Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 312828, 11 pages
doi:10.1155/2010/312828

Research Article

Utilization-Based Dynamic Scheduling Algorithm for Wireless Mesh Networks

Miray Kas,[1] Ibrahim Korpeoglu,[2] and Ezhan Karasan[3]

[1] *Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, USA*
[2] *Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey*
[3] *Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey*

Correspondence should be addressed to Miray Kas, miraykas@gmail.com

Received 15 April 2010; Accepted 5 October 2010

Academic Editor: Amiya Nayak

Channel access scheduling is one of the key components in the design of multihop wireless mesh networks (WMNs). This paper addresses the allocation/demand mismatch problem observed in oblivious WMN channel access scheduling schemes and proposes Utilization-Based Scheduling (UBS). UBS is a Spatial-TDMA- (STDMA-) based dynamic channel access scheduling scheme designed with the aim of increasing the application-level throughput. In UBS, each node has a weight, which is dynamically adjusted in accordance with the node's slot usage history and packet-queue occupancy. UBS is a fully distributed algorithm, where each node adjusts its own weight and makes pseudorandom transmission attempts using only the locally available information. To demonstrate the performance improvements of the dynamic weight adjustment, the performance of UBS is compared against other channel access scheduling schemes through extensive ns-2 simulations under both uniform and nonuniform traffic patterns.

1. Introduction

Wireless mesh networks (WMNs) emerged as a key technology having various advantages, especially in providing cost-effective coverage and connectivity solutions both in rural and urban areas. Channel access scheme is one of the major components of a wireless mesh network (WMN) which is required to keep the network functional. The channel access scheduling, which is controlled by the MAC protocol, significantly affects the overall network performance. In WMNs, typically FDMA-, CDMA-, or TDMA-based mechanisms are employed for multiple access coordination as CSMA-based schemes are known to result in inferior performance in multihop networks with high traffic demands [1].

MAC protocols currently available in the literature can be broadly classified into two groups. The first group of MAC protocols are contention-based protocols, where nodes contend for channel access and collisions are possible. The 802.11 MAC protocol [2] which is based on carrier sense multiple access/collision avoidance (CSMA/CA) and Aloha system [3] constitute two well-known examples of contention-based MAC protocols. Although they are widely used, contention-based MAC protocols are known to provide inferior performance in multihop wireless mesh/ad hoc networks, especially when the number of contending nodes is increased [1].

The second group of MAC protocols are schedule-based protocols, where the access of nodes or links to the channel is scheduled in advance. In schedule-based protocols, the overall achievable throughput and the delay performance of the network highly depend on the scheduling algorithm. TDMA-based protocols, for instance, operate in discrete (i.e., slotted) time and typically arrange the transmission of the nodes or links in the network based on a schedule, forming a subcategory of the schedule-based protocols. Given that the forthcoming WMN standards such as WiMAX [4] and 802.11s [5] consider STDMA-based MAC mechanisms and WMNs operate in multihop environments, in this paper, we focus on STDMA-based schemes at the MAC layer.

1.1. Motivation and Contributions. In statically weighted scheduling schemes, nodes are assigned static weights that

remain constant throughout the network lifetime. However, these statically assigned weights do not always map very well to the dynamic traffic conditions passing through the nodes. Unless a dynamic readjustment scheme is used, the mismatch between the statically assigned weight of a node and its actual traffic load may never be eliminated.

Excessive allocation and excessive demand are the two main cases where a node's statically assigned weight does not map well to its actual load.

(1) Excessive allocation: as static weight assignment schemes are unaware of the current network conditions, in the case of excessive allocation, a node is given transmission opportunities typically at the cost of suppressing other nodes even if it has no packets to transmit. This causes the channel resources to be wasted. The severity of this situation can be identified by keeping a counter for the number of slots each node wastes. A dynamic weighting scheme which tries to alleviate this problem should periodically measure the amount of waste caused by the most recent weight assignments and reset those counters.

(2) Excessive demand: in the case of excessive traffic demand, the amount of the transmission opportunities given to a node is insufficient, causing the node to become congested. Since the number of packets in a node's packet queue is a direct indicator of its demand for transmission slots, in order to alleviate the congestion observed, we directly monitor the state of the nodes' packet queues.

Realization of excessive allocation and excessive demand phenomena motivate us to devise a utilization-based packet scheduling policy to alleviate these problems. The main contributions of this paper are (i) the proposal of a novel dynamic weighting scheme and (ii) the proposal of a channel access scheduling scheme called Utilization-Based Scheduling (UBS), which employs our proposed dynamic weighting scheme. UBS is a fully distributed STDMA-based scheduling scheme which aims to prevent slot wastes while providing enough transmission opportunities to all nodes. To achieve this, we make use of the following ideas: (i) maintaining the slot usage statistics, (ii) periodically monitoring each node's packet queue, and (iii) adjusting each node's slot allocation in an adaptive fashion.

1.2. Paper Organization. The remainder of this paper is organized as follows. Section 2 provides background on STDMA-based channel access schemes and briefly reviews the proposals available in the literature. Section 3 presents the proposed dynamic weighting scheme and UBS's scheduling decision mechanism in detail. Section 4 reports our ns-2 simulation results, and finally, Section 5 concludes the paper emphasizing the key insightful points.

2. Literature Review

In the literature, there are many different studies on TDMA-based channel access schemes, each following a different methodology. Most of the channel access schemes proposed in the early 2000s are oblivious to demand/allocation mismatch and designed under strong assumptions. For instance, some are cluster based [6], some are hybrid (e.g., Z-MAC [7], NAMA [8]), and there are some others such as TRAMA [9] that perform node access scheduling considering design-specific metrics. However, these studies are valuable as they focus on generic mechanisms and pave the way to the current more complex proposals.

