{\text{MHR}} = y_{\text{MHR}^-}$ when x is small, which shows min-consumption algorithms consumes less bandwidth than min-hop algorithms when the same amount of requests are accepted. Here, $y_{\text{MHR}} = y_{\text{MHR}^-}$ because MHR and MHR$^-$ choose different paths only after certain shortest paths are saturated. Notice that, $y_{\text{MCR}} < y_{\text{MHR}} < y_{\text{MCR}^-} < y_{\text{MHR}^-}$ when x approaches 5000. This is not surprising because as shown in Fig 3, min-consumption algorithms have accepted more requests than min-hop algorithms at this moment. y_{MCR} has an abrupt turn at $x \approx 2000$. Before this point, y_{MCR} decreases quickly and almost linearly as requests are loaded one by one, and after this point, y_{MCR} begins to decrease at a very low speed. Similar point can be observed for y_{MHR} at $x \approx 1900$, although the turn is less abrupt than that of y_{MCR}. We call this point the saturation point in that it roughly indicates the moment when the network bandwidth is exhausted. No obvious saturation points can be observed for y_{MCR^-} and y_{MHR^-}. The existence of saturation point for stateful algorithms shows that they are more aggressive at bandwidth allocation than stateless algorithms.

Next, we evaluate the delay performance of each algorithm, where delay is measured by the number of

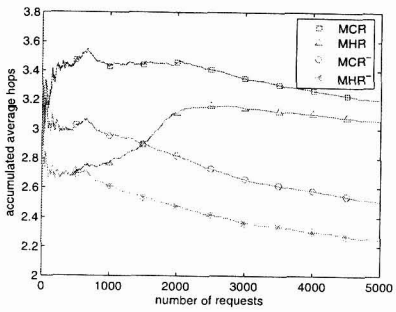
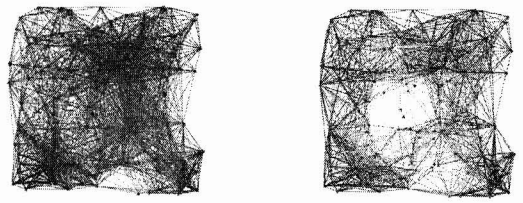


Figure 5. Accumulated average hops

hops of the path computed for each accepted request. In Fig 5, We show the accumulated average hops, y, after x requests are loaded, where $y = \sum_{i=1}^{x} h(i)/a(x)$ with $h(i)$ and $a(x)$ defined as follows. If request i is accepted, then $h(i)$ is the number of hops of the path found by each routing algorithm; if request i is rejected, then $h(i) = 0$. $a(x)$ again is the number of accepted requests out of x loaded requests. At first when x is small, $y_{\text{MCR}} > y_{\text{MCR}^-} > y_{\text{MHR}} = y_{\text{MHR}^-}$. This indicates that min-hop based algorithms have better delay performance than min-consumption based algorithms, which conforms to their design objective to reduce delay. Here, $y_{\text{MHR}} = y_{\text{MHR}^-}$ is again because MHR and MHR$^-$ make the same routing decision when no path is saturated. When more requests are loaded, more and more min-hop paths are saturated, and MHR is forced to pick sub-shortest paths. This is why y_{MHR} keeps increasing until it reaches a peak at the saturation point of $x \approx 1900$. After that, y_{MHR} begins to keep dropping, because when the network becomes more and more saturated, it is only capable of supporting shorter and shorter requests. In comparison, y_{MCR} maintains a value of around 3.4 up to the saturation point of $x \approx 2000$. After that, y_{MCR} starts to keep dropping (although still higher than y_{MHR}). The reason that y_{MCR} does not follow the climbing-descending pattern as that of y_{MHR} is because MCR is better at load balancing(to be further illustrated shortly): when a short path is saturated, so are the longer ones. Finally, just as MHR versus MCR, the delay performance of MHR$^-$ is always better than that of MCR$^-$. No hill climbing is observed for both y_{MCR^-} and y_{MHR^-} because they are stateless: no longer paths are tried when shorter ones are saturated.

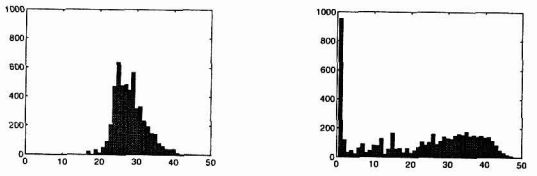
To study the load balancing abilities of min-consumption and min-hop algorithms, we examine the network for each algorithm after the first 1000 requests are loaded, as shown in Fig 6. Here, 1000 is an arbi-



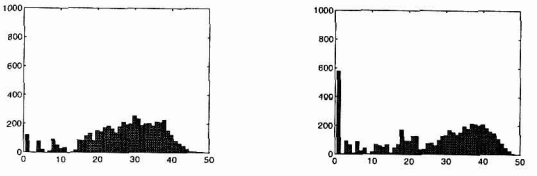
(a) MCR (b) MHR

(c) MCR$^-$ (d) MHR$^-$

Figure 6. Residual networks after 1000 requests



(a) MCR (b) MHR



(c) MCR$^-$ (d) MHR$^-$

Figure 7. Distribution of available link bandwidth after 1000 requests

trarily chosen number. We performed the same studies for operation points other than 1000, and all the studies produced similar results. In Fig 6, we use darkness to indicate the amount of available bandwidth of each link. The darker a link is, the more bandwidth is available. From the figure we can see that the available bandwidth distributions of network (a) and (c) are more uniform than those of (b) and (d). This is because min-hop algorithms choose the shortest path for each request. Since the source and sink node of each request is random, the nodes sitting at the center of the network are more likely to be loaded first. This is not the case for min-consumption algorithms, however, because they do not particularly favor short paths.

