Wireless Netw (2011) 17:779–796
DOI 10.1007/s11276-010-0314-7

Flow admission control for multi-channel multi-radio wireless networks

XuFei Mao · Xiang-Yang Li · GuoJun Dai

Published online: 25 December 2010

Abstract Providing Quality of Service (QoS) is a major challenge in wireless networks. In this paper we propose a distributed call admission control protocol (DCAC) to do both bandwidth and delay guaranteed call admission for multihop wireless mesh backbone networks, by exploiting the multi-channel multi-radio (mc-mr) feature. We propose a novel routing metric for route setup, and present an efficient distributed algorithm for link reservation that satisfies the required bandwidth and reduces the delay by a local scheduling that minimizes one hop delay. To the best of our knowledge, this is the first distributed protocol that embeds mc-mr feature in Time Division Medium Access (TDMA) to do QoS call admission in wireless backbone networks. Extensive simulation studies show that our protocol significantly improves network performance on supporting QoS sessions compared with some widely used protocols.

Keywords Admission control · Delay · Bandwidth · Multi-channel · Multi-radio · Mesh networks · Link scheduling

X.-Y. Li (✉) · G. Dai
Institute of Computer Application Technology, Hangzhou Dianzi University, Hangzhou, People's Republic of China
e-mail: xli@cs.iit.edu

G. Dai
e-mail: daigj@hdu.edu.cn

X. Mao
Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing, People's Republic of China
e-mail: maoxufei@bupt.edu.cn

X.-Y. Li
Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

1 Introduction

With the emergency of real-time applications, the problem of providing QoS has received significant attention in the past years. One important domain, where providing QoS is essential, is infrastructure wireless backbone networks, such as wireless mesh network (MWN) [1]. This kind of wireless network provides backbone service between a large number of mobile clients and the wired networks. In this paper we focus on exploiting the multi-channel multi-radio (mc-mr) features to improve network performance on providing both bandwidth and latency guaranteed QoS call admission for wireless backbone networks.

The backbone network consists of three types of infrastructure routers, namely, access points (AP), gateways, and relay routers. These wireless routers prevalently are IEEE 802.11 [2] compliant and do not have power constraints. Channelization was added to IEEE 802.11 to increase the network capacity, e.g., IEEE 802.11a and IEEE802.11b offer 3 and 12 non-overlapping channels, respectively. Several companies such as MeshDynamics [3] have announced the availability of multi-radio routers, that is equipping each router with multiple network interface cards (NIC). Tuning radios to non-overlapping channels greatly alleviates the close-by interference, as simultaneous communication flows can be distributed across channels. While the limited number of non-overlapping channels cannot totally eliminate collisions, the radios at each router are capable of fast switching among channels to improve the performance, with a switching overhead. Typically this overhead is around 80 μs [4]. In this paper, we propose a new distributed call admission control protocol **DCAC** that will utilize the fast channel switching ability of radios to provide an enhanced QoS service. Our protocol counts the switching overhead to dynamically match radio with channels. In our study, we assume that the network supports both QoS and best effort traffics. Our admission protocol is MAC layer aware routing protocol, which performs route discovery and route reservation at the same time, according to a new proposed routing metric. Our admission control will carefully schedule activities of radios such that it relieves the inter-flow and intra-flow interferences and avoids potential network congestion. In our proposed DCAC protocol, we will mainly manage the following four aspects of QoS sessions: (1) Route discovery and reservation; (2) Route set up failure; (3) Route breakage and maintenance; and (4) Route release.

We use a hybrid scheme for channel assignment, which fixes one radio of each router to a common channel, and allows other radios to dynamically switch among other channels. The common channel will be used as a control channel for call admission control, reservation, and synchronization for data channels. Compared with approaches that do not have a fixed control channel, this has three advantages: (1) ensures the network connectivity, otherwise network partition may be induced by channel assignment if it is not carefully designed; (2) provides admission control and maintenance for QoS sessions; (3) relieves the stringent synchronization requirement, which is particularly hard to provide in multihop wireless networks [4]. The remaining bandwidth of the control channel will also be used by best effort traffic, which also compensates the performance reduction due to dedicated control channel. Other data channels are used by QoS traffic. To avoid *control channel saturation*, a bottleneck identified in prior work [4], our protocol transmits the control packets for established or partial established sessions by fully utilizing their reserved resources on data channels.

Our protocol does not restrict on stringent synchronization. Data transmissions do require local synchronization, since the MAC of each data channel is TDMA based. We embed the initial synchronization information into the control packet when doing route setup. For the on-going sessions, when the synchronization is not accurate, we can use other shared data channel if there exists one, or rely on the control channel to do re-synchronization. Lack of synchronization will be temporary and can always be adjusted easily through control channel.

The main contributions of this paper are as follows.

Distributed call admission protocol (DCAC) for multi-channel multi-radio wireless networks: To our best knowledge, this is the first distributed protocol for QoS call admission for multi-channel multi-radio (mc-mr) multihop wireless backbone network with large network size. Our protocol is based on the TDMA activity scheduling and does not need any traffic profile. By fully exploiting mr-mr configurations, our protocol significantly improves network performance on supporting QoS sessions. The radio reserved for data channels will use a TDMA MAC. At each time-slot, a radio will decide whether to transmit or listen or be idle; it will decide on which channel it will transmit/listen and to which next-hop router to transmit. We generally assume that each session will have the QoS requirement on the needed bandwidth and the delay requirement. Notice that to ensure the bandwidth achieved by a flow in wireless networks, we need to address both the intra-flow interferences and inter-flow interferences. The delay of a session is closely dependent on the nodes' scheduling, which is not a simple summation of the delay by each link.

Efficient one hop reservation algorithm for bandwidth and delay requirements: Very little work has been done to deterministically guarantee delays in multihop wireless networks, except [5]. In [5], Lee et al. proposed a centralized algorithm for admission control based on the whole network traffic demands, i.e., it assumes the traffic flows of the whole network is known in advance. Previous work (e.g., [6, 7]) on QoS had mainly focused on bandwidth guarantee. In this paper, we first present a novel definition for one hop delay, which precisely captures the joint impact of channel assignment and scheduling. Based on this, we propose an optimal one hop reservation algorithm to guarantee both bandwidth and delay requirements. We also conducted extensive simulations to study the performance of our proposed DCAC protocol, compared with some widely used MAC protocols. Extensive simulation studies show that our protocol significantly improves network performance of guaranteed bandwidth and delay. The average delay of sessions supported by our DCAC protocol is about 1/2 of that by AODV routing protocol, while the throughput of the network is about two times of that by AODV routing protocol. The number of sessions that can be admitted is also about two times that supported by AODV routing protocol. We also compared our protocol with M-AODV, where we run AODV protocol for each possible channel. Our simulation studies show that our protocol will ensure a delay that is about 66% of the delay achieved by the M-AODV protocol. The capacity achieved by our DCAC protocol is about 5/3 times of the capacity achieved by the M-AODV protocol. Futhermore, our protocol always admits more sessions than the M-AODV protocol.

The rest of the paper is organized as follows. In Sect. 2, we briefly review the related work. In Sect. 3, we discuss the network model, interference model, and formally define the problem we studied in this paper. Our distributed call admission control protocol is presented in Sect. 4, and we discuss in detail the method for a link to reserve time-slots and channels for a flow in Sect. 5. Our simulation studies are reported in Sect. 6. We conclude our paper in Sect. 7.

2 Related work

A large amount of work has explored call admission control algorithms in the wireline network. The most popular QoS scheduling schemes are based on Generalized Processor Sharing (GPS) [8], which was first proposed as Weighted Fair Queueing (WFQ) in [9]. Papers [10–12] studied the problems of obtaining a delay bound and backlog for a session in GPS schedulers, given a particular bandwidth allocation. Various algorithms were presented in [13–15], to meet the sessions' delay requirements and to allocate bandwidth in GPS schedulers. A measurement-based admission control algorithm to increase network throughput for soft delay guarantee was presented in [16]; a survey of admission control algorithms was reported in [17].

In the wireless domain a class of delay-aware QoS schedulers have been proposed for single-hop cellular networks [18–20]. Delay and rate aware scheduling in the context of 802.11-based multihop networks has been proposed [21] using a probabilistic approach with priority backoff. [22] proposed a scheduling and routing framework that provided upper bounds on end-to-end delays in multihop wireless networks. [5] proposed centralized call admission algorithms with respect to both rate and delay requirements. They required a traffic profile with all traffic demands known in prior, and restricted on tree-based pattern topology.

The broadcast feature of wireless radios which operate on a shared communication medium made QoS more difficult to achieve in wireless networks than in wired networks. The reduction of shared channel capacity due to interferences is a major problem in the wireless networks. The interference include the inter-flow interference and

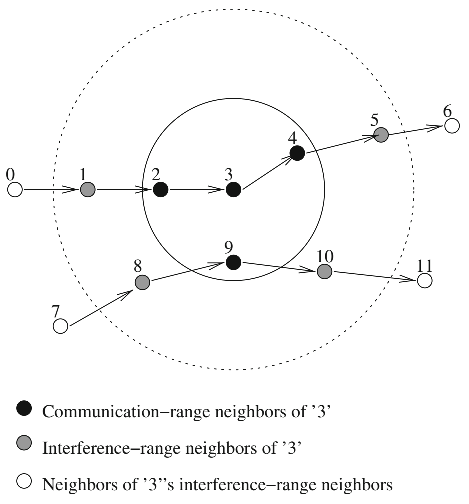

0
1
2
3
4
5
6
7
8
9
10
11
● Communication−range neighbors of '3'
● Interference−range neighbors of '3'
○ Neighbors of '3''s interference−range neighbors

Fig. 1 Inter-flow and intra-flow interference example in multihop wireless network

intra-flow interference [23, 24]. For example, in a network shown in Fig. 1, nodes 1, 2, 4, 5 are in the interference range of node 3, and cannot transmit/receive when node 3 is active for transmitting. Nodes 8–10 belonging to another node-disjoint flow also fall in the interference range of node 3. Thus none of the wireless links shown in the figure can simultaneously operate when node 3 is transmitting to node 4. In other words, the capacity of a link is not exclusively owned by this link; it is shared in certain fashion by a number of close-by links.

