Distributed Call Admission Protocol for Multi-channel Multi-radio Wireless Networks

Yi Hu
Dept. of Comp. Sci.
City U. of Hong Kong
Hong Kong, China
50096068@student.cityu.edu.hk

Xiang-Yang Li
Dept. of Comp. Sci.
Illinois Insti. of Tech.
Chicago, IL 60616, USA
xli@cs.iit.edu

Hai-Ming Chen
Inst. of Comp. Tech.
Chinese Academy of Sci.
BeiJing, China
chenhaiming@ict.ac.cn

Xiao-Hua Jia
Dept. of Comp. Sci.
City U. of Hong Kong
Hong Kong, China
jia@cs.cityu.edu.hk

Abstract - **In this paper we propose a distributed call admission control protocol (DCAC) to provide bandwidth and delay guaranteed Quality of Service (QoS) in multi-hop wireless mesh networks, by exploiting the multi-channel multi-radio (mc-mr) feature. We propose a distributed link scheduling algorithm to give the bandwidth with minimal one hop delay, and a routing metric for route setup. To the best of our knowledge, this is the first distributed protocol that embeds (mc-mr) feature into Time Division Medium Access (TDMA) to do QoS call admission in wireless mesh networks. Extensive simulations show that our protocol significantly improves network performance on supporting QoS flows compared with some widely used protocols.**

I. Introduction

Wireless mesh networks (WMNs) have recently been a solution for providing last-mile Internet access. A WMN consists of mesh routers, which are rarely mobile and do not have power constraints. The mesh routers are usually equipped with multiple wireless interfaces operating in non-overlapping channels. Therefore, a major challenge in design and operation of such networks is to efficiently allocate channels and schedule transmissions to provide better QoS.

A large amount of work has explored QoS in wireless networks. A delay and rate aware scheduling in 802.11-based multihop networks was proposed in [2]. A scheduling and routing framework that provided upper bounds on end-to-end delays in multihop wireless networks was proposed in [9].

Due to wireless interferences, accurate path bandwidth estimation is widely known to be difficult. A number of protocols have been proposed to estimate single channel path bandwidth. [8] proposed a QoS bandwidth routing scheme, which contained both link and path bandwidth calculations for mobile ad hoc networks. [10] used *path capacity* as one of the routing metrics to do dynamic channel assignment and routing for IEEE 802.11 based multi-channel WMN. But they only consider and spanning tree graph without delay requirement.

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 [3]–[6], [13]. However, all these methods are centralized and did not consider the dynamic traffic properties. In this paper, we will propose DCAC for routing, scheduling, and *dynamic* channel binding where the traffics arrive online.

The main contributions of this paper are as follows. 1. **DCAC for Multi-channel Multi-radio Wireless Networks**: To our best knowledge, this is the first distributed TDMA protocol exploiting multi-channel multi-radio for QoS call admission in multi-hop wireless network. Our protocol complies with IEEE 802.11 standard, and does not need any traffic profile. 2. **QoS Link Scheduling Algorithm**: Little work has been done to deterministically guarantee delay in multi-hop wireless networks, except [7], which proposed a centralized algorithm for admission control based on the knowledge of all network flows, and restricted on tree topology. Previous work (*e.g.*, [11], [12]) on QoS had mainly focused on bandwidth guarantee. This paper provides both bandwidth and delay guarantees for QoS flows. We address the intra-flow and inter-flow interferences to ensure the bandwidth, and analyze the scheduling delay, switching overhead to efficiently meet the delay bound.

The rest of the paper is organized as follows. In section II, we introduce the network model, interference model, and formally define the admission control problem. Our protocol is presented in section III. Simulation studies are reported in section IV. We conclude the paper in section V.

II. Preliminaries and Problem Definition

In this section, we first describe the network model and interference model. Then, we formally define the call

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

admission problem we are going to study.