NAMA [8] assumes that knowledge about two-hop neighborhood is achieved somehow, and the nodes have mutual knowledge. A seed value for random value generation is formed as the combination (node-id, contended-slot-number) for each of the 2-hop neighbors and a winner, the node that draws the highest random value for the contended slot, is elected as the winner of the slot. TRAMA (Traffic Adaptive Medium Access) [9] is another TDMA protocol which assigns time slots to the nodes through the use of one-hop traffic information and two-hop neighborhood information. For each transmission time, each node selects one of the transmitting, receiving, stand-by modes. The nodes with no data to send are not involved in the election. For instance, Z-MAC [7] is a dynamic scheme which starts as CSMA and switches to TDMA when the load exceeds a certain threshold. In Z-MAC, each node runs a distributed slot selection algorithm and the owner of the slot uses a smaller random backoff value.

Recently, emergence of new wireless standards [4, 5, 10] has accelerated the research in channel access schemes. For instance, IEEE 802.16 received wide attention from research community and industry mainly due to its being a promising technology and standards, leaving performance-sensitive parts open for vendors implementation. Although the standard specifies control messages, the details of scheduling mechanism (i.e., allocation of data slots for transmission) in mesh mode are left open for further research. However, most of the research efforts in this direction focus on forming a centralized routing tree and performing scheduling taking the root of the tree as a decision point [11–14].

In this respect, our contribution is orthogonal to these studies mainly due to two reasons. First, we focus on performing channel access scheduling in a fully distributed manner rather than following a centralized approach. There are three major reasons why a fully distributed algorithm is desirable [15]: (i) the distributed approach eliminates the potential problems with single point of failure, (ii) it avoids additional overhead of communicating with the central coordinator, and (iii) the nodes can continue communication even if the connection to the central coordinator is lost. The second reason for our study to be orthogonal to studies such as [11, 14] is that these recent standard specific proposals focus on the usage of standard-specific message types and their integration in the framework. However, we do not deal with integration of different standards' MAC-level packet types; we rather focus on a general channel access scheduling approach which might be applicable/integrable to different standards if desired.

Reference [16] is another distributed scheduling related work that has been recently proposed with the aim of

providing more fairness via propagating the scheduling requirements within the network. Reference [17] considers TDMA-based distributed link scheduling via providing modifications to the traditional edge-coloring algorithm. Reference [18] is another recent work which has also been inspired from one of the traditional algorithms, namely, the Bellman-Ford algorithm. In another work [19], the authors frame TDMA scheduling as a network flow problem on the conflict graph of the network, aiming for the minimization of the delay in multihop networks. References [20–22] are among other works that focus on throughput maximization in the context of distributed scheduling in wireless mesh networks.

3. Utilization-Based Dynamic Channel Access Scheduling Scheme

3.1. Network Model. The basic features of our network model are as follows.

(i) Each node is uniquely identifiable.

(ii) Node and time synchronization are available. However, the methods for achieving synchronization are out of the scope of this paper.

(iii) The targeted system operates in discrete (or, slotted) time. A maximum-sized packet can fit into a time slot.

(iv) Communication is established via omnidirectional antennas over a single physical radio channel.

(v) Each node in the network has a single half-duplex radio, and the nodes' radios are always on.

(vi) Each node keeps a single-packet queue, not differentiating the packets from different connections.

(vii) 2-hop interference model is used as the interference model, which assumes that the interference between the nodes that are separated by more than 2-hops in the physical topology is negligible [23].

3.2. Overview of UBS Mechanism. In UBS, each node aims to adjust its own weight independently such that the network resources are utilized effectively, resulting in an increase in the overall throughput. UBS divides the time into equal intervals called *frames*. Each node is assigned a dynamic weight value which approximates the node's demand for transmission slots in the next frame. The number of time slots assigned to each node in a single frame is proportional to its weight. Each node periodically runs our Additive Increase Multiplicative Decrease- (AIMD-) based weight adjustment algorithm (Algorithm 1) and decides to increase/decrease its weight using the readily available information such as its slot utilization history and queue occupancy. UBS weighting scheme is composed of 3 basic mechanisms:

(1) node state detection,

(2) dynamic weight adjustment, and

(3) weight sharing (weight dissemination).

```
if wastedSlots = 0 then
   if (q_avg > 0) then
      AdditiveIncrease();
   end if
else
   MultiplicativeDecrease();
end if
wastedSlots ← 0;
```

Algorithm 1: UBS weight adjustment.

Each node detects its state periodically and adjusts its own weight. Then, these weights are shared among the nodes within the same 2-hop neighborhood and used for schedule formation. Schedule formation is performed via a pseudorandomized election mechanism, where each node has as many agent as its weight contending on its behalf.

3.3. Dynamic Weight Adjustment Scheme. Considering configurations complying with our network model, following information is available at every node X and might be used for weight-adjustment purposes:

(1) detailed information about 1-hop and 2-hop neighbors of node X,

(2) instantaneous packet queue length of node X,

(3) maximum size of packet queue,

(4) number of time slots allocated to node X in each time frame,

(5) number of time slots wasted by node X in each time frame.

In the following subsection, we discuss the three mechanisms of UBS weighting scheme and explain how UBS exploits the above-listed information for dynamic weight information.

3.3.1. Node State Detection. Nodes periodically detect their current states. Node state detection is composed of two different parts: (1) detection of the packet queue state and (2) maintenance of the slot usage statistics.

At the end of each time frame, each node samples its instantaneous queue length. The instantaneous queue lengths are held in a sliding window which is then used to calculate the average queue length, q_{avg}, over T_w frames. T_w is a network parameter which denotes the *weight-adjustment period*. The impact of T_w on the overall network performance is discussed in Section 4.1.

Each node in the network also keeps its slot usage statistics through a counter for the number of slots it has wasted, which is reset every T_w frames. A slot is considered to be wasted if a node is given the opportunity to transmit at a particular time slot although it has no packets to send. The obtained state information is then used within the weight adjustment.

3.3.2. Weight Adjustment. UBS uses an Additive Increase/Multiplicative Decrease- (AIMD-) based weight adjustment mechanism to keep nodes' weights well-matched with their actual demands. AIMD in TCP literature represents a congestion control mechanism with a linear growth of congestion window combined with an exponential (multiplicative) decrease in the case of congestion [24].