Due to the wireless interferences, accurate path bandwidth estimation is widely known to be difficult, if not impossible. A number of protocols have been proposed in the literature to estimate path bandwidth. A QoS bandwidth routing scheme was proposed in [25] which contained both link and path bandwidth calculation and reservation for mobile ad hoc network. From the available bandwidth on each link along the path, and the scheduling of free time slots, it computed a bandwidth constrained shortest path. The same authors in [26] enhanced [25] by not only considering the shortest path to decrease the blocking probability but also improving the performance in mobile environment with the on-demand feature. However, these work only consider the single channel network environment. In this paper, we consider a more challenging multihop multi-radio multi-channel wireless networks. For IEEE 802.11 based multi-channel Wireless Mesh Networks (WMN), [27] presented distributed algorithms to dynamically assign channels and route packets. They used *path capacity*, which was the minimal residual bandwidth of the path, as one of the routing metrics. But they only consider spanning tree graph without delay requirement. As also discussed before, it is difficult to compute the path capacity in wireless networks due to wireless interferences.

This paper will estimate the path bandwidth and delay based on the TDMA MAC layer. TDMA scheduling, which aims to eliminate collision and guarantee fairness, has been studied extensively in the literature [28–32]. Kodialam and Nandagopal [28] studied the effect of interference on the achievable rate region in multihop wireless networks. They treated the interference models as linear constraints and solve the flow problem using linear program. In [29], the same authors considered the problem of jointly routing the flows and scheduling transmissions to achieve a given rate vector using the protocol model of interference. They developed necessary and sufficient conditions for the achievable rate vector. They formulated the problem as a linear programming problem and implemented primal-dual algorithms for solving the problem. The scheduling problem is solved as a graph edge-coloring problem using existing greedy algorithms. In [30], they extended their work to the multi-radio multi-channel wireless mesh networks. Kumar et al. [31] developed analytical performance evaluation models and distributed algorithms for routing and scheduling which incorporate fairness, energy and dilation (path-length) requirements and provide a unified framework for utilizing the network close to its maximum throughput capacity. Alicherry et al. [33] mathematically formulated the joint channel assignment and routing problem in multi-radio mesh networks, and established necessary and sufficient conditions under which interference free link communication schedule can be obtained and designed an simple greedy algorithm to compute such a schedule. All these methods are centralized and did not consider the dynamic arriving and leaving of traffics. In this paper, we will propose a distributed call admission control protocol for the routing, scheduling, and *dynamic* channel binding where the traffics arrive online.

The problem studied here is also closely related to the routing and wavelength assignment (RWA) problem studied in optical networks. A number of results [34–36] have been proposed for RWA problem. The major difference between the problem studied here and the RWA problem is that we have to address the wireless interferences among links while RWA does not have to. Wireless interference makes the problem much harder here. For example, finding a routing path with at least a given capacity and minimum delay can be solved in polynomial time for wired networks (thus optical networks also). Unfortunately, this problem becomes NP-hard in wireless networks.

3 Preliminaries and problem definition

In this section, we will first describe our network model and interference model. Then, we will formally define the call admission control problem we are going to study.

3.1 Network model

There is a set $V = \{v_1, v_2, \ldots, v_n\}$ of n communication routers (also referred as nodes) deployed in a plane. In the system, totally w non-overlapping frequency channels are available, denoted by the set $\mathcal{F} = \{f_1, f_2, \ldots, f_w\}$. Assume that any channel will not cause interference to the communication using different channels. Each node v_i is equipped with $\kappa_i \geq 1$ radios. For simplicity, $\mathcal{R}_{v_i}$ is the set of radios on node v_i, i.e., $\mathcal{R}_{v_i} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_\kappa}\}$. And each node may have different number of radios. Because of local interference, each node should ensure that no more than one radio is active in the same channel simultaneously. Therefore, the number of radios κ_{v_i} that can be used for communication at a node v_i is at most the number of channels w. We use an *undirected* graph $G = (V, E)$ to model the wireless backbone network, where V is the set of nodes and E is the set of undirected communication links. We assume that each node has the same transmission range R_T and uniform interference range R_I. So under the same channel the necessary condition for a terminal v_i to directly communicate with v_j successfully is $\|v_i - v_j\| \leq R_T$, where $\|v_i - v_j\|$ is the Euclidean distance between v_i and v_j. And a receiving node v_i is always interfered by the signal in the same channel from v_j if $\|v_i - v_j\| \leq R_I$ while v_j is not the sender of the communication to v_i. Typically it is assumed that $R_I \simeq 2R_T$. A link (x, y) (with x the sender) is called an interfering link of a link (p, q) (with p the sender) if the receiving node q is within distance R_I of the sending node x. Then link (p, q) is called the interfered link of link (x, y). Obviously, links (x, y) and (p, q) cannot be activated simultaneously if one interferes the other. Table 1 summarizes some symbols used in this paper.

Due to self-interference, one radio can only transmit or receive but not both at any time slot. While one radio at node v_i is transmitting or receiving on one channel, another radio at node v_i can simultaneously transmit/receive on a non-overlapping channel. Each link $e \in E$ has a physical capacity $c(e, f_i) \geq 0$ on the channel f_i, which is the maximum link capacity if no interfering links are sharing the same communication channel. We assume that across the same link, different channels have different capacities. When the channel is unavailable for this link, its capacity is 0. And for a given topology, the channel capacities vary slowly over time, and we use average estimation values to represent the channel capacity of a link.

3.2 Interference model

Given a communication graph $G = (V, E)$, we use $H = (U, \overline{E})$ to denote the interference graph where U is the set of communication links in G and $\overline{E}$ contains an edge (e_1, e_2) if the link $e_1 \in E$ and the link $e_2 \in E$ cannot be activated simultaneously using the same channel. The construction of H depends on the interference model. Notice that a number of other interference models have

Table 1 Symbols frequently used in this paper

V	Set of vertices $v_1, \ldots, v_n$	$\mathcal{R}_v$	The set of radios on node v
$\mathcal{F}$	Set of available channels	E	Set of communication links
f_i	An available channel in $\mathcal{F}$	e	An undirected communication link
$\mathcal{T}$	One scheduling period	κ_{v_i}	Total number of radios on node v_i
$c(e, f_i)$	Capacity of link e at channel f_i		

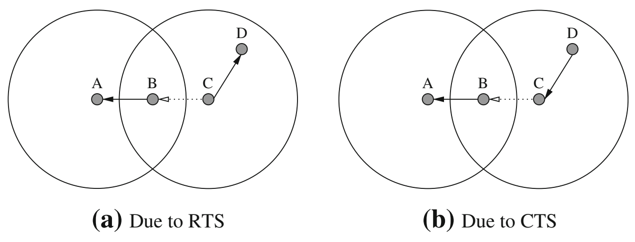


Fig. 2 Communication restriction by RTS/CTS

been studied in the literature, such as the RTS/CTS model, the protocol interference model, k-hop interference model, and the transmitter interference model. Our protocol is interference-model independent, although the performance could vary based on the interference-models.

In the RTS/CTS model proposed in IEEE 802.11 [2], for every pair of transmitter and receiver, all nodes that are within the interference range of either the transmitter or the receiver cannot transmit/receive in the same channel. Figure 2(a) [32] shows the case that communication from B to A and C to D cannot take place simultaneously using the same channel due to RTS. Figure 2(b) shows the case that communication from A to B and D to C cannot take place simultaneously using the same channel due to CTS. Thus, for every pair of simultaneous communication links using the same channel, say (u, v) and (x, y), it should satisfy that u and v are not in the interference ranges of x and y, and vice versa. The *interference region*, denoted by $I_{u,v}$ of a link (u, v) is the union of interference region of nodes u and v. When an undirected link (u, v) is active, all simultaneously transmitting links using the same channel cannot have an end-point inside the area $I_{u,v}$.

In the protocol interference model, a node v cannot receive information correctly from a sender u if there is another sender $w \neq u$ that is actively transmitting simultaneously and v is inside the interference region of node w.

In the k-hop interference model, a node v cannot receive information correctly from a sender u if there is another sender $w \neq u$ that is actively transmitting simultaneously and v is within k-hops of the node w in the communication graph G. As an example, in this paper we will focus on designing the distributed admission control protocol for the 2-hop interference model.

3.3 Problem statement

In this paper, data transmissions are performed by TDMA, which requires that time is slotted and synchronized. The size of a time-slot should be large enough for every link to send at least one packet in a time-slot. As we will see later that, smaller time-slot size implies that the delay for sending data from the source to the destination node will be smaller also. When we have variable sized packets, we assume the time-slot is large enough to support the largest packet for all applications. Thus small packets will be aggregated into a larger packet, or in a time-slot we will send several small packets. For simplicity, in this paper, we assume that all packets have the same size. A link scheduling $\mathcal{S}$ is to assign each link a set of time slots $\subset [1, \mathcal{T}]$ on which it will transmit and the channel it will use to transmit, where $\mathcal{T}$ is the scheduling period. A link scheduling is *interference-aware* (or called *valid*) if a scheduled transmission on a link $(v_i, v_j$ will not result in a collision at either node v_i or node v_j (or any other node). In this context, two types of co-channel collisions must be avoided, namely, primary interference and secondary interference.

In this paper, we assume that all links use the same $\mathcal{T}$. Here $\mathcal{T}$ is at least the chromatic number of the interference graph H such that every link has a chance to transmit at least once every $\mathcal{T}$ time-slots. Clearly, smaller period $\mathcal{T}$ implies a smaller scheduling delay for routing. On the other hand, smaller period $\mathcal{T}$ implies that the network will not be able to support many flows with a small bandwidth requirement. Thus, we assume that initially, the scheduling delay $\mathcal{T}$ is set as the chromatic number of the interference graph H. When there is a new flow that cannot be admitted, we will check whether doubling the scheduling period violates the delay requirement of any existing session. If it did not, we will then double the scheduling period as $\mathcal{T}_{new} = 2\mathcal{T}$ and update the existing schedule of all links by mapping an old slot $t \in [1, \mathcal{T}]$ to two consecutive time slots $2t - 1, 2t$ in the new schedule period $[1, \mathcal{T}_{new}]$. We leave it as a future work to investigate the minimizing-delay maximizing-capacity admission control when different links may have different scheduling periods.

Assume that we are given a network modeled by an undirected graph $G = (V, E)$ with a set of link capacities $c(e, f_i)$ for each channel $f_i \in \mathcal{F}$ and link $e \in E$. Also we assume that each node has stored the reservation information for currently on-going sessions with respect to radios, time-slots and channels. Further assume that we are given a *new* call request with source and destination (s, d) pair, bandwidth requirement $\mathcal{B}$, latency bound $\mathcal{D}$. We need to decide whether this new call request can be admitted or not. The acceptance reply includes a reserved route that guarantees both requirements.