Given an *undirected* network graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes and E is the set of undirected communication links. Each node v_i has $\kappa_{v_i} \geq 1$ half-duplex radios, denoted by the set $\mathcal{R}_{v_i} = \{\mathbf{R}_{i_1}, \mathbf{R}_{i_2}, \cdots, \mathbf{R}_{i_\kappa}\}$. In the system, there are w non-overlapping frequency channels, denoted by the set $\mathcal{F} = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_w\}$. For $\forall e \in E, \forall \mathbf{f}_i \in \mathcal{F}$, there is a link channel capacity $c(e, \mathbf{f}_i) \geq 0$. Self-interference limits the number of radios κ_{v_i} at any node v_i no more than the number of non-overlapping channels w. Because simultaneously transmit/receive on different radios at one node should be on non-overlapping channels.

Each node is assumed to have the same transmission range R_T and same interference range R_I. The interference model adopts the IEEE 802.11 **RTS/CTS** model.

On arrival of a *new* QoS request, specified by the source-destination pair (s, d), bandwidth requirement $\mathcal{B}$, and delay requirement $\mathcal{D}$, our protocol will admit this request only after setting up a route between (s, d) with transmission schedules satisfying $\mathcal{B}$ and $\mathcal{D}$ without affecting existing flows.

A link scheduling is to assign each link an incrementally updated transmission schedule, which is the list of time slots it send/recieve packets at certain radio in certain channel. A link scheduling is *interference-free* if at any time-slot in any channel no more than one interference link is scheduled transmission. We assume that a link e will be able to collect the scheduling information of its interference links by exchanging with its 2-hop neighbors.

III. DCAC

In this section, we first introduce the basic data structures. Then we analyze one hop delay and propose an optimal link scheduling algorithm. Finally, we explain the major management aspects for QoS flows.

A. Basic Data Structure

In our protocol, each node only knows its own communication links and the associated link channel capacities in the network graph. Besides each node has the scheduling knowledge of its on-going QoS flows. Our **DCAC** protocol uses a hybrid channel assignment scheme, where each node fixes one radio to a common channel, and switches other radios among other channels to serve QoS traffics. The common channel not only works for call admissions on QoS flows but also ensures network connectivity and helps re-synchronization in other channels when necessary.

Each radio, except the default one on common channel, is time synchronized and uses fixed frame-length TDMA. In each repeatable schedule period $\mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_m\}$ (m is the total number of time-slots in $\mathcal{T}$), the process of scheduling one hop (v_i, v_j) is to assign v_i and v_j a set of time-slots and specify which pair of radios will do transmission in which channel. Multiple pairs of radios may be activated in non-overlapping channels at one time-slot.

Each node $v_i \in V$ maintains the *assignment tables* to record schedules for existing flows, details are as follows.

- The *neighbor-radio broadcast table*,

  $$NB(\mathcal{T}, \mathcal{F}) = \{NB(\mathbf{t}_i, \mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$$

  contains the boolean variable $NB(\mathbf{t}_i, \mathbf{f}_i) = 1$ if a neighbor node has announced its radio to transmit in $\mathbf{t}_i$ using $\mathbf{f}_i$. A list of such neighbor radios is associated, e.g. $NB.list(\mathbf{t}_i, \mathbf{f}_i) = \{\mathbf{R}_{j_1}, \mathbf{R}_{q_2}, \ldots\}$.
- The *neighbor-radio receiving table*,

  $$NR(\mathcal{T}, \mathcal{F}) = \{NR(\mathbf{t}_i, \mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$$

  similar to the *neighbor-radio broadcast table*.
- The *self-radio broadcast table*,

  $$SB(\mathcal{T}, \mathcal{F}) = \{SB(\mathbf{t}_i, \mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$$

  contains the boolean variable, $SB(\mathbf{t}_i, \mathbf{f}_i) = 1$, if a radio on v_i has announced to transmit in $\mathbf{t}_i$ using $\mathbf{f}_i$. A list $SB.list(\mathbf{t}_i, \mathbf{f}_i) = \{(\mathbf{R}_{i_1}, \mathbf{R}_{j_2}, flowID)\}$, indicates transmitting radio, receiving radio, and which flow it belongs to.
- The *self-radio receiving table*

  $$SR(\mathcal{T}, \mathcal{F}) = \{SR(\mathbf{t}_i, \mathbf{f}_i) | \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$$

  similar to the *self-radio broadcast table*.

