A Novel Queue Length Aware Distributed Link Scheduler for Multi-Transmit Receive Wireless Mesh Networks

Yuanhuizi Xu, Kwan-Wu Chin, Raad Raad
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, NSW, Australia
Email: yx879@uowmail.edu.au, {kwanwu, raad}@uow.edu.au

Sieteng Soh
Department of Computing
Curtin University of Technology, WA, Australia
Email: s.soh@curtin.edu.au

***Abstract*—Next generation Wireless Mesh Networks (WMNs) will require a link scheduler that exploits the full advantage of Multi-Transmit-Receive (MTR) commuication. To this end, we design a distributed link scheduler called Voting-ALGO that is aware of queue lengths and uses the celebrated max weight policy to achieve 100% throughput.**

I. Introduction

In Multi-Transmit-Receive Wireless Mesh Networks (MTR WMNs), whereby nodes are equipped with adaptive arrays, nodes can transmit simultaneously to (SynTx), **or** receive simultaneously from (SynRx) their respective neighbouring nodes. Hence, a key constraint of any channel access or link scheduler over such WMNs must adhere to this so called Mix-Tx-Rx capability.

To this end, we require a sheduler that addresses the following problem: given an MTR WMN with stochastic arrival rates, design a throughput optimal, distributed algorithm that activates the highest weighted links. Since the seminal contribution of Tassiulas et al. [1], who proposed a *centralized* maximum weight policy that is shown to be throughput optimal using only topological and queue length information, a number of researchers have sought distributed schedulers; see [2]. These schedulers, however, assume a k-hop interference model, and thus do not consider the MTR capability of nodes.

To date, only a handful of works have investigated centralized and distributed link scheduling algorithms for MTR WMNs. The authors of [3] proposed a distributed solution that allows nodes to hold the channel longer based on traffic demand. A key observation is that this algorithm has not considered queue lengths. Indeed, only Bao et al. [4], who proposed receive-oriented multiple access (ROMA) scheme, have considered link weights, whereby a heavier weight corresponds to a higher probability of activation. However, ROMA only provides four different weight values and no solution is presented on how they can be adjusted to achieve queue stability. In contrast, our proposed algorithm, called Voting-ALGO, provides a deterministic method that schedules the highest weighted links in each transmission slot. Moreover, it provides the following advantages: 1) it does not have excessive communication overheads associated with data collection, 2) it uses queue lengths to evenly split nodes into transmitter and receivers at each time slot. Experimental results show Voting-ALGO achieves higher capacity than prior distributed link schedulers designed for MTR WMNs.

Next, we outline our network model, and formalize the problem in Section III. Voting-ALGO, is outlined in Section IV. Section V presents our evaluation and results, followed by our conclusions in Section VI.

II. Network Model

Consider a WMN represented by a directed graph $G(V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and $E = \{(u, v) \in E \mid u, v \in V\}$ is the set of directed links. Each node has a unique ID. All nodes are equipped with $b \geq 1$ directional antennas that share a single channel. We assume $b_u \geq |N(u)|$ for all nodes, where $N(u)$ is the set of neighbors for node u. Each node has a transmission range of r. We consider a link exists if node u and v are within each other's transmission range. Thus, two nodes that are located within a distance smaller than the transmission range are regarded as neighbors. At any point in time, a node u can transmit to or receive from $|N(u)|$ neighbors at the same time, but it cannot receive *and* transmit simultaneously because of side-lobes. We will refer to this as the *no Mix-Tx–Rx* interference constraint. In addition, we consider the 1-hop traffic model where all routes consist of one link. In this respect, we will denote the transmitting end of a link as source, and the corresponding end as receiver.

We assume time is divided into slots of equal length, denoted by t. The packets are of unit length. In this paper, we consider the Bernoulli process with parameter λ, which is known as the arrival rate. We use a 0-1 vector with dimension $|E|$, and denoted as $A(t)$, to represent a packet arrival at each queue/link at the beginning of slot t. We assume the arrivals are independent identically distributed. The arriving packet is stored in the queue associated to said link. The vector $A(t)$ is calculated based on the arrival rate λ. Let $Q_{u,v}(t) \geq 0$ represent the number of packets in the queue of link (u, v) at time t. Thus, the queue size vector is denoted by $Q(t) = [Q_{u,v}(t)]$. The vector $D(t) \subseteq \{0, 1\}^{|E|}$ denotes transmitting links at the end of slot t that adhere to the no Tx-Rx constraint. Thus a link (u, v) is active if $D_{uv}(t) = 1$.

TABLE I
MESSAGES USED IN VOTING-ALGO

Message	Description
$Q_{s,d}$	The queue-length of link (s, d).
$INq(v)$ and $OUTq(v)$	The total number of incoming and outgoing packets at node v respectively.
INsetT and INsetR	Notification messages sent by nodes in *setT* and *setR* respectively to inform their neighbors which set they joined.
$Vote\{s \rightarrow d, Q_{s,d}\}$	Vote message sent by nodes in *setT* or *setR* to their neighbors in *setU*.
$INq'(v)$ and $OUTq'(v)$	The number of incoming and outgoing packets from neighbors in *setT* or to neighbors in *setR* at node v based on received vote messages.