A link channel assignment specifies for each node at a time slot it will transmit using which radio and which channel. Our objective is to give each link $e \in E$ an incrementally updated transmission schedule $\mathcal{S}(e)$, which is the list of time slots it could send packets at certain radio using certain channel such that the schedule is interference-free and the new call can be satisfied without affecting the existing sessions. Let $X_{e,t}, R, f \in \{0, 1\}$ be the indicator variable which is 1 iff e will transmit at time slot t using radio

R and channel f. We will focus on periodic schedules in this paper. A schedule is periodic with period $\mathcal{T}$ if, for every link e and time slot $t, X_{e,t,R,f} = X_{e,t+i\cdot\mathcal{T},R,f}$ for any integer $i \geq 0$. For a link e let $I(e)$ denote the set of links e' that will cause interference if e and e' are scheduled at the same time slot using the same channel. We assume that a link e will be able to collect this information $I(e)$, e.g., $I(e)$ can be collected within a small constant hops of communication links since it is often assumed that the nodes interfered by a transmitting node v_i are 2-hop neighbors of v_i. A schedule $\mathcal{S}$ is *interference-free* if $X_{e,t,R,f} + X_{e',t,R',f} \leq 1$ for any t any f, $e' \in I(e)$, and any R, R'. In the graph theory terminology, the interference free link scheduling problem is essentially the *vertex coloring* of the interference graph [32]. However, besides interference, we consider one more node-radio constraint on scheduling, that is at any time slot a node v_i can use at most κ_{v_i} radios for transmission or reception. Let $Y_{v_i,t,R,f} \in \{0, 1\}$ be the indicator variable which is 1 iff v_i will transmit or receive at time slot t using radio R and channel f. A schedule $\mathcal{S}$ is *available* if $\sum_{f,R} Y_{v_i,t,R,f} \leq \kappa_{v_i}$ for any t, any node v_i.

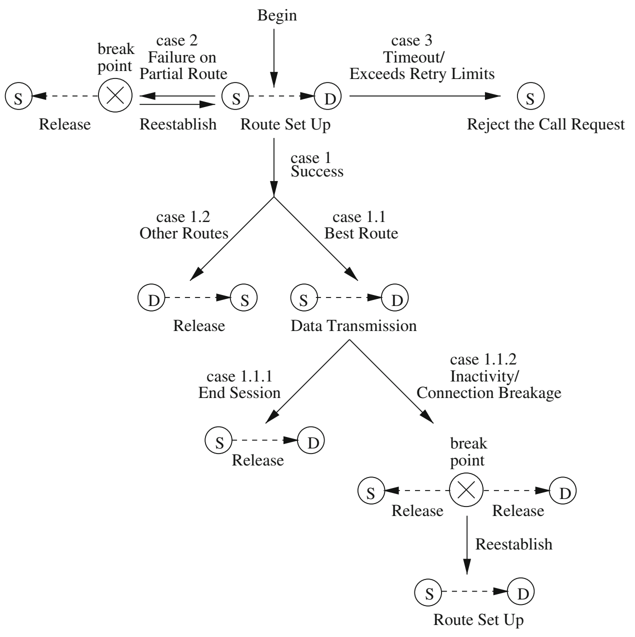


Fig. 3 Overview of the distributed call admission control protocol (DCAC). Here S stands for source node, D stands for destination node

4 Distributed call admission control protocol

In this section, we first introduce the basic data structures for our distributed call admission control protocol (DCAC). Then we give a novel definition for one hop delay. Based on this we define a routing metric to do route discovery and propose an optimal link reservation algorithm. Finally, we explain the four major aspects our protocol to manage QoS sessions.

4.1 Overview of the DCAC protocol

Figure 3 gives an overview of our protocol, where all the cases will be referred in the rest of the paper with detail explanation. Table 2 lists all control packet types. Our DCAC protocol will essentially be composed of the following components:

(1) Reservation update component to announce the reservation of a link to potential conflicting nodes so that all reservations will be conflict-free;
(2) One-hop reservation component in which a link will decide the time-slots and channels that will serve a flow;
(3) The route setup component in which the actual route is connected;
4) Unsuccessful Reservation component deals with unsuccessful reservations;
5) Connection Breakage Route Maintenance component deals with the scenario when some links break down and we need to update the schedule accordingly;
6) Route Release component to update the schedule information when a flow is terminated.

In the rest of the section, we will describe these components in detail.

4.2 Basic data structure

Each radio is assumed to have accurate timing synchronization, using fixed length TDMA channel access. In each repeatable schedule period $\mathcal{T}$, every radio could be assigned multiple time-slots. The time-slot is also of fixed time duration, in which a node can transmit one or several packets. m is the total number of time-slot in one schedule period. The process of scheduling for each hop (v_i, v_j) along a route, just assigns that pair v_i, v_j a set of time-slot numbers from $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ associated with some channel from $\{f_1, f_2, \ldots, f_w\}$. Single channel or multiple channels may be used in one time-slot.

The data structure for recording assignment status with respect to node radios, channels, and time-slots is the *assignment tables*, which consist of the following three tables maintained at each node $v_i \in V$.

- The *neighbor radio active table*,

$$NB(\mathcal{T}, \mathcal{F}) = \{NB(t_i, f_j) \mid \forall t_i \in \mathcal{T}, f_j \in \mathcal{F}\}.$$

It contains the integer variable $NB(t_i, f_i)$ to denote the number of links (x, y) that could interfere with node v_i and link (x, y) has announced its radio will be broadcasting or receiving in time-slot t_i using channel f_i. It also contains a

Table 2 All packet types, functions and formats

Packet type	Function	Format
ROUTE_REQUEST (RREQ)	Send to discover route and do the reservation	Reservation_list, s-d ID, route_list, session #, RLb, TTL
ROUTE_REPLY (RREP)	Send to acknowledge route choice	Reservation_list, s-d ID, route_list, session #
RESERVE_FAIL (RFAI)	NICK for unsuccessful reservation	Reservation_list, s ID, route_list, session #
ROUTE_BROKEN (RBRO)	NICK for route broken	Reservation_list, breakpoint ID, s ID, route_list, session #
CLEAN_RREQ (CREQ)	Clean surplus RREQs	s-d ID, session #
Route_RELEASE (RREL)	Send to release the reserved resources	Reservation_list, s ID, route_list, session #
DATA	Used to transmit datagrams	Data

list, for example, $NB.list(t_i, f_j) = \{R_{q_1}, R_{q_2}, \ldots\}$, indicating which of these neighboring nodes' radios will be active and another list to indicate all nodes that will be active at time t_i using channel f_j. Notice that here $NB(t_i, f_j)$ also include the neighbor node that has reserved time slots and channel for sending or receiving in time-slot t_i using channel f_j for some current new flow request, although this reservation later may be released due to the reservation failure in other links.

- The *self-radio broadcasting table*,

$SB(\mathcal{T}, \mathcal{F}) = \{SB(t_i, f_i) \mid \forall t_i \in \mathcal{T}, f_i \in \mathcal{F}\}$.

It contains the boolean variable, $SB(t_i, f_i) = 1$, when a radio on v_i has announced it is broadcasting in time-slot t_i using channel f_i and a list $SB.list(t_i, f_i) = \{(R_{i_1}, R_{j_2}, sessionID)\}$, indicating which radio it is, which neighbor node's radio it broadcasts to, the ID of the session that this reservation belongs to.

- It also contains the *self-radio receiving table*

$SR(\mathcal{T}, \mathcal{F}) = \{SR(t_i, f_i) | \forall t_i \in \mathcal{T}, f_i \in \mathcal{F}\}$

and all corresponding boolean variables and lists, similar to the self-radio broadcasting table.

4.3 Two hop neighborhood reservation update

The uniform transmission range of each node makes the network graph $G = (V, E)$ a unit-disk graph. Thus network topology can be modeled as every node's communication region is its 1-hop neighborhood and interference region is 2-hop neighborhood. In order to do interference free link reservation, each node should maintain its 2-hop neighbors' assignment status. The distributed algorithm proposed in [37] can be used for each node to compute its 2-hop neighborhoods in an asynchronous environment and update its 2-hop neighbors when nodes join or leave the network. This algorithm uses a total of $O(n)$ messages under the constraint of wireless interferences, where n is the total number of nodes in the network.

Each node maintains a *2-hop neighbor list* to record its two-hop neighbors. In order to be aware of latest 2-hop neighbors assignment status, every time a node makes new reservation, some of its 1-hop neighbors should relay broadcast to all its 2-hop neighbors. In order to minimize the broadcast overhead, each node should determine a minimal subset of its 1-hop neighbors as the relay nodes to cover all its 2-hop neighbors. This problem is called *selecting forwarding neighbors* in [38] and *minimum forwarding set problem* in [39]. For a general communication graph G, this can be modeled as set cover problem and we translate the greedy algorithm in [40] for each node to select its 1-hop relay neighbors. Each node iteratively picks a 1-hop neighbor who can cover the largest number of 2-hop neighbors and removes the covered 2-hop neighbors, until all 2-hop neighbors are covered. This guarantees an approximation factor of $H(\Delta)$, where Δ is the maximum degree and H is the harmonic function. Each node informs its selected relay 1-hop neighbors. Thus for every node it computes a *relay list*, i.e., a subset of its 1-hop neighbors whose updated reservations need to be relayed by this node. Algorithm 1 summarizes our method for updating the

Algorithm 1 Reservation Update to 2-Hop Neighbors

```
Input: Two hop neighbors list $2hopList(v)$ of $v$ and 1-hop neighbors list $1hopList(v)$ of node $v$
Output: relay nodes $R(v)$
1: $S \Leftarrow 2hopList(v); R(v) \Leftarrow \emptyset$
2: while $S \neq \emptyset$ do
3:     Find a node $u$ in $1hopList(v)$ that is the neighbor of the most number of nodes in $S$. Let $c(u)$ be the set of nodes from $S$ that is covered by $u$, i.e., one-hop neighbor of $u$.
4:     $R(v) \Leftarrow R(v) \cup \{u\}; S \Leftarrow S_\backslash(u)$
```

reservation information using a control channel. Here we assume that a node v made a new reservation, and node v wanted to inform all its 2-hop neighbors about this.

Since the maximum node degree in the communication graph G is Δ, it is easy to show that the maximum number of nodes that are within k-hops of a node v in G is at most $O(k^2\Delta)$. The following theorem is straightforward.

