Experimental Analysis of Backpressure Scheduling in IEEE 802.11 Wireless Mesh Networks

Jae-Yong Yoo*, Cigdem Sengul†, Ruben Merz† and JongWon Kim*
* Gwangju Institute of Science and Technology
† Deutsche Telekom Laboratories, Berlin
{jyyoo,jongwon}@nm.gist.ac.kr, {cigdem.sengul,ruben.merz}@telekom.de

***Abstract*—Transmission scheduling to improve the capacity in wireless mesh networks (WMNs) is challenging. One promising candidate solution is backpressure scheduling, which was proved to provide throughput optimality and queue stability in theory. Additionally, several recent practical systems that implement approximations of backpressure scheduling show performance benefits in WMNs based on IEEE 802.11, which is the most widely adopted MAC protocol. However, a detailed analysis of the queue stability is still missing for practical systems. In this paper, we experimentally show that applying backpressure scheduling over IEEE 802.11-based WMNs presents significant queue instability due to the delayed interaction between MAC and network layers. To understand how and when queue instability occurs, we present weighted backpressure scheduling (WBS), which uses two additional weight factors β and γ for the queue lengths of a node and its next-hop node when computing the backpressure value. By using WBS, we experimentally investigate the interactions between the components of backpressure scheduling implementations, i.e., packet scheduling and link scheduling, and show that, usually, $\beta > \gamma$ leads to queue stabilization.**

I. Introduction

IEEE 802.11-based wireless mesh networks (WMNs) are becoming popular since they can be flexibly deployed to expand Internet access coverage. Despite their benefits, WMNs require solving several networking challenges to operate in full capacity [1]. To improve bandwidth utilization, there have been numerous scheduling proposals. One of the most well-known algorithms is *backpressure scheduling* [2], which computes a backpressure value as the queue differential of a node and its next-hop node for every flow, and uses these values to determine the central scheduling sequence. In theory, it has been proved that backpressure scheduling can stabilize the network queues and hence, provide optimum throughput [2]. However, these results are obtained under the assumption of a synchronized time-slotted MAC protocol, and a central controller that computes and distributes a schedule based on the global per-flow queue and link information. Unfortunately, these assumptions are far from the operation of typical IEEE 802.11-based WMNs.

Realizing backpressure scheduling in practical systems requires approximating the centralized solution into a distributed solution that handles packet and link scheduling separately at different layers while achieving coherent operation [3]–[5]. Due to this challenge, there has been limited work on backpressure scheduling in practice. One example, Horizon [4], approximates backpressure scheduling by directly implementing its packet scheduling part. But for link scheduling, it assumes that the underlying IEEE 802.11 MAC informs a node whenever it gets an opportunity to transmit. Applied to multi-path routing, Horizon achieves better fairness and throughput performance compared to pure multi-path routing. Another example is DiffQ [3], which uses IEEE 802.11e [6] for link scheduling. DiffQ shows better fairness among several transport-layer protocols (e.g., TCP). These examples show the potential of approximating backpressure scheduling in practice. However, to the best of our knowledge, experimental analysis has not been made on whether the practically approximated packet and link scheduling can stabilize the queues as the theoretically-proven backpressure scheduling does. In this paper, we show that the delayed interaction between link (i.e., IEEE 802.11 MAC layer) and packet (i.e., network layer) scheduling results in queue instability[1]. This queue instability leads to large delay jitter, and in the worst case can lead to network capacity degradation.

To investigate the effect of different queues (i.e., the node's and the next-hop node's queues) on the instability, we present *weighted backpressure scheduling* (WBS) that extends the parameter space of original backpressure scheduling. WBS introduces two weight factors, β and γ, for the queue differential computation, where β is the weight of the node's own queue length and γ is the weight of the next-hop node's queue length. By using WBS, it becomes trivial to represent the original backpressure scheduling (i.e., $\beta = \gamma$) as well as other schedulings, for instance, [5], [7]. Based on extensive experiments in the BOWL indoor WMN testbed[2] [8], we show that the queue instability typically occurs when $\beta < \gamma$. Also, in the case of $\beta = \gamma$, we identify the link conditions under which the queue instability occurs (e.g., when the link quality of next-hop node is worse than the link quality of previous-hop node). Thus, to avoid the queue instability, our results suggest using $\beta > \gamma$, i.e., giving a higher weight to a node's own queue size.

The rest of this paper is organized as follows. In Section II, we explain the basics of theoretical backpressure scheduling. In Section III, we highlight the related work and present the

[1] Adding more resources (i.e., increasing the buffer space) can be considered as a possible solution to the problem but as long as the burst size cannot be bounded it would be hard to guess how much resource is necessary to add to each node.

[2] http://www.bowl.tu-berlin.de/menue/home/.

motivation for this paper, especially the challenges emerging from interactions among backpressure scheduling components in a practical system. In Section IV, we present WBS and describe our system model. In Section V, we present our experimental study on the queue stability when using WBS. Finally, we conclude in Section VI.