Consequently, the dynamic queue length of the network model can be described as $Q(t) = A(t) + Q(t-1) - D(t-1)$.

III. PROBLEM DEFINITION

We now formalize the problem at hand. Given queue length information, design a throughput optimal distributed algorithm. Formally, $\arg\max_D(T)\lambda$ s.t. $\lambda \in \Lambda$ (or $\arg\max_D(T)\sum Q(T) * D(T)$ s.t. $\lambda \in \Lambda$, where T denotes the throughput). As mentioned earlier, the Max Weight policy is throughput optimal. However, it is centralized and requires instantaneous queue length information, which is not practical in WMNs. In particular, the centralized policy will incur many rounds of requests, and signalling overheads as well as propagation delays, which result in stale queue size information. These limitations thus motivate the design of a *distributed* algorithm that is throughput optimal.

IV. THE DISTRIBUTED POLICY

We now expound a distributed policy called Voting-ALGO. We assume each node has a distinct ID. The basic idea is as follows. We divide each time slot into two sub-slots: *control* and *data*. The first sub-slot is used for scheduling, during which time each node determines whether it is in transmitting set *setT* or receiving set *setR*. The second sub-slot *data* is used for packets transmission. That is, at such time, all the nodes in *setT* transmit one packet to each of their neighbors that are in *setR*. Each control sub-slot is further divided into a number of mini-slots. In the mini-slots, nodes exchange control messages listed in Table I. Note, we denote the vote message as "$Vote\{s \rightarrow d, Q_{s,d}\}$", where s and d represent the source and destination nodes of link (s, d), and $Q_{s,d}$ is the queue length of link (s, d).

Our Voting-ALGO consists of three key stages: **initialize**, **notify** and **vote**. In the first stage, as shown in Algorithm 1, the goal is to select the first batch of nodes that are to join *setT* or *setR*. All other nodes move into *setU*, where $setU = V - setT - setR$. Algorithm 2 outlines the pseudocode performing **notify**, whereby nodes that have entered *setT* or *setR* in Stage-1 inform their neighbors which set they have joined by sending the message "INsetT" and "INsetR". After that, nodes proceed to the key stage **vote** which is demonstrated in Algorithm 3. The main idea is to determine which set, i.e., *setT* or *setR*, should nodes in *setU* join based on the votes cast by their neighbors. To achieve this, after receiving all the votes, each node calculates its INq' and $OUTq'$ which represents the total number of votes that elect the node a *setR* or a *setT* node. If INq' is greater than or equal to $OUTq'$, the node is included in *setR*. Otherwise, it joins *setT*. From this point onwards, Stage-2 and Stage-3 is carried out alternately until there is no node left in *setU*.

Algorithm 1: Stage-1: initialize

Input: node v, neighbors $N(v)$, $INq(v)$, $OUTq(v)$, $INq(N(v))$ and $OUTq(N(v))$
Output: $set(v)$

```
if INq(v) > max(INq(N(v))) then
    set(v) ← "setR"
else if OUTq(v) > max(OUTq(N(v))) then
    set(v) ← "setT"
else if INq(v) = max_{∀u∈N(v)}(INq(u)) or OUTq(v) = max_{∀u∈N(v)}(OUTq(u)) then
    for u ∈ N(v) do
        if INq(u) = max_{∀u∈N(v)}(INq(u)) and v > u then
            set(v) ← "setR"
        else if OUTq(u) = max_{∀u∈N(v)}(OUTq(u)) and v > u then
            set(v) ← "setT"
    end
else
    set(v) ← "setU"
end
return set(v)
```

Algorithm 2: Stage-2: notify

Input: node v, $set(v)$, neighbors$N(v)$ and $minislot$

```
minislot ← minislot + 1
if set(v) ="setT" then
    v transmits "INsetT" to every u ∈ N(v)
end
minislot ← minislot + 1
if set(v) ="setR" then
    v transmits "INsetR" to every u ∈ N(v)
end
```

V. EVALUATION

In order to investigate the performance of Voting-ALGO, we use MatGraph [5], a Matlab toolkit that works with simple graphs. Our Matlab experiments are conducted over topologies with 50 nodes, whereby the degree for each node is fixed and equals to five. We compare Voting-ALGO against link based centralised algorithm (LBC-ALGO), node based centralised algorithm (NBC-ALGO), JazzyMAC [3] and ROMA [4]. Both LBC-ALGO and NBC-ALGO require the global queue length information. LBC-ALGO selects the set of heaviest-weighted,