Theorem 1 *The number of relay nodes $R(v)$ found by Algorithm 1 is within $O(\log\Delta)$ of the minimum where Δ is the maximum node degree in G.*

When the communication graph G is a unit disk graph (UDG) of nodes deployed in two dimensions, an algorithm with approximation ratio 3 and running time $O(n_2 \log^2 n_2)$ was presented in [38]. Recently this problem was solved exactly in time $O((n_2)^3)$ for network modeled by UDG, where n_2 is the number of two hop neighbors of a node.

4.4 One hop reservation

Our call admission protocol uses the default radio and the default channel to reserve resource on data radios and data channels for QoS session setup. For each communication link, the two endpoint nodes maintain a capacity estimation for each available channel across this link. This will be used by one-hop admission control. Since the energy supply for each router is not constrained, we use hop count to do restricted flooding routing. Every node maintains a routing table recording the shortest path hop counts to other nodes. This can be used to handle the traffic not only between mesh client and gateway node but also between mesh clients themselves.

One hop reservation is the most critical part of our protocol, we will discuss this in Sect. 5 separately.

4.5 Route set up

Assume that a client node wants to initiate a session to another node. The nearby mesh router will serve as the source node. The source node will initiate the ROUTE_REQUEST (RREQ) for a session (refer to Begin in Fig. 3). Hop count is used to do restricted flooding when broadcasting RREQ packets. The source sets its hop count to the destination plus some constant as initial *TTL* (i.e., RREQ initial $TTL =$ source hop count $+ \delta$). This additional δ allows multiple routes to be set up for one RREQ from the source other than only the shortest path. Here $\delta > 0$ is a constant fixed by the protocol which could depend on the network G, e.g., networks with larger network diameter often needs larger δ. Algorithm 2 summarizes our method for nodes to process a route request.

As is to be expected, a destination node will receive more than one RREQ, refer to case 1 in Fig. 3. Every RREQ packet indicates a unique feasible QoS path from the source to the destination. Thus the destination node can choose the best routes among all candidates. In order to reduce the overhead of flooding, the destination node can broadcast a CLEAN_RREQ packet to clean RREQ packets that are still roaming around the network after choosing a satisfactory route.

When the destination node chooses one RREQ packet from the source node (refer to case 1.1 in Fig. 3), it returns a ROUTE_REPLY (RREP) packet by unicasting back to the source following the route recorded in the *route_list* using the reserved resources recorded in *reservation_list*. The destination node copies the fields <*route_list, reservation_list*> from RREQ to RREP.

When the source receives a RREP, the end-to-end bandwidth reservation is successful under QoS requirements, and the session is established. Then the source node can begin transmitting data. It is notable that this establishment protocol for a session connection is a two-way handshaking for the source and destination pair, for each hop it is a three-way handshaking. When a new call request arrives, the call admission control drives this route set up procedure. This new call will not be accepted until the reservation process is successfully completed.

4.6 Unsuccessful reservation

The reservation failure will occur for the partially established route (refer to case 2 in Fig. 3). The node that runs the one hop reservation algorithm and gets a reservation failure will send RESERVE_FAIL (RFAI) packet back to the source following the route recorded in the *route_list* using the reserved resources recorded in *reservation_list*. This node copy the fields <*route_list, reservation_list*> from PREQ to RFAI. As the RFAI traverses back to the source, each node along the path frees the reserved resources in the *reservation_list*, updates its assignment table, and announces this release information to its neighbors in its next *A-slot* or *C-slot*.

On reception of RFAI, the source reestablishes the route. After Route Set Up timeout or reestablish exceeds a time limit, the source rejects the call (refer to case 3 in Fig. 3). In order to increase the acceptance probability, when re-discovering the route, the source may relax the bandwidth or delay requirement or change the destination to other root routers with less traffic demands.

4.7 Connection breakage route maintenance

During the active period of a connection, a link breakage may destroy a session (refer to case 1.1.2 in Fig. 3).

Algorithm 2 Route Request Processing

1: The source node s broadcasts a control packet to all its one-hop neighbors RREQ, including the following information reservation_list, source-destination ID s and d, currently known route_list, the session number, remaining latency bound (RLb) which is $\mathcal{D}$ originally, the requested capacity $\mathcal{B}$ for this session, TTL of the packet RREQ. Here TTL is $\delta + H_G(s, d)$, where $H_G(u, v)$ is the hop-distance between two nodes u and v in the communication graph G.

2: When any non-destination node v receives a RREQ, it will retrieve the TTL from the packet and perform the following operations:

1) It checks its hop count distance to the destination d. If $H_G(v, d) > TTL$, node v discards this request. This ensures that RREQ packet will be forwarded in the right direction towards destination.

2) Otherwise it runs Algorithm 3 to do one hop reservation.

3) If one hop reservation by Algorithm 3 is successful, node v updates $TTL \Leftarrow TTL - 1$, updates the remaining latency bound RLb based on the result of Algorithm 3, and appends v to the reservation list. Node v then update its reservation to its two-hop neighbors $2hopList(v)$ using the protocol described in Algorithm 1. Notice that a 2-hop neighbor u of v will update $NB(t_i, f_j) \leftarrow NB(t_i, f_j) + 1$, and mark this reservation as the reserved time-slot and channel pair by v, but not the final schedule of node v.

Node v broadcasts the updated RREQ to all its one-hop neighbors in the communication graph.

3: If the destination node d received packet RREQ, we know that there is a path from s to d that satisfies the delay requirement and also the capacity requirement. If multiple paths exist, the destination node chooses the path with the minimum delay and send a route_reply packet RREP to the source node.

This breakage possibly results from the environmental changes damage the signal-to-noise ratio and gets a high packet-error rate on the wireless link. The breakpoint is responsible for rerouting the session over a new path. That is once the next hop becomes unreachable or the link quality decreases below the required threshold, the breakpoint sends a ROUTE_BROKEN (RBRO) packet back to the source to blocking the data of this session. It will also prepare a RREQ packet for re-routing. Each node receives this new RREQ will perform *one hop reservation algorithm* for re-routing this session. The source will wait for a new RREP for this session until a timeout occurs, the source stops this session and notifies all the remaining relay nodes to release the reservations for this session.

The route maintenance also requires each node monitors the activity of the time-slots and radio-channel pairs it reserved. If the inactivity period exceeds a threshold, the reserved resources will be purged from the node assignment tables. And a RBRO packet will be sent to the source. Also the breakpoint will prepare new RREQ to re-route to the destination for this session. The progress packet which serve as stay-to-alive purpose for the partially established route is needed to avoid the partial route be purged due to inactivity before it is completed establishment.

4.8 Route release

The last data packet of the session from the source will piggyback a ROUTE_RELEASE (RREL) packet to inform all the nodes along the path to free the resources belong to this session (refer to case 1.1.1 in Fig. 3). Another case for route release is when destination receives multiple RREQ for the same session, it choose the best path and sends RREL packets for the other paths to inform the relay nodes release the reserved resources (refer to case 1.2 in Fig. 3). Or the destination has received enough RREQ packets for choosing and sent CREQ to stop the roaming RREQ. The nodes who receive the CREQ should stop and RREQ and release the partial reserved path back to source.

Route release is also used in the case link breakage and reroute for the on-going session, the destination should broadcast RREL along the old path to inform the relay nodes until the breakpoint.

5 One hop reservation method

The admission control for one call request is to reserve a route satisfying the given delay and bandwidth requirements. In this section, we discuss in detail our method for a link to reserve time-slots and channels for a flow, and compute the delay incurred by this link and the capacity achievable for this link.

5.1 Time-slots for control packets

Our protocol uses a three-way handshaking between the two end-nodes to do one hop reservation. This requires 3 consecutive time-slots, namely the *request*, the *announce*, and the *confirm* time-slots, and each has a specific role to perform. The 3 time-slots must be consecutive in the control channel to assure that only one pair in the interference region is making the reservation; otherwise conflicting reservations may be made. In case when conflict reservations are made due to asynchronous neighbor assignment information update, our protocol uses a random arbitration. That is to associate each reservation packet with a randomly generated number. When one node detects conflict reservations, it always preserves the one with larger generated random number, and ignores the others.

The node, who made unsuccessful reservation, will always be aware of this, and it will undo such reservation following the reservation failure procedure. Observe that our distributed reservation algorithm can be run at all nodes, who do not interfere with each other, in parallel. We then discuss these three slots in detail as follows.

(1) *Request*: In the *request time-slot* (called R-slot for short), if a node v_i desires to establish a forward link to node v_j,[1] it sends to node v_j the following information: (1) node v_i's *blocked table*: $BL(\mathcal{T}, \mathcal{F}) = \{NB(t_i, f_i) \cup SR(t_i, f_i) \cup SB(t_i, f_i) \mid \forall t_i \in \mathcal{T},\ \ f_i \in \mathcal{F}\}$. In other words, all (time-slot, channel) pairs which node v_i cannot operate on. (2) the ID of the destination node d; (3) the session ID; (4) current *route_list*; (5) the bandwidth requirement $\mathcal{B}$; (6) the remaining latency bound RLb; (7) current *self radio broadcasting table* and *self radio receiving table*; (8) set TTL (Time-To-Live) as its own hop distance toward destination; (9) *reservation_list* contains the path of the links from the source node to the current node that already reserved time-slots and channels for this new flow.

The *route_list* records the routing information of the path, i.e., the relay nodes' ID, along which the request packet transverses. The remaining latency bound RLb is the allowable time left for the remaining path: initially it is the $\mathcal{D}$ for the entire route set up; for every hop where the reservation is made successfully, the node should update RLb (by deducting the delay that will be incurred on this link) when forwarding it to the next hop. The *reservation_list* records the reserved information on each hop along the route. It consists of 4-tuples like $(t_i, f_j, R_{v_q}, R_{u_r})$, t_i represents which time-slot is reserved, f_j represents which channel this time-slot will use, R_{v_q}, R_{u_r} represent the transmitting and receiving radios which will use this reserved time-slot with this channel, respectively. Each hop may reserve multiple time-slots, and each time-slot may reserve multiple channels for multiple radios transmitting.

(2) *Announce:* In the *announce time-slot* (called *A-slot* for short), node v_j broadcasts the following: (1) the updated *reservation_list*; (2) the destination ID; (3) append its ID to *route_list*; (4) the session ID; (5) itself reservation release information; (6) the updated remaining latency bound RLb; (7) the updated TTL as v_j's hop count towards destination; (8) synchronization message; (9) the updated reservation information of the 1-hop neighbors in its *relay list*. Each neighbor of node v_j will receive the *A-slot* packet and will enter this information into its assignment tables.