Different from AIMD in TCP, in UBS, what is adjusted is not the size of the congestion window, but the weight of the node. AIMD approach is shown to converge [25], and it is also shown to be the right approach especially when the source is operating close to the availability of the network [26].

Each node invokes UBS weight-adjustment algorithm (Algorithm 1) once in every T_w frames and the algorithm is triggered to take action under two different conditions:

(1) If there are no wasted slots and there are packets waiting in the queue. (In this case, the node's weight is increased.)

(2) If there are wasted slots observed since the last adjustment of the weight. (In this case, the node's weight is decreased.)

If there are no slots wasted by a node and if there are packets still waiting in the queue, it can be inferred that the node is in need of more slots because its queue could not be drained although it did not waste any of the time slots assigned to it. Hence, the weight of the node X, W_t^x, is increased using the formula given in (1)

$$W_t^x = W_{t-1}^x + \max\left(1, \left\lfloor \gamma * \log_{10}\left(\frac{q_{\text{avg}} * 100}{q_{\max}}\right) \right\rceil\right). \quad (1)$$

According to (1), a node's weight is increased in accordance with its average queue occupancy percentage. We name the parameter γ as the *increase coefficient*, which is able to change the range of values used for incrementing the weight values.

When $\gamma = 1$, the result from the multiplication of γ by the logarithm of the queue percentage is rounded to the closest integer, returning 0, 1, or 2. Since the maximum of 1 or the logarithm of the queue percentage in base-10 is chosen as the increase step size, the increase step size is either 1 or 2. We use $\max(1, \gamma * \log_{10}((q_{\text{avg}} * 100)/q_{\max}))$ as the increase step size in order to ensure that a node's weight is incremented whenever incrementing the weight value is determined to be required. In our simulations, we use γ as 2, allowing 1, 2, 3, or 4 to be used for incrementing the weight values. In Section 4.1, the impact of the *increase coefficient* parameter, γ, is further investigated. additive increase algorithm (Algorithm 2) implements (1).

On the other hand, if the node has wasted one or more slots since the last invocation of the UBS weight adjustment algorithm, then it implies that node has been assigned more slots than it actually needed. In this case, the node's weight is reduced via multiplicative decrease

$$W_t^x = \max\left(1, \left\lfloor W_{t-1}^x * \left(1 - \frac{\#(\text{Wasted Slots})}{2 * \#(\text{Allocated Slots})}\right)\right\rfloor\right). \quad (2)$$

```
increment ← 0
if q_avg / q_max ≠ 0 then
    increment ← round(γ * log10((q_avg * 100) / q_max));
enf if
if increment = 0 then
    increment ← 1;
end if
weight ← weight + increment;
```

Algorithm 2: Additive increase.

```
weight ← ⌊(weight * (1 − #(Wasted Slots) / (2 * #(Allocated Slots))))⌋;
if weight = 0 then
    weight ← 1;
end if
```

Algorithm 3: Multiplicative decrease.

By using (2), the weight is at most reduced to its half. If the weight rounds down to 0, it is restored to 1 to guarantee the participation of each node in the schedule formation elections because only nodes with nonzero weights are eligible to participate. Multiplicative decrease algorithm presented in Algorithm 3 implements (2). Multiplicative decrease algorithm is also useful in ensuring that nodes are not allowed increase their weights maliciously. When a node increases its weight maliciously, then the number of slots it has wasted will come into play as a factor stopping this artificial increase, making UBS robust to such attacks.

Alternatively, weight adjustment could be performed using an Additive Increase/Additive Decrease- (AIAD-) based algorithm. We have observed that AIMD performs better than AIAD. If a node wastes its assigned slots, it is very likely that this node will not need more slots in the following frames. When the number of its assigned slots is reduced aggressively as in AIMD, the nodes that need more slots will be able to get what they need quicker. In AIAD, the transition of slots is slower; hence, the performance is lower.

3.4. Scheduling-Formation Mechanism. In networks using the 2-hop interference model, there are two main types of conflicts that should be avoided in order to achieve a collision free schedule:

(1) primary conflict: observed if a node is scheduled to transmit and receive at the same time, and

(2) secondary conflict: observed if a node is scheduled to receive from two different nodes simultaneously.

In order to ensure that both kinds of conflicts are avoided, no two nodes within the same 2-hop neighborhood should be scheduled to transmit at the same time slot [27, 28].

```
localAgtLst ← FormLocalAgents()
nbrAgtLst ← FormNbrAgents()
nbrAgtLst ← nbrAgtLst ⋃ localAgtLst
for i = 1 to FRAME_SIZE do
   hashAgtPairs ← MeshElection (contestTime, nbrAgtLst)
   winnerAgt ← FindMaxHash(hashAgtPairs)
   if localAgentsLst.contains(winnerAgt) then
      slots[i] ← WON
   end if
end for
```

Algorithm 4: Scheduler algorithm.

UBS is a distributed weighted channel access scheme where each node determines the time slots it will use for transmission based on the information about its 1-hop and 2-hop neighbors (including the weight information) that can be collected by a link-layer or network-layer mechanism.

The scheduler algorithm (Algorithm 4) is independently run by each node at the end of each frame in order to select the time slots the node is eligible to transmit during the next frame.

The first two lines of the scheduler algorithm is where UBS weight information is actually used. In the first two steps of the scheduler algorithm, the set of AgentIDs for the owner node's agents (localAgtLst) and the set of AgentIDs of its 2-hop neighbors' agents (nbrAgtLst) are generated. Each node, including the local node, has as many agents as its weight where all of its agents compete to win slots on its behalf. Therefore, for each slot, the winning probability of a node is roughly proportional with the number of its agents, hence its weight. All agents generated in these two steps are involved in all contentions held for the entire frame.

In the for loop, a separate contention (MeshElection) is held for each time slot in a frame. MeshElection takes two parameters: (1) the identifier of the time slot the contention is held for (e.g., *contestTime*) and (2) the set of contenders (e.g., *nbrAgtLst*). MeshElection function returns a set of pairs where each pair involves the AgentID and its corresponding hash value. The agent with the largest hash value is then elected as the winner of the contended time slot. If the winner AgentID belongs to localAgtLst, then the node marks the slot as one of the slots it is eligible to transmit.