Each node v_i computes a *relay list* consisting of parts of its 1-hop neighbors, who are responsible for broadcast v_i's schedule to update all v_i's 2-hop neighbors.

B. One Hop scheduling

In our **DCAC** protocol, the two end-point nodes use three way handshaking to do link scheduling. This requires 3 consecutive time-slots: the *request*, *announce*, and *confirm* slots. These 3 slots must be consecutive to avoid making conflicting link schedules inside an interference region. In case conflictions occur due to asynchronous neighbor information update, our protocol adopts a random arbitration. That is to associate each scheduling a random generated number, if one node detects conflictions, it only preserves the one with largest number, and ignores others. The one who made unsuccessful scheduling, will be aware of this, and undo such scheduling following the failure procedure. Nodes who do not interfere with each other can perform the scheduling algorithm in parallel. We then discuss these 3 slots in detail.

1) Request: In the *request time-slot* (called *R-slot* for short), if v_i desires to establish a forward link scheduling to v_j, [1] it sends to v_j the following information: 1) v_i's

[1] We just consider one direction scheduling here. It is easy to extend to the bidirectional case. Also only 3 time-slots are required for the two end-point nodes to do both forward and return link scheduling.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

blocked table:
$BLv_i(\mathcal{T},\mathcal{F}) = \{NR(\mathbf{t}_i,\mathbf{f}_i) \cup SR(\mathbf{t}_i,\mathbf{f}_i) \cup SB(\mathbf{t}_i,\mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$. *blocked table* records all (time-slot, channel) pairs where this node cannot be scheduled. 2) source-destination ID (s,d); 3) the flow ID; 4) current *route_list*; 5) the bandwidth requirement $\mathcal{B}$; 6) the remaining latency bound *RLb*; 7) *self-radio broadcast table* and *self-radio receiving table*; 8) set TTL(time-to-live) as its hop distance to destination; 9) current *reservation_list*.

The *route_list* records the relay nodes this request has traversed through. The remaining latency bound *RLb* is the maximum time left for the remaining path. Initially it is the $\mathcal{D}$ for the entire route set up; after each successful link scheduling, the node deducts one hop delay from received *RLb* when forwarding to next hop. The *reservation_list* records the scheduled resources at previous hops.

Scheduling Preparation: On receiving a *R-slot* packet, v_j computes its *blocked table* $BLv_j(\mathcal{T},\mathcal{F}) = \{NB(\mathbf{t}_i,\mathbf{f}_i) \cup SR(\mathbf{t}_i,\mathbf{f}_i) \cup SB(\mathbf{t}_i,\mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$ and finds all blocked (time-slot, channel) pairs for link (v_i, v_j) by performing logical *OR* of the *blocked tables* of two endpoints, $BLv_i, v_j(\mathcal{T},\mathcal{F}) = \{BLv_i(\mathbf{t}_i,\mathbf{f}_i) \bigcup BLv_j(\mathbf{t}_i,\mathbf{f}_i) \mid \forall \mathbf{t}_i \in \mathcal{T}, \mathbf{f}_i \in \mathcal{F}\}$. The boolean variable $BLv_i, v_j(\mathbf{t}_i,\mathbf{f}_i) = 0$ indicates the pair $(\mathbf{t}_i,\mathbf{f}_i)$ is available to be scheduled.

Node v_j sorts the available channels in each time-slot on the descending link capacity order. Node v_j also builds an assignment table (e.g. Table I) for v_i and v_j. In this two-dimension table, horizontal label is time-slot and vertical label is radio. Each table entry is filled with a channel or left blank. Here when we reserve channel $\mathbf{f}_1$ to the radio $\mathbf{R}_{i_1}$ of v_i at $\mathbf{t}_j$, we put $\mathbf{f}_1$ to the entry $(\mathbf{R}_{i_1}, \mathbf{t}_j)$. When no channel is assigned, the entry is empty.