(3) *Confirm:* In the *confirm time-slot* (called *C-slot* for short), node v_i copies the *A-slot* information to the *C-slot* packet except the change of the item 9) according to its own *relay list* and broadcasts. Each neighbor of node v_i will receive the *C-slot* packet and update its assignment tables. Additionally, the *C-slot* packet confirms the reservation and appoints node v_j to establish the next hop reservation for this session.

[1] We just consider one direction traffic reservation here. It is easy to extend this to the bidirectional reservation. Also only 3 time-slots are required for the two end-nodes of each link to do both forward and backward traffic reservations.

Table 3 Channel assignment

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
R_{i_1}	f_1		f_1		f_1		
R_{i_2}	f_2	f_2					
R_{j_1}	f_1		f_1		f_1		
R_{j_2}	f_2	f_2					
R_{j_3}							

5.2 Reservation procedure

We then discuss in detail how a node reserves the time-slots and channels for a flow. On reception of a R-slot packet, the receiving node v_j first computes its *blocked table* $BL(\mathcal{T}, \mathcal{F})$ and finds the possible available pairs of time-slot and channel for link $e = (v_i, v_j)$. This can be computed by performing logical *OR* of the *blocked tables* of two endpoints as $BLv_i, v_j(\mathcal{T}, \mathcal{F}) = \{BLv_i(t_i, f_i) \bigvee BLv_j(t_i, f_i) \mid \forall t_i \in \mathcal{T}, f_i \in \mathcal{F}\}$. The boolean variable $BLv_i, v_j(t_i, f_i) = 0$ indicates in time-slot t_i channel f_i is available to be assigned.

Node v_j sorts the available channels for each time-slot on the descending order of their link capacities. For example, the set of available time-slot, channel pairs can be (t_1,f_1, f_2,f_4), (t_3,f_1,f_3), (t_4,f_1,f_2,f_3,f_4). This means, time-slots t_1, t_3, t_4 will be available and for time-slot t_1 the available channel is f_1,f_2,f_4 with $c(e,f_1) \geq c(e,f_2) \geq c(e,f_4)$. Node v_j also builds a two-dimension assignment status table (see Table 3 for illustration) with regard to radio, channel and time-slot for v_i and v_j. This table uses time-slot as horizontal label and radio as vertical label. Each table entry is filled with a channel or left blank. Here when we assign channel f_1 to the radio R_{i_1} of node v_i at time-slot t_j, we put f_1 to the entry (R_{i_1}, t_j) of the table. When no channel is assigned, the entry (R_{i_1}, t_j) is left empty. Nodes v_j uses this information to run reservation algorithm for this hop.

Scheduling delay: *Scheduling delay* on node v_i counts from the last packet it received for this session until the last packet it sent out. For one hop scheduling delay, we count the delay at sending node except the destination, since the destination does not have such delay from our definition. In other words, this is the buffering interval at v_i for this session during one schedule period. This definition is from the perspective of one session, instead of the perspective of one packet. This cannot be computed simply by subtracting the scheduled last receiving time-slot (*lastIn*) from the scheduled last sending-out time-slot (*lastOut'*). We should first perform a mapping between the scheduled receiving time-slots and the scheduled sending-out time-slots, then count the time elapse from the lastIn slot to the mapped *lastOut*.

After the mapping, this hop scheduling delay is lastOut−lastIn, if lastOut slot is within the same period as lastIn slot, otherwise is $lastOut - lastIn + m$. For example, consider a scheduling along a path: the previous hop scheduled at time-slots t_1, t_4, t_5 and current hop scheduled at time-slots t_2, t_3, t_6. Although $6 > 5$, that is last sending-out time-slot is after the last receiving time-slot, but the scheduling delay is not $|t_6 - t_1| = 5$. Due to mc-mr configuration, each time-slot may have multiple channels scheduled in one hop, we need consider the available channel list for each available time-slot when doing the mapping. Let us consider a simple example. Here we just consider one uniform capacity channel for each available time-slot. Later in our algorithm, we will present the actual mapping algorithm in details. We do a mapping first, $t_1 \rightarrow t_2, t_4 \rightarrow t_6, t_5 \rightarrow t_3$. This means at time-slot t_2 current node will relay the data packet that was received at t_1 of the same scheduling period, at time-slot t_6 it will relay data received at t_4, and at time-slot t_3 (of next scheduling period) it will relay the data received at t_5. Here $t_a \rightarrow t_b$ means the data scheduled to be received at slot t_a is transmitted using the scheduled slot t_b (of next scheduling period if $b \leq a$) at node v_i. Refer to our previous example, the lastOut is t_3 of next period, and lastIn is t_5, so the scheduling delay is $3 - 5 + m$. However, the tricky point is we can improve the mapping above to shorten the scheduling delay, as follows, $t_1 \rightarrow t_3, t_4 \rightarrow t_6, t_5 \rightarrow t_2$. Such that the new scheduling delay is $2 - 5 + m$. Then one of the challenging questions is to find a scheduling at all links such that the overall delay of a route is minimized.

Switching delay: In addition to the scheduling delay, there is another delay called switching delay. *Switching delay* is the channel switching overhead incurred at each hop (v_i, v_j) for the session's packet transmissions in one schedule period. It should consider both endpoint radios for this link. Referring to the assignment table for the two endpoint nodes, we count the channel switching overhead for the channel assignment at this hop. For example, Table 3 records the current assignment status on nodes v_i, v_j when we want to reserve slot t_2, t_6, t_7 for this link (v_i, v_j). If we assign f_1 to R_{i_1} and R_{j_1}, no channel switching, switching delay is 0. If we assign f_3 to R_{i_1} and R_{j_1}, both radios incur 3 channel switches (at time-slots t_2, t_3 and t_6), the switching delay is $(3 + 3) \times$ switchOverhead. If we assign f_3 to R_{i_1} and R_{j_3}, R_{i_1} incur 3 switches and R_{j_3} does not, so switching delay becomes $3 \times$ switchOverhead.

Our reservation algorithm is to satisfy bandwidth requirement and minimize one-hop delay. One-hop delay consists of scheduling delay, channel switching delay, and the transmission delay incurred at this hop. Given packet size, the transmission delay is fixed. Our aim to minimize the first two delay components. To amortize the overhead of channel switching (i.e., 80 µs), the time-slot will be set to relatively large, for example, SSCH chooses 10 ms, so that the scheduling delay is dominant. Therefore, we first schedule time-slots to minimize the scheduling delay, then assign channels to radios to minimize the switching delay.

5.3 Reservation algorithm

To successfully achieve our reservation objective on link (v_i, v_j), node v_j needs to find a valid time-slot schedule for v_i sending-out and v_j receiving-in, with an objective to minimize the delay between the time-slots scheduled for v_i receiving-in at previous link. Our reservation method divides the scheduling into two complementary cases, namely *uniform channel capacity* and *heterogeneous channel capacity*, which will be studied separately.

(1) *Uniform channel capacity:* In this case, for any link, all channels have the same bandwidth capacity. While the current hop will still possibly use different number of time-slots from previous hop, because in different time-slots the number of radio links distributed among different channels may be different. Assume that we are considering a node v_j, which has received a one-hop scheduling request from node v_i. Assume that the time-slots reserved at previous hop (v_i, v_j) are $\{t_1, t_2, \ldots, t_k\}$, and we also know the corresponding channels to be used at each time-slot t_a by link (v_i, v_j). Node v_j will consider all its one-hop neighbors to find a relay node. Assume that node v_j is considering a neighbor v_p, and the set of available time-slots in this hop (v_j, v_p) is $\{t'_1, t'_2, \ldots, t'_d\}$. Here generally $k \neq d$. Notice that here a slot t is said to be available for link (v_j, v_p) if there is a channel f such that node v_j can send data to node v_p using channel f at time-slot t in the scheduling period $\mathcal{T}$ without causing any interferences.

Due to the mc-mr configuration, for each *available* time-slot we should consider the slot output capacity $B(t'_i)$, and for each *scheduled* time-slot we should consider slot input load $B(t_i)$. $B(t'_i) = x'_i * c$ and $B(t_i) = x_i * c$, where c is the uniform

Algorithm 3 Multi-radio Uniform Channel Capacity Scheduling

```
Given Input: $T = \{(t_1, x_1)(t_2, x_2)\cdots(t_k, x_k)\}, T' = \{(t'_1, x'_1)(t'_2, x'_2)\cdots(t'_d, x'_d)\}, S = \{\emptyset\}, h = 0$.
1: if $(\sum_{i=1}^{k} x'_i < \sum_{i=1}^{k} x_i)$ then
2:     Failure and exit because the total residual capacity of this link is less than that required.
3: else
4:   for each $t_i, i \leftarrow 1\cdots k$ do
5:     while $(x_i > 0)$ do
6:       if there exists $t'_j > t_i$ then
7:       $j = \arg\min\{t'_j \mid t'_j > t_i\}$
8:       if $x'_j > x_i$ then
9:         $S = S \cup \{(t'_j, x_i)\}, x'_j = x'_j - x_i, x_i = 0$
10:        else
11:          $S = S \cup \{(t'_j, x'_j)\}, x_i = x_i - x'_j$, remove $(t'_j, x'_j)$ from $T'_1$
12:        else
13:        $j = \arg\min\{t'_j\}$
14:        if $x'_j > x_i$ then
15:          $S = S \cup \{(t'_j, x_i)\}, h = h + x_i, x'_j = x'_j - x_i, x_i = 0$
16:        else
17:          $S = S \cup \{(t'_j, x'_j)\}, h = h + x'_j, x_i = x_i - x'_j$, remove $(t'_j, x'_j)$ from $T'_1$
18:      Sort the pairs $(t''_i, x''_i)$ in $S$ in ascending order by $t''_i$.
19:      $LastOut = t''_q, q = \min\{j \mid \sum_{i=1}^{j} x''_i \geq h\}$
20:     if $t_k < t''_q$ then
21:      $Sche\_delay = t''_q - t_k$
22:     else
23:      $Sche\_delay = t''_q + m - t_k$
```

channel capacity, x'_i is the available number of channel radio pairs in t'_i at the current hop, x_i is the number of used channels in t_i of previous hop. Here x'_i is the minimum of the number of available channels, the number of available radios on transmitting node, and the number of available radios on receiving node in t'_i. Since the capacity is uniform, we associate each time-slot with its x_i or x'_i to denote its slot load or capacity. We require $\{t_1, t_2, \ldots, t_k\}$ sorted in increasing order of time, (i.e., t_1 may not be the first slot in a period, it must be the first reserved slot in previous hop (v_i, v_j) for v_i to receive the packet). So t_k is *lastIn*, we need find a perfect mapping, which has minimal scheduling delay, that is (*lastOut* − *lastIn*) if lastOut slot and *lastIn* slot are within the same period, otherwise (*lastOut* − *lastIn* + *m*).