To calculate the hash values, we use the smear function given in 802.16-2004 standard [4] that converts a uniform value to an uncorrelated uniform hash value through mixing. To elaborate more, it is a function with a number of arithmetic operations (e.g., $\ll$,$\gg$, +, −). It uses these operations to mix the positions of the bits that represent the seed value given as the parameter to the function, and the outcome is a uniformly distributed hashing value. Hash-value generation is the pseudorandomized part of UBS as it takes the combination of the AgentID and the ContestTime as its unique, pseudorandom seed value.

The hash value is obtained as follows:

$$\text{hash\_value} = \text{smear}(\text{AgentID} \wedge \text{ContestTime}). \quad (3)$$

Smear function is very simple to implement in hardware, and the time it takes to return an answer is significantly less than those of traditional random number generator functions. Therefore, we prefer using this smear function over a random number generator.

3.5. Weight Sharing. To be able to form a channel access schedule, each node needs to know the weights of nodes in its vicinity. To keep the messaging overhead required for the weight dissemination at minimum, we propose a distributed pseudorandom scheduling mechanism and exploit the capabilities of the network layer routing protocol to learn about the neighboring nodes and the network topology. For the routing protocol, we use OLSR [29], which is a table-driven link state routing protocol collecting various information in its tables. To disseminate UBS weight information without incurring any communication overhead, we replace the unused reserved fields in OLSR's periodic HELLO messages. The mechanism of UBS is independent of OLSR's functionality. Any routing protocol performing periodic status broadcasts can be used with UBS. However, in the rest of the paper, we assume that OLSR is the network layer routing protocol.

The HELLO message structure given in RFC 3626 [29] has "Reserved" fields which are unused and filled with zeros. "Reserved" field within the local information section is 2 bytes while "Reserved" field in the link information section is 1 byte long.

We extend the HELLO message structure specified in RFC 3626 to include weight information for the originating node itself and its listed 1-hop neighbors. The proposed modified message structure is shown in Figure 1. In the new message structure, the second half of the "Reserved" field within the local information section is replaced with the "Weight" field and the "Reserved" field in the link information section is substituted with "Nb_Weight" field. In a single HELLO message, there is only one "Weight" field, but there might be multiple "Nb_Weight" fields depending on the number of the 1-hop neighbors advertised. Both "Weight" and "Nb_Weight" fields are of 1 byte long. "Weight" field holds the weight information of the originating node. The "Nb_Weight" field holds the weight information for the advertised neighbor node.

Using this new HELLO message structure, every node is able to collect the weight information of all the nodes in its 2-hop neighborhood without requiring the MAC layer to exchange any further messages. There is *no* communication overhead introduced by UBS weighting scheme as the unused parts of OLSR HELLO messages are utilized for the dissemination of the weight information.

In the implementation of UBS, only Algorithm 1 is allowed to change a node's weight at the MAC layer. OLSR is not allowed to trigger the recalculation of the weight when a HELLO message is about to be sent. When preparing a HELLO message, OLSR extracts the most recently calculated weight from the MAC layer to fill in HELLO message's weight field and the most recently learned neighbor weights to fill in Nb_Weight fields.

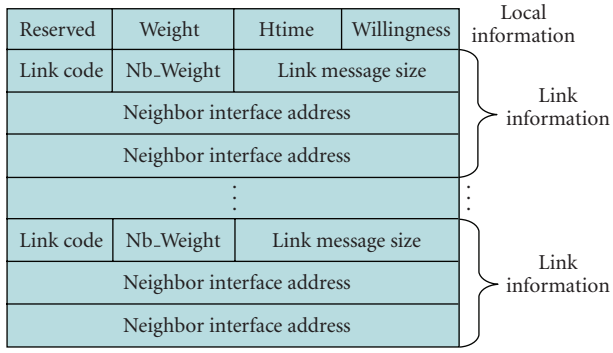


Figure 1: OLSR HELLO message formats modified to include weight information.

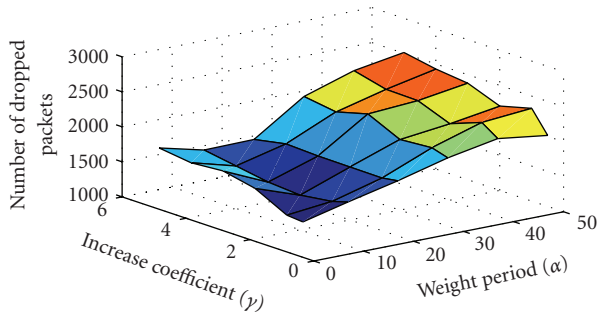


Figure 2: The effects of UBS-related parameters on the overall performance of the network, in terms of number of dropped packets.

4. Simulation Results and Analysis

In this section, we first analyze the effects of UBS design parameters on the overall performance of the network. Then, we present our ns-2 simulation results and qualitatively and quantitatively compare UBS against two other schemes: uniform-weighted TDMA and nonconcurrent TDMA. In uniform-weighted TDMA, weight information is not used, assuming all nodes weights are equal to 1. In nonconcurrent TDMA, concurrent transmissions are not allowed; each time slot can be used by a single node. Simulations are performed using both uniform and nonuniform traffic patterns aiming to show (i) the impact of allowing concurrent transmissions on nonconflicting parts of the network (i.e., exploiting spatial reusability) and (ii) the contribution of UBS's dynamic weighting scheme to the overall network throughput. In order to alleviate the topology effect which is discussed further in Section 4.2, the simulation results presented in this section are calculated as the averages of multiple simulations over 10 different randomly generated 20-node connected networks.

4.1. Effects of UBS Design Parameters. In this section, the effects of two UBS parameters, the *weight-adjustment period* (T_w), and the *increase coefficient* (γ) on the overall performance are discussed.

Table 1: Ns-2 simulation parameters.