Delay Definition: *Scheduling delay* on v_i counts from the last packet it received for the flow until the last packet it sent out. This is the transmission time plus the buffering interval at v_i for this flow during one schedule period. This can not be simply computed as abstracting the last receiving time-slot(lastIn) from the scheduled last sending out time-slot. We should first perform a mapping between the receiving time-slots and the available sending out time-slots, then count the time elapse from the lastIn slot to the mapped last sending out time-slot(lastOut).

For one hop delay, we count scheduling delay at the sender except the source. After mapping, the scheduling delay at this hop is $lastOut - lastIn$, if lastOut is in the same period as lastIn. Otherwise, it is $m + lastOut - lastIn$. For example, $\mathbf{t}_1, \mathbf{t}_4, \mathbf{t}_5$ scheduled at last hop and $\mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_6$ at this hop. Although $\mathbf{t}_6 > \mathbf{t}_5$, the last sending out time-slot is after the last receiving time-slot, but the scheduling delay is not $|\mathbf{t}_6 - \mathbf{t}_5| = 1$. For simplicity, we consider one uniform channel capacity for each available time-slot. We first map $\mathbf{t}_1 \rightarrow \mathbf{t}_2$, $\mathbf{t}_4 \rightarrow \mathbf{t}_6$, $\mathbf{t}_5 \rightarrow \mathbf{t}_3$. Data received at $\mathbf{t}_1$, $\mathbf{t}_4$will be sent out in $\mathbf{t}_2$, $\mathbf{t}_6$, respectively. And data from $\mathbf{t}_5$ will be sent out at $\mathbf{t}_3$ of next period. The lastOut is $\mathbf{t}_3$ of next period, and lastIn is $\mathbf{t}_5$, so the scheduling delay at this hop is $3 - 5 + m$. However, another mapping: $\mathbf{t}_1 \rightarrow \mathbf{t}_3$, $\mathbf{t}_4 \rightarrow \mathbf{t}_6$, $\mathbf{t}_5 \rightarrow \mathbf{t}_2$, can shorten the scheduling delay to $2 - 5 + m$.

Switching Overhead counts the total number of channel switchings incurred for radios at both endpoints (v_i, v_j) for the flow transmissions. Switching overhead calculation is based on the assignment tables of (v_i, v_j).

TABLE I. Assignment Table

	$\mathbf{t}_1$	$\mathbf{t}_2$	$\mathbf{t}_3$	$\mathbf{t}_4$	$\mathbf{t}_5$	$\mathbf{t}_6$	$\mathbf{t}_7$
$\mathbf{R}_{i\_1}$	$\mathbf{f}_1$		$\mathbf{f}_1$		$\mathbf{f}_1$		
$\mathbf{R}_{i\_2}$	$\mathbf{f}_2$	$\mathbf{f}_2$					
$\mathbf{R}_{j\_1}$	$\mathbf{f}_1$		$\mathbf{f}_1$		$\mathbf{f}_1$		
$\mathbf{R}_{j\_2}$	$\mathbf{f}_2$	$\mathbf{f}_2$					
$\mathbf{R}_{j\_3}$							

Scheduling Objective: Our scheduling algorithm first schedules time-slots to allot required flow bandwidth with minimal scheduling delay, then assign channels to minimize radios switching overhead. To amortize the switching overhead (i.e.$80\mu s$), the time-slot will be set relatively large, for example, SSCH chooses $10ms$.

Link Scheduling Algorithm:

We first discuss uniform channel capacity then extend to heterogenous channel capacity.