Algorithm 3: Stage-3: vote

Input: node v, $set(v)$, neighbors$N(v)$, $location(N(v))$
Queue-lengths $Q_{v,N(v)}$, $Q_{N(v),v}$ and $minislot$
Output: $set(v)$

$minislot \leftarrow minislot + 1$
$W \leftarrow \emptyset$
$Y \leftarrow \emptyset$
if $set(v) \neq$ *"setU"* **then**
  for $u \in N(v)$ ***and*** $set(u) =$*"setU"* **do**
    if $set(v) =$*"setT"* **then**
      v transmits $Vote\{v \rightarrow u, Q_{v,u}\}$ to u
    else if $set(v) =$*"setR"* **then**
      v transmits $Vote\{u \rightarrow v, Q_{u,v}\}$ to u
  end
else if $set(v) =$*"setU"* **then**
  for $u \in N(v)$ ***and*** $set(u) =$*"setT"* **do**
    $W \leftarrow W \cup \{u\}$
  end
  forall the $w \in W$ **do**
    v receives $Vote\{w \rightarrow v, Q_{w,v}\}$ from w
  end
  $INq'(v) =) \sum_{\forall w \in W} Q_{w,v}$
  for $u \in N(v)$ ***and*** $set(u) =$*"setR"* **do**
    $Y \leftarrow Y \cup \{u\}$
  end
  forall the $y \in Y$ **do**
    v receives $Vote\{v \rightarrow y, Q_{v,y}\}$ from y
  end
  $OUTq'(v) = \sum_{\forall y \in Y} Q_{v,y}$
  if $INq'(v) \leq OUTq'(v)$ **then**
    $set(v) \leftarrow$"setT"
  else
    $set(v) \leftarrow$"setR"
  end
return $set(v)$

non-interfering links at every time slot to transmit, where weight is the length of queue of each link. While the queue length variable used in NBC-ALGO denotes the sum of the queue length of every outgoing links at each node. In every slot, the nodes with longest queue length are placed into *setT* and their neighbours into *setR*.

In the experiments, we assume all traffic traverse a single hop. A packet arrival occurs with probability λ at the beginning of a time slot. The arrivals are independent and identically distributed according to a Bernoulli process.

Figure 1 shows the mean total queue backlog summed over all links as the arrival rate λ increases. The error bars shown indicate 95% confidence interval of the mean value, which represents the true queue length. When λ approaches a certain limit, the average total backlog will increase to infinity. This limit is viewed as the boundary of the capacity region. The results are collected after 100,000 time slots.

From Figure 1, we can see that, generally, Voting-ALGO is superior to other algorithms as it ensures all queues are stable with the same traffic load. We see that Voting-ALGO achieves the same capacity region as LBC-ALGO. This result indicates that the performance of Voting-ALGO is approximately similar to centralized approaches. In other words, Voting-ALGO is throughput optimal.

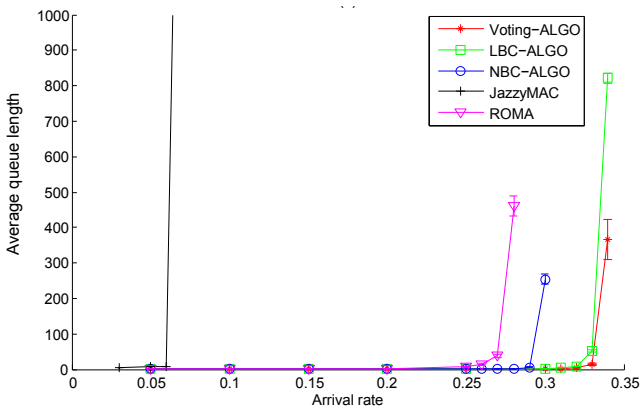

Fig. 1. Queue length under increasing arrival rate with 95% confidence interval

VI. Conclusion

This paper has presented a novel queue aware distributed link scheduler for MTR WMNs whereby nodes have the capability to form multiple links simultaneously. Experiment results show the throughput achieved by Voting-ALGO is two to nine times higher than that of JazzyMAC. An immediate future work is to further reduce signaling messages and to consider the case when the number of directional antennas is less than the number of neighbors.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[2] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross layer control in wireless networks," *IEEE Transactions on Networking*, vol. 14, pp. 302–315, Apr. 2006.

[3] S. Nedevschi, R. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer, "An adaptive, high performance mac for long-distance multihop wireless networks," in *ACM MOBICOM*, (New York, NY, USA), pp. 259 –270, 2008.

[4] B. Lichun and J. Garcia-Luna-Aceves, "Transmission scheduling in ad hoc networks with directional antennas," in *ACM MOBICOM*, (New York, NY, USA), 2002.

[5] E. R. Scheinerman, "Matgraph: a MATLAB toolbox for graph theory," *Department of applied mathematics and statistics, the Johns Hopkins University, Baltimore, Maryland*, pp. 1– 7, 2008.