We use a greedy method to schedule input load of each slot at previous link in order, with the earliest output capacity at current link. If there exists output slot later than the input slot, we choose the earliest output slot. During the scheduling, we count the number of unit channel load that was scheduled to the next period output capacity, say h. In the last step of mapping, we always map the first h unit output capacity to transmit the previous link's last input load to minimize the scheduling delay. So *lastOut* is the time-slot in the next period which can earliest send out the delayed h input load from the previous hop. The detailed algorithm is given in Algorithm 3.

Lemma 2 Algorithm 3 *provides the correct one-hop time-slot schedule with minimized scheduling delay.*

Proof Given the previous hop schedule, *lastIn* is fixed, so the scheduling delay is determined by *lastOut*. Our proof divides into two cases.

Case 1: *lastOut* is within the same period as *lastIn*. In this case, every packet received and sent in the same period. Our algorithm delivers the packets in First In First Out (FIFO) order, and in the earliest available output time-slot. If the optimal ζ is different from our solution, ζ must has at least a swapped pair, that is a pair of packets, P_a and P_b, P_a was received earlier than P_b but sent out later. We can switch output order of P_a and P_b, since arriving time P_a is smaller than P_b, the earliest available time-slot for P_b must be available for P_a. So switching output order of P_a and P_b and the total scheduling delay cannot be longer. For the same reason after switching all the swapped pairs, the scheduling delay cannot be longer than that of ζ, but in this way ζ becomes our solution. So for this case, our solution is optimal.

Case 2: *lastOut* is in the next period of *lastIn*. Our algorithm delivers all the delayed h packets, in the first h available time-slots in the next period. Assume the optimal solution ζ is different from our solution. Notice our solution always sends packets in FIFO order and considers first whether there is available time-slot in the same period, and always chooses the earliest available time-slot. Thus, given the same input schedule from previous hop, ζ cannot have smaller number of delayed packets than ours, which is h. If delay of ζ is shorter than ours, ζ need to complete all the packets including at least h delayed packets before the earliest h available time-slot in next period from this hop available time-slot set. This is conflicting with the available time-slot for ζ is the same for our solution. So the optimal solution ζ cannot be different from our solution.

Although we find a schedule for a link to minimize the scheduling delay, this does not guarantee that the overall delay over the found path is minimized. The challenging question now is to find a path, given the current reservation status of all links, whose bandwidth (the minimum among all links) is at least the required value, and whose delay is minimized among all potential paths. Unfortunately, this question is NP-hard for wireless networks with interference constraints since even computing a path with the maximum capacity is already NP-hard [28–32].

Theorem 3 *Given a wireless networkG with time-slot and channel reservations for all links, it is NP-hard to find whether G has a path connecting a pair of nodess andd with the delay at most $\mathcal{D}$ and the capacity at least $\mathcal{B}$.*

To minimize the delay of the routing, the following method is a naive approach. We first only consider all links of G (say $\mathcal{L}$) whose residual link capacity is at least the required value $\mathcal{B}$. We then compute the scheduling delay for each of these links and assign this scheduling delay as "length" of the link. We find the shortest path from source to destination using links in $\mathcal{L}$. However, it is not difficult to show that the scheduling delay of a minimum-delay path does not necessarily satisfy the optimum sub-path property. In other words, given a path $(v_1, v_2, \ldots, v_i, v_{i+1}, v_k)$ that is delay optimum for the pair of nodes v_1 and v_k, the sub-path $(v_1, v_2, \ldots, v_i)$ is not necessarily the optimum path for minimizing schedule delay. Thus, this naive approach may not always find the path with the optimum schedule delay.

(2) *Heterogeneous channel capacity*: In this case, across the same link, different channels have different bandwidth capacities. For every link $e \in E$, two endpoint nodes maintain a sorted list of channel capacity for this link, $\{c(f_1), c(f_2), \ldots, c(f_w)\}$, in the descending order. Given the time-slots reserved for previous hop say $\{t_1, t_2, \ldots, t_k\}$ and a set of available time-slots in this hop say $\{t'_1, t'_2, \ldots, t'_d\}$, generally $k \neq d$. Due to non-uniform channel capacity, we associate each available time-slot t'_i with a list of candidate channels $L'(i)$, which contains the x'_i maximum capacity channels available in this t'_i. The definition for x'_i is the same as for uniform capacity case. Thus we define the slot output capacity $B(t'_i)$ for each available time-slot t'_i in current link as the sum of capacity for all channels in $L'(i)$ and for each scheduled time-slot t_i the slot input load $B(t_i)$ is the sum of all used channel capacities in that slot.

The scheduling algorithm for different capacity case is similar as uniform capacity case. The major difference is mapping the input load for one scheduled time-slot, e.g., for t_i, we should greedily choose an output time-slot with same requirement, in other words t'_j should schedule integer number of largest capacity channels in its current channel list, (i.e., always from the head of $L'(j)$ and update $B(t_i)$ and $L'(j)$ accordingly). For example, assume the available channels for current node is $L'(j) = \{f_1, f_3, f_4\}$ and $c(f_1) < B(t_i)$ and $c(f_1) + c(f_3) \geq B(t_i)$. We will choose f_1, and f_3, so $B(t_i) = 0$ and $L'(j) = \{f_4\}$. For locating the *lastOut*, we record the total delayed input load, say h. So *lastOut*, say t_q is the one such that q is the minimum of j such that $\sum_{i=1}^{j} B'(t'_i) \geq h$.

5.4 Channel assignment

After determining scheduling delay, we need assign channels to the radios, such that the total channel switching delay is minimized. Given a list of partial reservation for this hop:

$\{(t_1, L_1, R_1), (t_2, L_2, R_2), \ldots, (t_b, L_b, R_b)\}$

$t_1, \ldots, t_b$ is the reserved time-slots in time increasing order. L_i is the set of reserved channels in t_i, R_i is the set of available radios for this link in t_i. We need to assign all the channels in L_i to some pair of radios in R_i, such that all the reserved channels in t_i can be used by some pair of radios at this link. R_i consists of available radios on both endpoint nodes, and we assign radios on the nodes sequentially.

By considering each (t_i, L_i, R_i) tuple in order, the problem of assigning all channels in L_i to some pair of radios in R_i with minimal total switching delay can be converted to finding a perfect matching with minimal total weight in a bipartite graph from L_i to R_i. This can be solved by Kuhn-Munkras-Algorithm [41]. Therefore, *switching delay = minimal total weight × unit switching overhead.* We construct the weighted bipartite graph as follows. For each r in R_i, check current assignment table in order of t_{i-1}, t_{i+1} then t_{i-2}, t_{i+2} until there is assignment made for

r or until the end of the table, that is t_1, t_m. If the latest assigned channel before t_i to r, say f_1 which is in L_i, associate r with f_1 with weight 0. If the earliest assigned channel after t_i to r, say f_2 which is in L_i, associate r with f_2 with weight 0. For other channels in L_i, if it, say f_3, differs from f_1 and f_2 but has some blank space from f_2, associate r with f_3 with weight 1. If f_3 is just in between with f_1 and f_2 without blank space,associate r with f_3 with weight 2. If r has not been assigned to any channel, associate r with every channel in L_i with weight 0. In such a way, we match each $r \in R_i$ with each $f \in L_i$ and associate each match a weight $\in \{0, 1, 2\}$, the channel radio assignment for a node is to find a minimal weighted matching in bipartite graph from L_i to R_i. For a link, we find two matches separately for the transmitter and the receiver in the same way.

For example, referring to Table 3, the partial reservation for the hop (i, j) is (t_4, L_4, R_4), where $L_4 = \{f_1, f_2\}$ and $R_4 = \{R_{i_1}, R_{i_2}, R_{j_1}, R_{j_2}, R_{j_3}\}$. We firstly construct two bipartite weighted graph for $\{f_1, f_2\}$ to $\{R_{i_1}, R_{i_2}\}$ and $\{f_1, f_2\}$ to $\{R_{j_1}, R_{j_2}, R_{j_3}\}$, respectively. The resulted graphs are presented as follows: $\{f_1 \xrightarrow{0} R_{i_1}, f_2 \xrightarrow{0} R_{i_1}, f_1 \xrightarrow{1} R_{i_2}, f_2 \xrightarrow{0} R_{i_2}\}$ and $\{f_1 \xrightarrow{0} R_{j_1}, f_2 \xrightarrow{0} R_{j_1}, f_1 \xrightarrow{1} R_{j_2}, f_2 \xrightarrow{0} R_{j_2}, f_1 \xrightarrow{0} R_{j_3}, f_2 \xrightarrow{0} R_{j_3}\}$. Then we do the channel assignment, i.e., $\{f_1 \rightarrow (R_{i_1}, R_{j_1}), f_2 \rightarrow (R_{i_2}, R_{j_2})\}$, whose switching delay is $0 \times \textit{unit switching overhead} = 0$.

6 Performance evaluation

The performance of our proposed protocol was extensively evaluated by simulation in Qualnet [42]. Notice that our proposed protocol was essentially a cross-layer protocol for multi-channel and multi-radio wireless mesh networks, which integrates channel assignment and scheduling into routing and performs call admission control for each session. We firstly established a multi-channel and multi-radio (MCMR) physical layer model, and then we implemented the distributed call admission control (DCAC) protocol based on the MCMR model.

For simplicity, we implemented the TDMA scheme on the basis of the existing 802.11 MAC protocol. Some modifications to implement the TDMA scheme are described briefly as follows. When a packet is routed to the next hop and delivered to the MAC layer, it is altered to carry some necessary information, such as when to send it (i.e., the scheduled time slot) and via which channel to send it (i.e., the scheduled channel). When receiving the packet, the MAC protocol holds the packet until the scheduled time slot comes. In this way, the contention based MAC protocol was adapted to a TDMA based protocol. The length of the slot is set to be sufficient to send a packet of maximum size, which is 2,346 bytes. Since the time slot was scheduled based on the reservation status disseminated by the neighbor nodes during the period of route setup, it can guarantee the avoidance of collision when accessing the channel. As for the default channel, we adopted the traditional CSMA/CA scheme to ensure reliable transmission of control packets among nodes in the network. Transmitting and relaying the RREQ packet was used for delivering reservation information and establishing path between the source and the destination, while reversing the RREP packet was used for making reservation in the path selected.