II. Backpressure Scheduling

This section briefly describes how backpressure scheduling works in theory [2]. In backpressure scheduling, each node i maintains a separate queue b_i^f for each flow f. Assuming a slotted MAC protocol, at each slot, a set of links are scheduled to transmit concurrently by a central controller. To this end, for each link $e = (i, j)$, the central controller chooses the flow that maximizes the queue differential $b_i^f - b_j^f$ of nodes i and j. We denote the backpressure value, D_e^f, for a given flow f and link e as:

$$D_e^f = b_i^f - b_j^f. \tag{1}$$

We denote F_e as a set of flows that pass link e. Then, the maximum link weight among flows F_e on link e is:

$$W_e = \max_{f \in F_e} D_e^f. \tag{2}$$

The capacity μ_e of each wireless link e is the maximum transmit rate at the current channel state. Each transmission set of links defines a network capacity vector $c = (\mu_e)$ and the collection of these vectors defines the network capacity region S. Backpressure scheduling selects the link capacity vector c^* that satisfies [9]:

$$c^* = \arg\max_{c \in S} \sum_e \mu_e W_e. \tag{3}$$

For each link $e = (i, j)$, a transmission rate of μ_e is offered to flow f. The backpressure scheduling algorithm was proved to be throughput optimal as flows send with rates from the capacity region without allowing any queue to be unstable [2]. Although the queue stability of backpressure scheduling is proved in theory, the detailed queue stability analysis of backpressure scheduling in practical IEEE 802.11 WMNs is still required due to the differences between the backpressure scheduling in theory and in practice (e.g., in theory synchronized time-slotted MAC layer is assumed while practical networks use IEEE 802.11).

III. Practical Challenges of Backpressure Scheduling

To realize backpressure scheduling in a practical system, two sub-problems must be solved: *packet scheduling*, which is finding a flow among the set of flows that corresponds to Eq. (2) and *link scheduling*, which is finding a set of links that corresponds to Eq. (3). The two sub-problems fall into the different layers of networking stack (i.e., network and MAC layers, respectively) and hence, need to be solved independently, which is one of the main issues of realizing backpressure scheduling in practice.

One recent example, Horizon [4], implements multi-path routing using backpressure-based heuristics in IEEE 802.11 WMNs to achieve better fairness and load balancing. However, Horizon uses pure IEEE 802.11 and does not perform link scheduling based on the backpressure values. DiffQ [3], on the other hand, handles packet scheduling similarly. But it performs approximated link scheduling using IEEE 802.11e EDCA (Enhanced Distributed Channel Access) and assigns high priority to the links with high W_e values. From this combination of the packet and the approximated link scheduling, DiffQ shows better fairness results than variants of wireless TCP and UDP-based protocols.

These existing implementations make several relaxations to realize backpressure scheduling in practical systems. They rely on IEEE 802.11 and let IEEE 802.11 spontaneously schedule links, rather than using a centralized link scheduling on a synchronized time-slotted MAC layer. This relaxation hinders solution optimality but enables backpressure scheduling to be realized in practice. The approximations using IEEE 802.11e obtains additional benefits from utilizing the priority queues supported by IEEE 802.11e. In IEEE 802.11e, these priority queues prioritize access to the channel depending on traffic classes (e.g., video and audio). Hence, taking advantage of these priority queues allows better control on the link scheduling. However, this is still a rough approximation as IEEE 802.11e provides 4 priority queues and, therefore, W_e values need to be quantized to decide which queue to place the packets coming from the network layer. Furthermore, priority queues in IEEE 802.11e are typically implemented in hardware, as the actions involving them require microsecond granularity. Even the most well-known driver in the research community, MadWiFi [10], does not provide refined control over these priority queues. In these circumstances, the only way to implement backpressure scheduling is to build per-flow queues on top of IEEE 802.11e. We refer to this implementation inevitability as 2-*stage queuing* and categorize the resulting issues as follows:

- *Delayed Link Schedule Activation:* When the backpressure algorithm schedules links, it expects that the scheduled links are activated immediately. However, the actual link activation is done at the physical layer after the backoff of IEEE 802.11 and queuing delay of the priority queue. Moreover, monitoring the queue length of neighbor nodes incurs additional delay in computing the backpressure values. Note that, to monitor flow queue length, the neighbor nodes piggyback the flow queue length to the IP header (e.g., FO - Fragmentation Offset - field is used in our implementation). Then, by promiscuously listening packets, the node learns the flow queue length from the neighbor nodes.
- *Coarse Link Scheduling:* As the number of priority queues is limited, the link weights W_e have to be quantized, which consequently leads to coarse link scheduling. This coarse link scheduling further causes the following *drift* problem. Note that the per-flow queue evolution

at nodes depends on packet arrival and service rates, and becomes stationary if the packet arrival and the service rates are equal on average. However, in wireless networks, the packet arrival and the service rates depend on current network conditions such as link quality and the number of neighboring nodes in the contention region. To stabilize the queue evolution, the backpressure algorithm schedules different priority queues to balance the queue lengths. However, due to the coarse link scheduling, the arrival and service rates might differ, which we refer to as drift. We refer to *up-drift* if the queue length is increasing, and *down-drift* if the queue length is decreasing.