MAC Parameters	Value
Bandwidth	3 Mbps
Max packet length	1500 bytes
Frame size	100 slots
Number of nodes	20 nodes
Sliding window size	10

Figure 2 presents the impact of UBS parameters γ and T_w on the performance of UBS in terms of the number of dropped packets, while other simulation parameters are set to the values presented in Table 1. The results presented in Figure 2 represent the average of multiple simulation runs performed on 3 different 20-node networks under 3 different traffic scenarios with different values of γ and T_w.

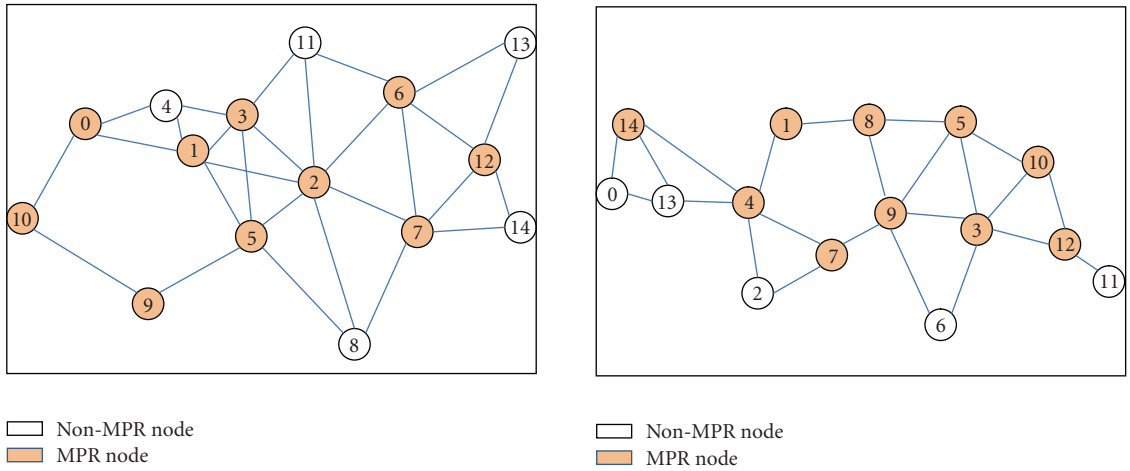
As shown in Figure 2, increasing or decreasing the value of T_w (*weight adjustment period*) too much have a negative effect on the overall performance of the network. If T_w is set to be too large, UBS cannot respond to the changes in nodes' demand for time slots in a timely manner, trying to enforce stale weight values. On the other hand, if T_w is too small (UBS) reacts to the queue state changes too hastily, resulting in high fluctuations in node weights. This again achieves inferior performance.

The other parameter affecting the overall performance of the network is the *increase coefficient* (γ), which changes the range of increment values used in the additive increase method given by (1). The range of allowed step sizes should be large enough so that nodes' demands for time slots can be differentiated. However, if a too large range of step sizes is allowed, the weights of nodes increase very rapidly and the convergence of the weights is delayed, leading to a lower performance.

In Figure 2, the number of dropped packets is smaller along the line *weight-adjustment period* = 10. Likewise, when the *increase coefficient* = 2, the number of dropped packets tends to get smaller compared to the other values of the *increase coefficient*. Considering the results presented in Figure 2, we choose $\gamma = 2$ and $T_w = 10$ in the set of experiments presented in Sections 4.3 and 4.4.

4.2. Effects of Network Topology: Case Study. Network topology is another factor that also has a considerable effect on the obtained performance results. In this section, through a case study, we provide a general discussion on the discussed protocols' behaviors in different network topologies and how the topology can affect the overall network performance.

Figures 3(a) and 3(b) provide examples of two randomly generated 15-node connected networks for which the obtained performance results tend to be quite different. We name the topology in Figure 3(a) as Topology-1 (T-1) and the topology depicted in Figure 3(b) as Topology-2 (T-2). In OLSR-enabled networks, multipoint relay (MPR) nodes are the only nodes that forward data for the other nodes. In other words, not all nodes flood the network with link state updates but only a set of selected special nodes, called MPR nodes,



(a) Topology-1 (b) Topology-2

Figure 3: Sample 15-node network topologies.

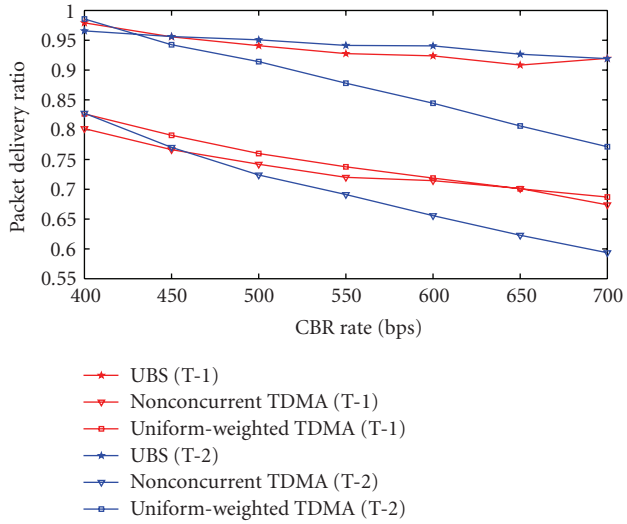


Figure 4: Packet-delivery ratio on two different network topologies (T-1 and T-2).

disseminate link state information and data. In Figures 3(a) and 3(b), we present two different topologies with their MPR nodes marked.