Seeing that both DCAC and AODV are kind of on-demand routing protocols, they have similar procedures of route setup and route maintenance in general. So we take AODV as the object of comparison. Notice that the traditional AODV protocol, typically used in Ad hoc networks, is not able to exploit multi-channels and multi-radios in wireless mesh networks. For fairness, we did some modifications to make AODV support multi channel and multi radio, which is referred to as M-AODV in the following parts of the paper. Here, we give a brief introduction of M-AODV.

As in DCAC, M-AODV needs a default channel to exchange control packets. When a source node launches a new session, it uses an idle radio and a free channel to start the procedure of route setup. Only the neighbor nodes, which have idle radios and can use the same channel as the source without interfering neighbors in two hops, will make channel reservation and forward the request packet. As a result, each session will establish a path consisting of links with an identical channel. Compared with DCAC, M-AODV removes the scheduling delay and switching delay, but it cannot manage QoS sessions and the number of admitted sessions is directly limited by the minimum of the number of radios and the number of channels.

Each node is equipped with 3 radios, and there are 3 channels available in the scenario, whose frequency are 2.412, 2.437 and 2.462 GHz, respectively. These channels are free of adjacent-channel interference when they are used simultaneously. All these channels have the same capacity, which is 2 Mbps. In our protocol, the channel with the frequency of 2.412 GHz is defined as the default channel to perform reservation. The other two channels are dedicated to deliver QoS traffics.

Following two metrics are used to evaluate the performance guaranteed by the protocols.

- *Average end to end delay per session*, which is the average time taken for a packet in a session to be transmitted from source to destination.
- *Average throughput per session*, which is the average number of bits per second received by the destination of

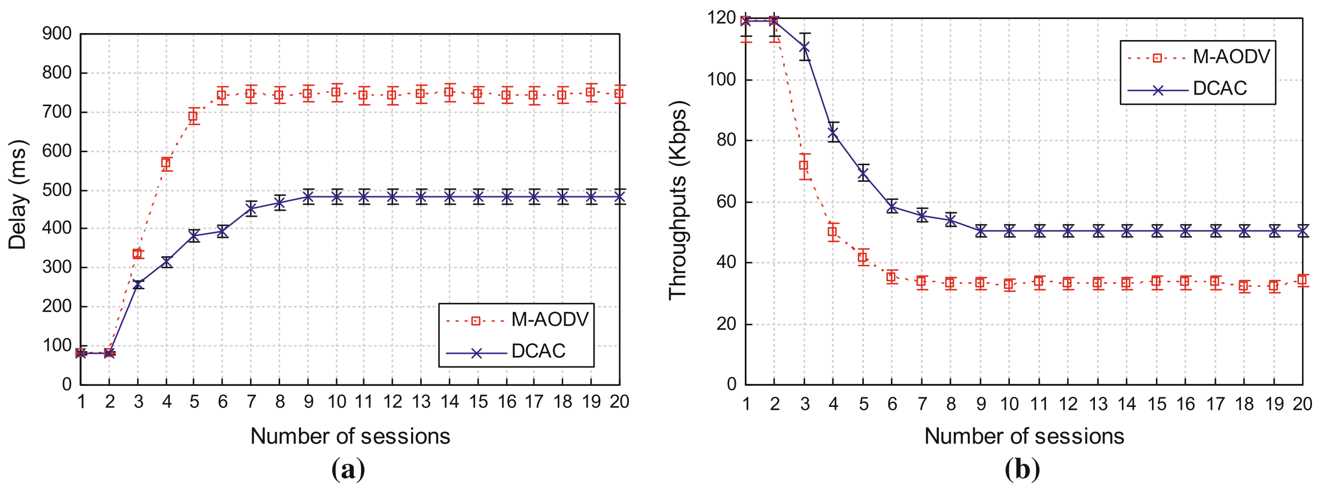


Fig. 4 Different QoS bounded by M-AODV and DCAC, with different number of sessions poured into the grid network. **a** Average end-to-end delay of each session. **b** Average throughput of each session

a session. This metric can be viewed as the bandwidth gained by a session.

6.1 Grid networks

We evaluate the performance of the protocols with 49 nodes put in a grid of 7 by 7, in a square area of 1,500 m by 1,500 m. We set the transmission range of each node to 250 m (by setting the transmission power of each node to 8.2 dBm in Qualnet). The sources and destinations of sessions are located in the left-bottom corner and right-upper corner of the square, respectively. So the expected distance between the source node and the destination node is about 12 hops. For both M-AODV and DCAC, an increment to TTL, that is δ in Algorithm 2, is set to 2.

Incremental number of sessions, that is from 1 to 20, are poured into the network. Each node launches no more than three sessions. We set the delay requirement of each session to 500 ms, and the bandwidth requirement to 50 Kbps. Different QoS bounded by M-AODV and DCAC are showed in Fig. 4. The results shown in these figures are averages of 5 runs with 95% confidence intervals.

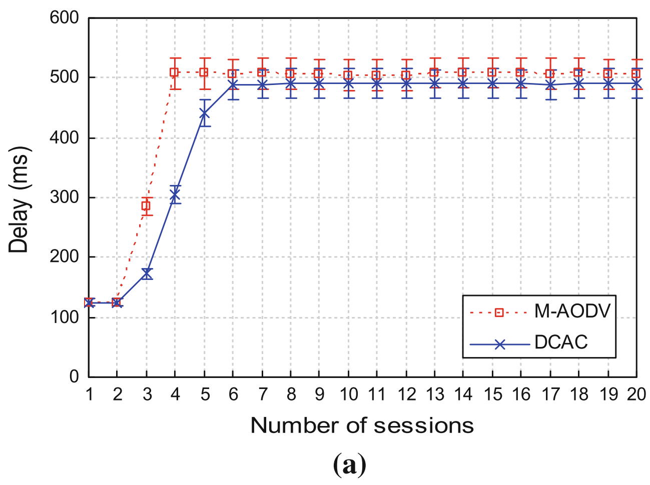
From the above figures, we can see that when the number of sessions is up to two, there is no difference between M-AODV and DCAC, since they can both exploit the two channels to reserve two separate pathes. As more sessions are poured into the network, the delay bound by M-AODV rises faster than that bounded by DCAC. At the point when four sessions are poured into the network, the delay bound by M-AODV is larger than 500 ms, which has exceeded the delay requirement of sessions. However, more sessions will still be admitted into the network, until the setup request is rejected. In this scenario, at most six sessions can be admitted to the network for M-AODV. On the contrary, DCAC guarantees the QoS requirement for all the admitted sessions. Besides that, DCAC can make better use of the multi-channels based on TDMA MAC, hence more session are admitted. In this scenario, nine sessions are admitted to the network for DCAC. Thus, our DCAC protocol also admits more sessions than the M-AODV protocol.

So we can conclude that M-AODV admits sessions without taking their QoS requirement into account, while DCAC guarantees the QoS requirement for all the admitted sessions. Even with this constraint, DCAC can admit more sessions than M-AODV, due to the advantage of DCAC over M-AODV in the aspect of link utilization.

6.2 Random networks

Next, we study the performance of our proposed protocol in a random network. We put 50 nodes uniformly in a square area which is 1,500 m by 1,500 m. The transmission range of each node is about 250 m, and the interference range is about 500 m. As in the grid network, we specify these nodes located in the left-bottom corner and right-upper corner as the sources and destinations of sessions, respectively. The QoS requirement for each session and other parameters of simulation are the same in the grid networks.

Figure 5 shows the change of average delay and average throughput for each session, when increasing number of sessions are poured into the network. The trend is the same as what is observed in the grid network. In more detailed words, when there is only one or two session exists, the QoS bounded by M-AODV and DCAC are the same. As more sessions are poured into the network, M-AODV admits sessions without taking their QoS requirement into



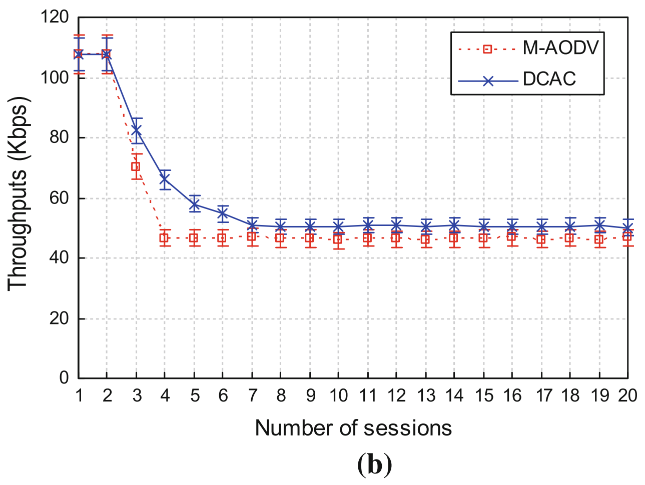


Fig. 5 Different QoS bounded by M-AODV and DCAC, with different number of sessions poured into the random network. **a** Average end-to-end delay of each session. **b** Average throughput of each session

account, while DCAC guarantees the QoS requirement for all the admitted sessions. In the random network, M-AODV can admit at most four sessions, while DCAC can admit at most seven sessions.

From the simulation results shown above, we conclude that our DCAC protocol performs much better than M-AODV protocol when nodes are regularly deployed, and performs at least as good as M-AODV protocol when nodes are randomly deployed. The reason why our protocol performs better in the regular deployment environment is that in this environment the delay of the path actually is closely related to the number of hops in that path. Because our protocol uses a greedy method to minimize the delay of each hop, thus the delay of our protocol is much smaller than M-AODV protocol. Note that M-AODV does not use any heuristic. The better throughput performance of our protocol in the regular deployment can be interpreted in a similar way; because each time we choose the number of relay nodes as small as possible (as in Theorem 1), the potential interference caused by relay nodes will decrease, thus a better throughput. Consider we usually deploy nodes regularly in our daily life, our protocol DCAC actually outperforms the M-AODV protocol in practical.

7 Conclusion

In this paper we show how to perform admission control while providing certain QoS performance guarantees for the traffics such as the required bandwidth and delay of the traffic. We assume a TDMA link scheduling for a multi-radio, multi-channel, and multihop wireless mesh network where the mesh routers are typically static. We present an efficient distributed call admission control method that takes into account the bandwidth demand and the delay demand of the traffic. Our protocol will *not* cause the service interruption to the existing traffics. We also performed extensive simulations to study the performance of our admission control protocol and found that its performance is much better than the worst case analysis.