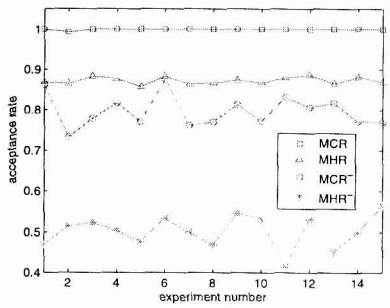
Fig 7 plots the bandwidth distributions of the four networks in histograms. That is, this figure plots $y = d(x)$ for each algorithm, where $d(x)$ is the number of links with x available slots, $0 \leq x \leq 50$. In the four networks, the bandwidth spectrum of the MCR network is the narrowest. More accurately, the variances for MCR, MHR, MCR$^-$, and MHR$^-$ are 4.16, 14.83, 9.80, and 14.15 respectively, which shows that the load balancing abilities of min-consumption algorithms are significantly better than min-hop algorithms.

5.2 Mixed experiment

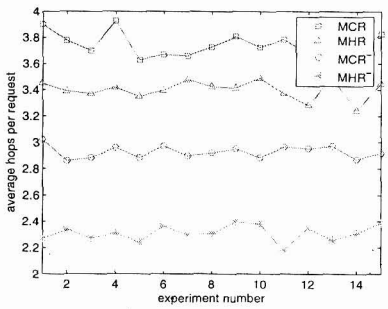
In this experiment, we consider both static and dynamic requests. Dynamic flows have limited life-time. After they expire, the allocated bandwidth can be recycled to support other new flows. The arrival time of each dynamic request is modeled as a Poisson process, where the average inter-arrival time is set to 0.5 second. The life-time of a dynamic flow is modeled as a random variable with uniform distribution from 0 to 5 seconds. We first load the network with 1000 static unit requests, and then another 4000 dynamic unit requests. We perform 15 such experiments, each with a randomly generated network with the same parameters as those of the static experiment, and a set of mixed static and dynamic requests. Fig 8(a) shows the acceptance ratio of the 4000 dynamic requests for each of the 15 experiments. Fig 8(b) shows the average number of hops for the 4000 dynamic requests for each of the 15 experiments. Results confirm again that min-consumption algorithms have better bandwidth performance while min-hop algorithms have better delay performance.

6 Conclusion and Future Work

In this paper, we study the problem of end-to-end on-demand bandwidth allocation in infrastructure wireless mesh networks. We prove that this problem is NP-hard and propose a 2-approximation algorithm, MCRS. Our scheme exhibits better bandwidth



(a) acceptance rates



(b) average hops per request

Figure 8. Mixed experiment

efficiency than traditional methods in simulations.

References

[1] M. Alicherry, R. Bhatia, and L. E. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking(Mobicom)*, pages 58–72, 2005.

[2] C. Ferguson. *Routing in a Wireless Mobile CDMA Radio Environment.* PhD thesis, Computer Science Department, University of California, Los Angeles.

[3] R. A. Guerin, A. Orda, and D. Williams. Qos routing mechanisms and ospf extensions. In *IEEE Global Telecommunications Conference(Globecom)*, Phenix, AZ, 1997.

[4] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *ACM/IEEE Transactions on Information Theory*, 34(5), 1988.

[5] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking(Mobicom)*, pages 66–80, 2003.

[6] J. Joseph C. Liberti and T. S. Rappaport. *Smart antennas for wireless communications.* Prentice Hall PTR, 1998.

[7] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *Proceedings of the 9th annual international conference on Mobile computing and networking(Mobicom)*, pages 42–54, 2003.

[8] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless with orthogonal channels. *ACM/IEEE Transactions on Networking*, 2005.

[9] M. Kodialam and T. Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking(Mobicom)*, pages 73–87, 2005.

[10] S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, 7(6):575–584, 2001.

[11] V. A. Kumar, M. V. Marathey, S. Parthasarathyz, and A. Srinivasan. Algorithmic aspects of capacity in wireless networks. In *International Conference on Measurement and Modeling of Computer Systems(Sigmetrics)*, 2005.

[12] C. R. Lin. Admission control in time-slotted multihop mobile networks. *IEEE Journal on Selected Areas in Communications*, 19(10), 2001.

[13] C. R. Lin. On-demand qos routing in multihop mobile networks. In *IEEE Conference on Computer Communications(Infocom)*, Anchorage, AK, 2001.

[14] C. R. Lin and J. Liu. Qos routing in ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8), 1999.

[15] H. C. Lin and P. C. Fung. Finding available bandwidth in multihop mobile wireless networks. In *IEEE Vehicular Technology Conference(VTC)*, 2000.

[16] H. Lu, S. Liu, and A. Jiang. End-to-end bandwidth provision in multihop wireless networks. Technical report, Department of Computer Science, Texas A&M University, TX, 2005.

[17] R. Ramanathan and R. Hain. Topology control of multihop wireless networks using transmit power adjustment. In *IEEE Conference on Computer Communications(Infocom)*, pages 404–413, 2000.

[18] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5, 1999.

[19] S. Ramanathan and E. L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transaction on Networking*, 1(2):166–177, 1993.

[20] Z. Wang and J. Crowcroft. Qos routing for supporting resource reservation. *IEEE Journal on Selected Areas in Communications*, 14(7), 1996.

[21] C. Zhu and M. S. Corson. Qos routing for mobile ad hoc networks. In *IEEE Conference on Computer Communications(Infocom)*, June 2002.