Uniform channel capacity: Across a link all channels have a uniform capacity. Given the time-slots $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k\}$ reserved at previous hop and a set of available time-slots $\{\mathbf{t}'_1, \mathbf{t}'_2, \ldots, \mathbf{t}'_d\}$ at this hop, generally $k \neq d$, we need to schedule enough time-slots to meet the flow bandwidth requirement. We define the output capacity for each available time-slot $B(\mathbf{t}'_i) = x'_i * c$ and input load for each scheduled time-slot, $B(\mathbf{t}'_i) = x'_i *$, where c is the uniform channel capacity, x'_i is the number of available (channel, radio) pairs in $\mathbf{t}'_i$, x_i is the number of reserved channels in $\mathbf{t}_i$ at previous hop. x'_i = min(♯ of available channels, ♯ of available transmitting radios, ♯ of available receiving radios in $\mathbf{t}'_i$). Since the capacity is the same, each time-slot is associated with its x_i or x'_i to denote its input load or output capacity. $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k\}$ is sorted such that $\mathbf{t}_1$ is the first scheduled slot for v_i to receive and $\mathbf{t}_k$ is lastIn. We first greedily schedule the input load to the earliest available output capacity. If there exists output slot later than the input slot within one schedule period, such earliest output slot is picked. Otherwise, we record down the amount of input load scheduled to the next period in numbers of unit channel capacity, say h. At last we map the first h unit output capacity to transmit the last h unit 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^{th} input load. The detailed algorithm is given in Algorithm 1.

Lemma 1: Algorithm 1 provides the valid schedule with minimized one-hop scheduling delay.

PROOF: Fixed lastIn given in the previous hop schedule, the scheduling delay is determined by lastOut. Our proof consists of two cases.

1930-529X/07/$25.00 © 2007 IEEE

Algorithm 1 MR 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.

if $(\sum_{i=1}^{d} x'_i < \sum_{i=1}^{k} x_i)$ **then**
  RETURN *Scheduling Failure.*
else
  for each $t_i, i \leftarrow 1 \cdots k$ **do**
    while $(x_i > 0)$ **do**
      if there exist $t'_j > t_i$ **then**
        $j = \arg\min\{t'_j | t'_j > t_i\}$
        if $x'_j > x_i$ **then**
          $S = S \cup \{(t'_j, x_i)\}, x'_j = x'_j - x_i, x_i = 0$
        else
          $S = S \cup \{(t'_j, x'_j)\}, x_i = x_i - x'_j$, remove (t'_j, x'_j) from T'_1
      else
        $j = \arg\min\{t'_j\}$
        if $x'_j > x_i$ **then**
          $S = S \cup \{(t'_j, x_i)\}, h = h + x_i, x'_j = x'_j - x_i, x_i = 0$
        else
          $S = S \cup \{(t'_j, x'_j)\}, h = h + x'_j, x_i = x_i - x'_j$, remove (t'_j, x'_j) from T'_1
  Sort the pairs (t''_i, x''_i) in S in ascending order on t''_i.
  $LastOut = t''_q, q = \min\{j | \sum_{i=1}^{j} x''_i \geq h\}$
  if $t_k < t''_q$ **then**
    $Sche\_delay = t''_q - t_k$
  else
    $Sche\_delay = t''_q + m - t_k$

Case 1: lastOut is in the same schedule period as lastIn. In this case, every packet is received and sent out in the same period. Our algorithm greedily schedules the received packets to the earliest available output time-slot. If the optimal ζ is different from our solution, ζ must have at least one swapped pair, that is a pair of packets p_a and p_b, p_a is received earlier than p_b but sent out later. We can switch output time-slot of p_a and p_b. Since p_a arrives earlier 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 the total scheduling delay cannot be longer. For the same reason, after switching all swapped pairs, the scheduling delay cannot be longer than that of ζ, but in this way ζ becomes our solution. In this case, our solution has minimized scheduling delay.

Case 2: lastOut is in the next period of lastIn. Our solution first schedules packets in receiving order and chooses the available output time-slot with minimal buffering delay. Thus, given the input schedule at previous hop and the available time-slots at this hop, our greedy scheduling makes the minimal data delayed to next period. After mapping the delayed h data to the earliest h available output time-slot in next period, our scheduling delay is minimized.