There are several interesting questions that we did not address here are left for future research. The first question is to bound the delay of the call admission itself. The second question is that although we provided a distributed call admission control that may find a path with required bandwidth and delay requirement, it is interesting to find a path whose maximum bandwidth is within a certain constant factor of the optimal path. Notice that it has been shown in the literature that it is NP-hard to find a maximum bandwidth path in wireless networks due to intra-path and inter-path interferences.

Acknowledgments The research of authors are partially supported by NSF CNS-0832120, NSF CNS-1035894, National 973 Project of China under Grant No.2011CB302701, National Science Fund for Distinguished Young Scholars under Grant No.60925010, National Natural Science Foundation of China under Grant No. 60828003, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), and Tsinghua National Laboratory for Information Science and Technology (TNList).

References

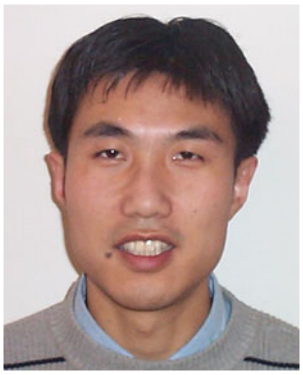
1. Akyildiz, I., Wang, X., & Wang, W. (2005). Wireless mesh networks: A survey. *Computer Networks, 47*(4), 445–487.
2. *IEEE 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, std., 1999.
3. Meshdynamics: Wireless for the outdoor enterprise. http://www.meshdynamics.com/.
4. Bahl, P., Chandra, R., & Dunagan, J. (2004). SSCH: Slotted seeded channel hopping for capacity improvement in IEEE

802.11 ad-hoc wireless networks. In *Proceedings of ACM MobiCom*, pp. 216–230.

5. Lee, S., Narlikar, G., Pál, M., Wilfong, G., & Zhang, L. (2006). Admission control for multihop wireless backhaul networks with QoS support. In *Proceedings of IEEE WCNC*, 92–97.
6. Sriram, S., Reddy, T. B., Manoj, B. S., & Murthy, C. S. R. (2005). On the end-to-end call acceptance and the possibility of deterministic QoS guarantees in ad hoc wireless networks. In *Proceeding of ACM Mobihoc*, pp. 169–180.
7. Tang, J., Xue, G., & Zhang, W. (2005). Interference-aware topology control and QoS routing in multi-channel wireless mesh networks. In *Proceedings of ACM Mobihoc*, pp. 68–77.
8. Parekh, A. K., & Gallager, R. G. (1993). A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transaction on Networking, 1*(3), 344–357.
9. Demers, A., Keshav, S., & Shenker, S. (1989). Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM*, pp. 1–12
10. Yaron, O., & Sidi, M. (1994). Generalized processor sharing networks with exponentially bounded burstiness arrivals. In *Proceedings of IEEE INFOCOM*, 628–634.
11. Zhang, Z. L., Liu, Z., & Towsley, D. (1998). Closed-form deterministic end-to-end performance bounds for the generalized processor sharing scheduling discipline. *Journal of Combinatorial Optimization, 1*(4), 457–481.
12. Zhang, Z. L., Towsley, D., & Kurose, J. (1995). Statistical analysis of the generalized processor sharing scheduling discipline. *IEEE Journal on Selected Areas in Communications, 13*(6), 1071–1080.
13. Kurose, J., Zhang, Z. L., Liu, Z., & Towsley, D. (1997). Call admission control schemes under generalized processor sharing scheduling. *Telecommunication Systems, 7*(1), 125–152.
14. Szabó, R., Barta, P., Nemeth, F., Biro, J., & Perntz, C. (2000). Call admission control in generalized processor sharing schedulers using non-rate proportional weighting of sessions. In *Proceedings of IEEE INFOCOM*, pp. 1243–1252.
15. Nandita, D., Kuri, J., & Jamadagni, H. S. (2001). Optimal call admission control in generalized processor sharing (GPS) schedulers. In *Proceedings of IEEE INFOCOM*, pp. 468–477.
16. Jamin, S., Danzig, P. B., Shenker, S. J., & Zhang, L. (1997). A measurement based admission control algorithm for integrated service packet networks. *IEEE/ACM Transaction on Networking, 5*(1), 56–70.
17. Jamin, S. (1996). A measurement based admission control algorithm for integrated services packet network. Ph.D. dissertation, USC.
18. Abedi, S. (2004). Improved stability of QoS provisioning for 3G systems and beyond: Optimum and automatic strategy selection for packet schedulers. In *Proceedings of IEEE ICC*, pp. 1979–1985.
19. Srinivasan, R., & Baras, J. S. (2004). Understanding the trade-off between multiuser diversity gain and delay—an analytical approach. In *Proceedings of IEEE VTC*, pp. 2543–2547.
20. Wu, D., & Negi, R. (2004). Downlink scheduling in a cellular network for quality-of-service assurance. *IEEE Transaction on Vehicular Technology, 53*(5), 1547–1557.
21. Kanodia, V., Li, C., Sabharwal, A., Sadeghi, B., & Knightly, E. (2001). Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of ACM MobiCom*, pp. 200–209.
22. Narlikar, G., Wilfong, G., & Zhang, L. (2006). Designing multihop wireless backhaul networks with delay guarantees. In *Proceedings of IEEE INFOCOM*, pp. 1–12.
23. Lim, H., Lim, C., & Hou, J. C. (2006). Improving throughput through spatial diversity in wireless mesh networks: a coordinate-based approach. In *Proceeding of ACM Mobicom*, pp. 14–25.
24. Raniwala, A., Gopalan, K., & Chiueh, T. (2004). Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *ACM Mobile Computing and Communication Review, 8*(2), 50–65.
25. Lin, C. R., & Liu, J. S. (1999). Qos routing in ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications, 17*(8), 1426–1438.
26. Lin, C. R. (2001). Admission control in time-slotted multihop mobile networks. *IEEE Journal on Selected Areas in Communications, 19*(10), 1974–1983.
27. Raniwala, A., & Chiueh, T. (2005). Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings of IEEE INFOCOM*, pp. 2223–2234.
28. Kodialam, M., & Nandagopal, T. (2004). The effect of interference on the capacity of multi-hop wireless networks. In *Proceedings of IEEE Symposium on Information Theory*, p. 472.
29. Kodialam, M., & Nandagopal, T. (2003). Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *Proceedings of ACM MobiCom*, pp. 42–54.
30. Kodialam, M., & Nandagopal, T. (2005). Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of ACM MobiCom*, pp. 73–87.
31. Kumar, V. S. A., Marathe, M. V., Parthasarathy, S., & Srinivasan, A. (2005). Algorithmic aspects of capacity in wireless networks. *SIGMETRICS Performance Evaluation Review, 33*(1), 133–144.
32. Wang, W. Z., Wang, Y., Li, X. Y., Song, W. Z., & Frieder, O. (2006). Efficient interference-aware TDMA link scheduling for static wireless networks. In *Proceeding of ACM MobiCom*, pp. 262–273.
33. Alicherry, M., Bhatia, R., & Li, L. (2005). Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of ACM MobiCom*, pp. 58–72.
34. Ramaswami, R., & Sivarajan, K. (1995). Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking, 3*(5), 489–500.
35. Zang, H., Jue, J., & Mukherjee, B. (2000). A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine, 1*(1), 47–60.
36. Lee, T., Lee, K., & Park, S. (2000). Optimal routing and wavelength assignment in WDM ring networks. *IEEE Journal on Selected Areas in Communications, 18*(10), 2146–2154.
37. Calinescu, G. (2003). Computing 2-hop neighborhoods in ad hoc wireless networks. in *Proceedings of AdHoc-Now*, 175–186.
38. Calinescu, G., Mandoiu, I., Wan, P., & Zelikovsky, A. (2004). Selecting forwarding neighbors in wireless ad hoc networks. *ACM/Baltzer Mobile Networks and Applications, 9*(2), 101–112.
39. Baysan, M., Sarac, K., Chandrasekaran, R., & Bereg, S. (2009). A polynomial time solution to minimum forwarding set problem in wireless networks under unit disk coverage model. to appear.
40. Vazirani, V. V. (2003). *Approximation Alogrithm* (2nd ed.). Berlin, Germany: Springer.
41. Gupta, A., & Ying, L. (1999). On algorithms for finding maximum matchings in bipartite graphs. IBM T. J. Watson Research Center, Tech. Rep. 21576(97320).
42. Scalable Network Technologies. http://www.scalable-networks.com.

Author Biographies

XuFei Mao currently is an Assistant Professor at the Computer Science Department at Beijing University of Posts and Telecommunications. He received PhD degree from Computer Science Department at Illinois Institute of Technology in 2010. He received B.S. from Shenyang Univ. of Tech. and M.S. from Northeastern University, China. His research interests include design and analysis of algorithms for wireless networks and the design and implementation of large-scale wireless sensor net- works.



Xiang-Yang Li has been an Associate Professor (since 2006) and Assistant Professor (from 2000 to 2006) of Computer Science at the Illinois Institute of Technology. He is also a visiting professor of Microsoft Research Asia from May, 2007 to August 2008. He is recipient of China NSF Outstanding Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign. He received a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. The research of Dr. Li has been supported by USA NSF, HongKong RGC, and China NSF. His research interests span the wireless networks, computational geometry, and cryptography and network security. Dr. Li served various positions (as chairs and TPC members) at numerous international conferences. He is a co-chair of Algorithm Aspect in Information Management conference 2007, and a co-chair of the ACM FOWANC workshop (Foundation of Wireless Ad-Hoc Networking and Computing) 2008. Dr. Li has served as TPC member in numerous conferences and is an editor of IEEE TPDS, Ad Hoc & Sensor Wireless Networks: An International Journal, and editor of Networks. He recently also co-organized special issues of several journals such as ACM MONET, and IEEE Journal of Selected Area in Communications.

GuoJun Dai is a professor of Computer Science Department, Institute of Computer Application Technology, Hangzhou DianZi University, Hangzhou, P.R.China. He currently is vice-dean of the school of computer science, Hangzhou DianZi University. He received his Ph.D degree from Zhejiang University, in department of Electronic Engineering, in 1998, and MS degree from Zhejiang University, in 1991. His research field is on wireless sensor networks, CFS system and application, embedded systems.