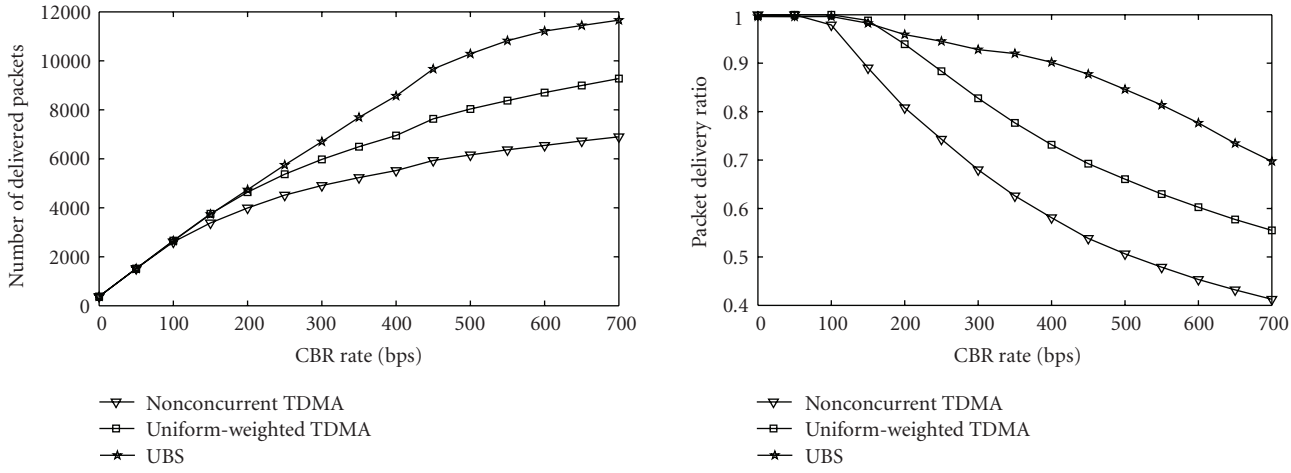
The performance difference between these two networks (see Figure 4) stems not only from the placement of nodes in the network and but also from the routes used by the individual packets. In Topology-1, *node-2* is a bottleneck node. Due to its position in the topology, no other node (except *node-10*) can transmit while *node-2* is transmitting because *node-2* involves all other nodes except no *node-10* in its 2-hop neighborhood and its 2-hop neighborhood is very large compared to the other nodes in the network. This situation also constitutes the reasons for *node-2*'s being chosen as MPR by many nodes to forward their data. Therefore, *node-2* is along the routes of many connections, usually carrying a heavy traffic. Since its traffic load is too much, UBS determines that *node-2* requires more slots almost every time it performs a state check and prioritizes this node's transmissions over other nodes'. In the results for Topology-1, we observe that the performance of nonconcurrent TDMA and uniform-weighted TDMA are very close, as the possibility of concurrent transmissions is very low.

In Topology-2, no node is the only bottleneck node as *node-2* in Topology-1. Therefore, the possibility of performing concurrent transmissions is much higher. However, the lengths of the paths between the two ends of the network are larger compared to Topology-1. Therefore, the performance of nonconcurrent TDMA is severely affected.

Considering the results on both topologies, we observe that the performance of uniform-weighted TDMA degrades significantly in Topology-1 while the performance of nonconcurrent TDMA degrades in Topology-2. However, in both topologies, the performance of UBS remains almost constant. Therefore, we conclude that UBS is highly adaptive to different topologies.

4.3. Uniform Traffic Simulation Results. The three protocols (UBS, uniform-weighted TDMA, and nonconcurrent TDMA) are simulated and compared under the same uniform traffic scenarios, where every node generates a connection to every other node in the network. In each of these simulation scenarios, there are $O(N^2)$ connections that start and end at the same time, where N is the number of nodes in the network. The packet generation rate (in bps) and the packet size (200 bytes) are kept the same for all connections in a single scenario, and a wide range of different CBR traffic rates is applied over different simulations.

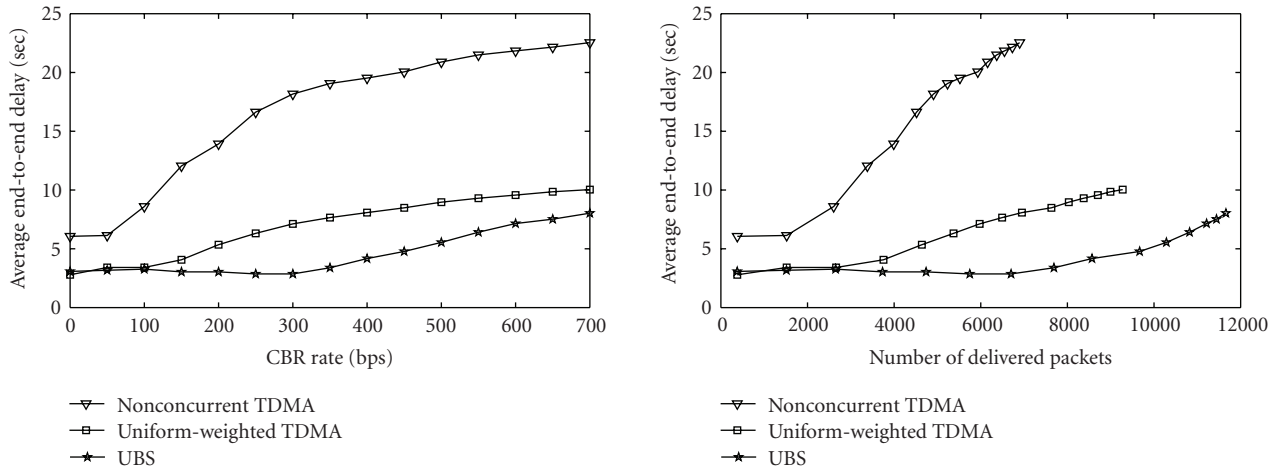
Simulation results with uniform traffic patterns are presented in Figures 5 and 6. Figure 5(a) illustrates the relationship between the packet generation rate and the total number of packets received at the application layer. The results indicate that concurrent protocols perform significantly better than the nonconcurrent TDMA protocol and