Heterogenous channel capacity: Across a link, different channels have different capacities. For each link $e \in E$, two endpoints sort the channel capacities in decreasing order, $C(e) = \{c(e, \mathbf{f}_1), c(e, \mathbf{f}_2), \ldots, c(e, \mathbf{f}_w)\}$.

The scheduling algorithm for heterogenous capacity is similar to uniform capacity case, except that each scheduled input time-slot and each available output time-slot should be associated with a list of used channels or available channels, instead of just the number of channels. The input load should be scheduled greedily using integer number of available channels in the output time-slot channel list.

For locating the lastOut, we record the total delayed input load instead of number of unit channel capacity. And lastOut is located similarly as uniform capacity case.

Channel Assignment: The reserved channels in each output time-slot need to be assigned to pairs of radios, such that the total channel switching overhead is minimized. Given a list of partial scheduling at this hop:

$$\{(\mathbf{t}_1, L_1, R_1), (\mathbf{t}_2, L_2, R_2), \ldots, (\mathbf{t}_b, L_b, R_b), \}$$

L_i is the set of reserved channels and R_i is the set of available radios on both transmitter and receiver in $\mathbf{t}_i$. We assign channels to radios on one node then the other.

For each tuple $(\mathbf{t}_i, L_i, R_i)$, the problem of assigning all channels in L_i to some pair of radios in R_i with minimal total switching overhead is to find a perfect matching with minimal total weight in a bipartite graph from L_i to R_i. And the weight represents the switching overhead when assign that channel to that radio. This can be solved by Kuhn-Munkras-Algorithm. Therefore, *switching overhead = total weight * unit switching overhead*. Due to the space limitation, the detailed construction of weighted bipartite graph is skipped here.

2) Announce: In the *announce time-slot* (*A-slot*), v_j broadcasts: 1) the updated *reservation_list*; 2) the (s, d) ID; 3) append its ID to *route_list*; 4) the flow ID; 5) any released reservation on itself; 6) the updated remaining latency bound *RLb*; 7) the updated *TTL* as v_j's hop count to d; 8) synchronization message; 9) relay the updated scheduling and release information of some 1-hop neighbors, whose *relay list* contains v_j.

Each neighbor of v_j receives the *A-slot* packet and enters this information into its tables.

3) Confirm: In the *confirm time-slot* (*C-slot*), v_i copies the *A-slot* information to the *C-slot* packet except that change the item 9) accordingly and broadcasts. Each neighbor of v_i receives the *C-slot* packet and updates its tables. Additionally, the *C-slot* packet confirms the scheduling and appoints v_j to establish the next hop for this QoS request.

C. Route Set Up

On receiving a QoS request, the source broadcasts a ROUTE_REQUEST (RREQ). Delay bound combined with

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

hop count are used to restrict RREQ packets flooding. The source sets the initial *TTL* as its hop count to the destination plus some constant. This additional constant allows multiple route set up for one RREQ instead of only the shortest path. When any node receives a RREQ, it will check the *TTL* and run the link scheduling algorithm.

The destination d is expected to receive more than one RREQ for a QoS request. Every RREQ packet indicates a unique path from source to destination under QoS requirements. When the destination d chooses a RREQ packet for the QoS request, it returns a ROUTE_REPLY (RREP) packet by unicasting to the source following the route recorded in *route_list*. The destination copies the <*route_list*> from RREQ to RREP. When the source receives a RREP, the end-to-end route set up is successful for that QoS request, and the data transmissions can begin.

D. Unsuccessful Scheduling

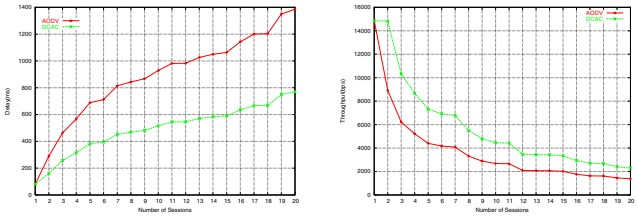
The scheduling failure will occur in partially established routes. The node that runs link scheduling algorithm and gets a scheduling failure will send RESERVE_FAIL (RFAI) packet back to the source following the route recorded in *route_list*. This breakpoint node copies the <*route_list,reservation_list*> from PREQ to RFAI. When RFAI traverses back to the source, each node along the path releases the resources in *reservation_list*, updates its tables, and announces this release information to its neighbors in next *A-slot* or *C-slot*.

IV. Performance Evaluation

Using Qualnet [1], the performance of our proposed protocol was evaluated in both regular and random networks. Figure 1 shows the difference of QoS bounded by the traditional routing protocol AODV and our proposed protocol DCAC in the regular grid network. In the random network, we can also see the advantage of DCAC in respect of throughput and delay. In addition, we evaluated the performance of our protocol in the random network with two QoS channels of different capacities. Due to the space limitation, please refer to the web [2] for the detail of the simulation study.

Note that a multi-channel and multi-radio(MCMR) physical layer model was first established by us. We set up 3 channels of same capacity, which is 2Mbps, in the scenario, whose frequency are 2.412GHz, 2.437GHz and 2.462GHz respectively. The channel of 2.412GHz is defined as the default channel to perform scheduling. The other two channels are dedicated to delivering QoS traffics. The reason why we take the AODV protocol as the comparison object to our protocol is that they share the some procedures of route setup and route maintenance.

[2]http://www.cs.cityu.edu.hk/~amyhuyi/calladmission-simulation.pdf



(a) Delay (b) Throughput

Fig. 1. Different QoS bounded by AODV and DCAC respectively in the grid network.

V. 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 present an efficient DCAC 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.

References

[1] Scalable network technologies, inc. http://www.scalable-networks.com/.

[2] KANODIA, V., LI, C., SABHARWAL, A., SADEGHI, B., AND KNIGHTLY, E. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proc. ACM MobiCom* (2001).

[3] KODIALAM, M., AND NANDAGOPAL, T. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *ACM MobiCom '03* (2003), pp. 42–54.

[4] KODIALAM, M., AND NANDAGOPAL, T. The effect of interference on the capacity of multi-hop wireless networks. In *IEEE Symposium on Information Theory* (2004).

[5] KODIALAM, M., AND NANDAGOPAL, T. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *ACM MobiCom '05* (2005), pp. 73–87.

[6] KUMAR, V. S. A., MARATHE, M. V., PARTHASARATHY, S., AND SRINIVASAN, A. Algorithmic aspects of capacity in wireless networks. *SIGMETRICS Perform. Eval. Rev. 33*, 1 (2005), 133–144.

[7] LEE, S., NARLIKAR, G., PÁL, M., WILFONG, G., AND ZHANG, L. Admission control for multihop wireless backhaul networks with QoS support. In *IEEE WCNC* (Apr. 2006).

[8] LIN, C. R., AND LIU, J. S. Qos routing in ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications* (Aug. 1999).

[9] NARLIKAR, G., WILFONG, G., AND ZHANG, L. Designing multihop wireless backhaul networks with delay guarantees. In *INFOCOM* (2006).

[10] RANIWALA, A., AND CHIUEH, T. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *INFOCOM* (2005).

[11] SRIRAM, S., REDDY, T. B., MANOJ, B. S., AND MURTHY, C. S. R. On the end-to-end call acceptance and the possibility of deterministic QoS guarantees in ad hoc wireless networks. In *Proc. 6th ACM Mobihoc* (2005).

[12] TANG, J., XUE, G., AND ZHANG, W. Interference-aware topology control and QoS routing in multi-channel wireless mesh networks. In *Proc. 6th ACM Mobihoc* (2005), pp. 68–77.

[13] WANG, W. Z., WANG, Y., LI, X. Y., SONG, W. Z., AND FRIEDER, O. Efficient interference-aware TDMA link scheduling for static wireless networks. In *Proc. of ACM MobiCom* (2006).