(a) Number of delivered packets versus CBR rate

(b) Packet-delivery ratio versus CBR rate

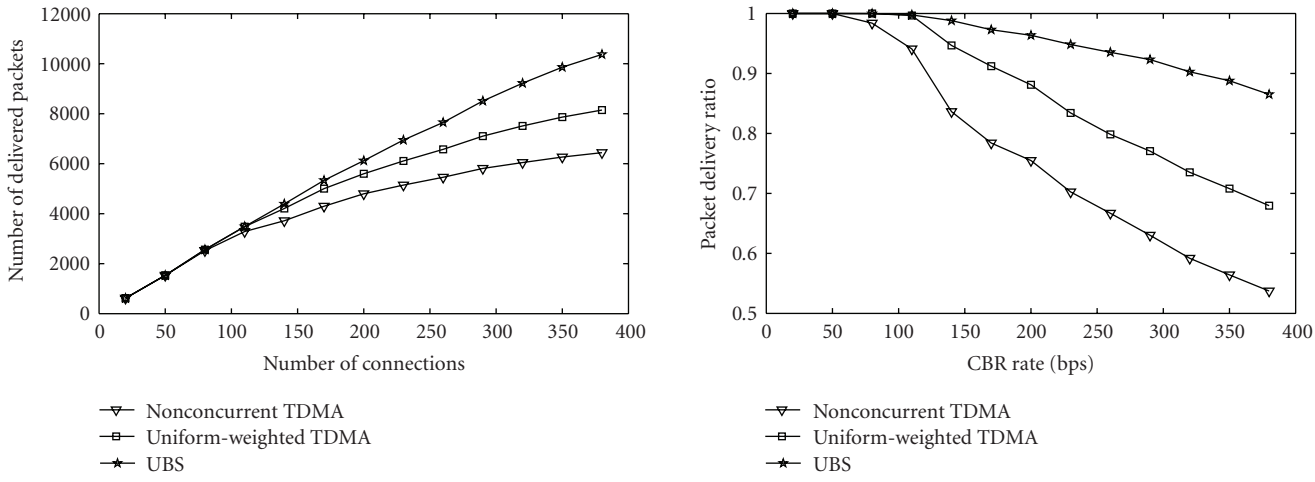
Figure 5: Simulation results with uniform traffic scenarios averaged over 10 different 20-node network topologies.



(a) Average end-to-end delay versus CBR rate

(b) Average end-to-end delay versus number of delivered packets

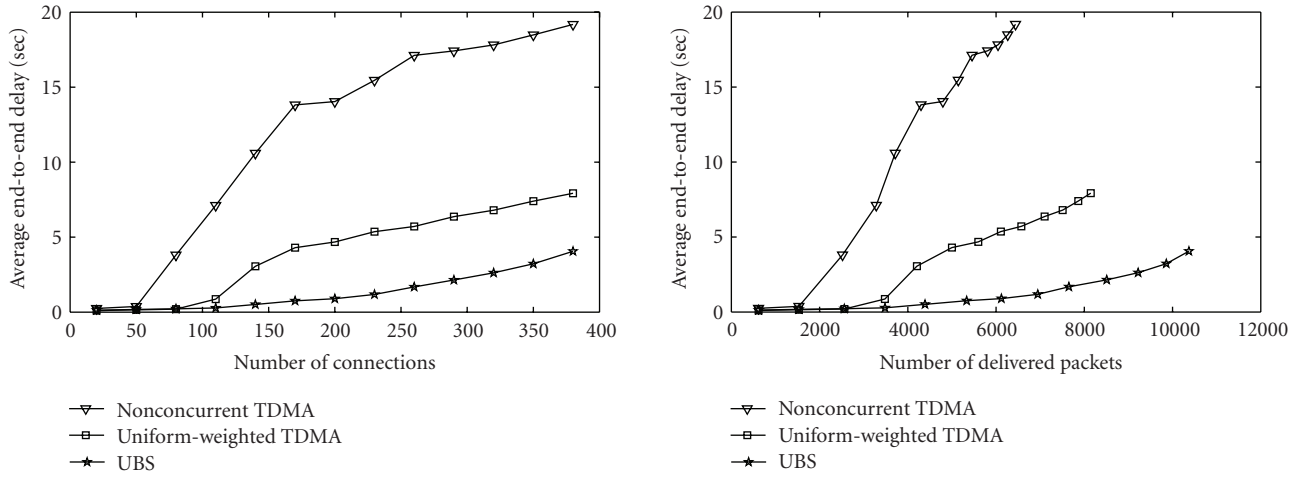
Figure 6: End-to-end packet delays in uniform traffic scenarios averaged over 10 different 20-node network topologies.



(a) Number of delivered packets versus CBR rate

(b) Packet-delivery ratio versus CBR rate

Figure 7: Simulation results with nonuniform traffic scenarios averaged over 10 different 20-node network topologies.



(a) Average end-to-end delay versus CBR rate

(b) Average end-to-end delay versus number of delivered packets

Figure 8: End-to-end packet delays in nonuniform traffic scenarios averaged over 10 different 20-node network topologies.

using dynamic weighted transmission scheduling provides additional improvements.

Figure 5(b) provides information about the ratio of the number of packets that are delivered successfully to the packet generation rate. While maintaining their relative performance rankings, the packet delivery ratios of all three protocols tend to decrease as the packet generation rate increases, due to the increasing congestion.

Figures 6(a) and 6(b) show how the average end-to-end delay changes as functions of the rate and the throughput, respectively. The end-to-end delay values are averaged only for the packets that are successfully delivered. In Figure 6(a), considering the performance gap between the nonconcurrent TDMA and the concurrent protocols, the impact of exploiting spatial reuse (i.e., allowing concurrent transmissions in the nonconflicting parts of the network) is clearly visible.

In Figure 6(a), it is also observed that UBS provides additional improvements not only in terms of the number of successfully delivered packets but also in terms of the average end-to-end delay. The delay of UBS remains below that of uniform-weighted TDMA although they tend to get closer as the packet generation rate increases. The reason for the two protocols' delay values getting closer is that UBS is able to deliver more packets while uniform-weighted TDMA has to drop them. The performances difference of the discussed protocols becomes more apparent, especially when the average end-to-end delay values are compared against the throughput, as depicted in Figure 6(b).

4.4. Nonuniform Traffic Simulation Results. In this section, we present our simulation results under a number of different nonuniform traffic scenarios in which some nodes either create or receive more traffic than the others. In this second set of simulations, again CBR traffic is used, and CBR rate is held fixed at 500 bps. The simulations last for 200 seconds and all connections start at sometime between 25th and 50th seconds and end at some time between 125th and 150th seconds. The connection pairs over a network in each single simulation are randomly chosen and the number of active flows in the network is changed between 20 and 380 with a step size of 30.

Figures 7 and 8 present the related results. The results obtained with nonuniform traffic patterns are quite similar to the uniform traffic simulation results and can be interpreted with similar reasoning. In Figure 7, the number of delivered packets and the packet delivery ratio are presented as the functions of CBR packet generation rate while Figure 8 presents how the average end-to-end delay values change with respect to the CBR packet generation rate and the number of successfully delivered packets. It is observed that both in Figures 7 and 8, the performance rankings of the protocols obtained in the simulations with uniform traffic patterns are preserved. As one apparent difference from the uniform traffic simulation results, in Figure 8(a), the difference between the average end-to-end delays of UBS and uniform-weighted TDMA is more clearly observable than in Figure 6(a).

5. Conclusion

This paper presents UBS, a utilization-based distributed dynamic scheduling scheme, where slot assignments are made based on node weights that are dynamically adjusted using packet-queue occupancies and slot utilizations. We compare the performance of UBS scheme with two other channel access schemes: uniform-weighted TDMA and nonconcurrent TDMA. The difference in the performances of UBS and uniform-weighted TDMA illustrates the effect of the UBS's dynamic weighting mechanism while the performance difference between uniform-weighted TDMA and nonconcurrent TDMA schemes shows the effect of exploiting the spatial reuse in multihop networks. Our simulation results further show that UBS adapts well to the dynamic/nonuniform traffic conditions as its performance improvement over uniform-weighted TDMA is

more significant in simulations with nonuniform traffic patterns.

Acknowledgment

This work was done when Miray Kas was an MS student at Bilkent University, Computer Engineering Department.

References

[1] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, 2001.

[2] "Wireless LAN Medium Access Control (MAC) and Phyical Layer (PHY) Specifications," *IEEE Standard*, vol. 802, no. 1, 1997.

[3] N. Abramson, "The ALOHA system: another alternative for computer communications," in *Proceedings of the Fall Joint Computer Conference*, vol. 37 of *AFIPS Conference Proceedings*, pp. 281–285, 1970.

[4] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems," IEEE Std 802.16-2004.

[5] "IEEE 802.11s Task Group, Draft Amendment to Standard for Information Technology—Telecommunications and Information Exchange between Systems—LAN/MAN Specific Requirements—Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment: ESS Mesh Networking, IEEE P802.11s/D1.0," 2006.

[6] H. T. Cheng and W. Zhuang, "Efficient resource allocation in clustered wireless mesh networks," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '07)*, pp. 79–84, ACM, August 2007.

[7] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid MAC for wireless sensor networks," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pp. 90–101, 2005.

[8] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 210–220, July 2001.

[9] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 181–192, November 2003.

[10] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems-Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands," *IEEE Std 802.16e-2005* (*Amendment to IEEE Std 802.16-2004*), 2005.

[11] J. Tao, F. Liu, Z. Zeng, and Z. Lin, "Throughput enhancement in wiMax mesh networks using concurrent transmission," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WCNM '05)*, vol. 2, pp. 871–874, September 2005.

[12] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. J. Haas, "Interference-aware IEEE 802.16 WiMax mesh networks," in *Proceedings of the 61st IEEE Vehicular Technology Conference (VTC '05)*, pp. 3102–3106, June 2005.

[13] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling in IEEE 802.16 mesh networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '06)*, pp. 147–152, April 2006.

[14] C. Cicconetti, A. Erta, E. Mingozzi, and L. Lenzini, "Performance evaluation of the mesh election procedure of IEEE 802.16/WiMAX," in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM '07)*, pp. 323–327, October 2007.

[15] N. Vaidya, A. Dugar, S. Gupta, and P. Bahl, "Distributed fair scheduling in a wireless LAN," *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 616–628, 2005.

[16] J. Ernst and M. Denko, "Fair scheduling with multiple gateways in wireless mesh networks," in *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA '09)*, pp. 106–112, IEEE Computer Society, Bradford, UK, May 2009.

[17] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in wireless sensor networks: distributed edge-coloring revisited," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1122–1134, 2008.

[18] P. Djukic and S. Valaee, "Distributed link scheduling for TDMA mesh networks," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 3823–3828, Glasgow, UK, 2007.

[19] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 28–36, May 2007.

[20] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, p. 324, 2007.

[21] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1846–1859, 2009.

[22] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks—a partitioning approach," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MOBICOM '06)*, pp. 26–37, ACM, September 2006.

[23] M. Macedo, A. Grilo, and M. Nunes, "Distributed latency-energy minimization and interference avoidance in TDMA wireless sensor networks," *Computer Networks*, vol. 53, no. 5, pp. 569–582, 2009.

[24] S. Floyd, "A report on recent developments in TCP congestion control," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 84–90, 2001.

[25] A. Lahanas and V. Tsaoussidis, "Additive increase multiplicative decrease-fast convergence (AIMD-FC)," in *Proceedings of the Networks*, pp. 511–522, Atlanta, Ga, USA, 2002.

[26] D. Rao and K. Reddy, "Simulation of congestion control and avoidance algorithms for TCP/IP networks," White Paper, University of Hyderabad.

[27] W. Yu and H. Ian, "A deterministic distributed TDMA scheduling algorithm for wireless sensor networks," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '07)*, pp. 2759–2762, September 2007.

[28] E. Lloyd, "Broadcast scheduling for TDMA in wireless multihop networks," in *Handbook of Wireless Networks and Mobile Computing*, pp. 347–370, John Wiley & Sons, New York, NY, USA, 2002.

[29] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR): RFC 3626," IETF Internet Draft, 2003, http://www.ietf.org/rfc/rfc3626.txt.