In current research, the impact of these issues rising in practical systems are not very well known, which serves as our main motivation. Prior implementations focus on comparing throughput and fairness and do not focus on the detailed queuing dynamics. The queuing dynamics, however, is an important property for the throughput maximality that backpressure scheduling originally pursues. To thoroughly analyze the resulting queuing dynamics, we expand the parameter space of backpressure scheduling and perform various experiments in a testbed, which are presented in the rest of the paper.

IV. Weighted Backpressure Scheduling

WBS extends the parameter space of the original backpressure scheduling algorithm so that we can analyze the interactions of backpressure scheduling components at various angles. Recall that $D^f_{e=(i,j)}$ is the queue differential between node i and its next hop node j, for flow f (see Eq. (1)). WBS extends this by using a weighted version of these two parameters and calculates a weighted queue differential, WD^f_e, as:

$$WD^f_e = (\beta \cdot b^f_i - \gamma \cdot b^f_j). \quad (4)$$

In Eq. (4), $\beta \in [0, \infty)$ and $\gamma \in [0, \infty)$ are importance factors, where β is the importance of node i's queue length and γ is the importance of the queue length of next-hop, node j. The advantages of WBS are twofold:

- *Understanding the effects of 2-stage queuing on backpressure scheduling*: By choosing $\beta \neq \gamma$, we can see the impact of giving higher importance to different queues, which is not possible with the original backpressure scheduling. This differentiation of the importance factors allows us to understand the impact of 2-stage queuing on backpressure scheduling. Furthermore, we can figure out the relationship between β and γ that results in better queue stability.
- *The ability to present various different scheduling policies*: By choosing the values of β and γ appropriately (such as $\beta = 0$ and $\gamma = 1$, or $\beta = 1$ and $\gamma = 0$), WBS can represent the recently proposed new scheduling policies [5] and [7], respectively. Also, $\gamma = 0$ and $\beta = 0$ represents no scheduling that represents pure IEEE 802.11. In Section V, we discuss how these different policies compare to each other.

V. Experimental Analysis of Backpressure Scheduling with WBS

The goal of our evaluation is to investigate the queue stability of backpressure scheduling in practice. Our results indeed show that queue instability occurs and mainly results from the drift effects and delayed link schedule activation: due to coarsely quantized backpressure values, we observe drifts. In turn, packets get queued at different priority queues at the MAC layer (typically two adjacent queue priorities are used). Also, link scheduling activation is not immediate due to, for instance, backoffs. This combination of drifts and delayed link scheduling results in queue fluctuations and leads to high jitter.

A. Experimentation Setup

We perform experiments on a subset of the BOWL indoor testbed using a line topology [8] - we report results mainly from experiments using a 2-hop path: Node 1-Node 2-Node 3. We observe similar behavior with longer hops, but we present only 2-hop results to illustrate the details of our setting and the results more clearly.

Each node is based on an Avila GW2348-4 board from Gateworks equipped with two Atheros-chipset WiFi cards (Wistron CM9 with the AR5213 chipset). The modulation rate is 6 Mbps with transmit power set to 18 dBm. The maximum MAC-layer retransmission count is set to 8 and RTS/CTS is off. Especially, by setting the modulation rate to the most stable rate of 6 Mbps, we try to equalize the link capacity μ_e for all links. Even though the modulation rate is 6 Mbps, we observe an up-drift in the path from Node 1 to Node 3 and a down-drift in the path from Node 3 to Node 1.

We implement packet scheduling using Click 1.6.0 [11] and link scheduling using MadWiFi 0.9.4 [10], respectively. We set the maximum per-flow queue length to 250, which is large enough to see the detailed backpressure scheduling behavior. We also experimented with different maximum per-flow queue lengths and saw that they do not affect the overall behavior. We limit the queue lengths at the MAC layer to 10 packets, which is the minimum empirical value that avoids system performance degradation due to too small priority queue sizes. We modified MadWiFi to use 8 priority queues [3]. To quantize WD to these 8 priority queues, we use a linear mapping. To accurately monitor the queue length of the MadWiFi priority queues and Click per-flow queues, we port PaPMo (Packet-accurate Protocol Monitor) [12] to MadWiFi and Click. In the remainder of the section, we explore the queuing dynamics and its effect on jitter and throughput.

B. Pure Backpressure Scheduling Case ($\beta = 1$ and $\gamma = 1$)

In our experiments, we generate 5 Mbps UDP traffic from Node 1 to Node 3, which is enough to saturate the path. By observing the intermediate node queue length (Node 2), we see that it fluctuates between 90 to 120 regularly (see Fig. 1). This is an unexpected result as our scenario is a very simple multi-hop scenario with very stable link conditions (with mild up-drift). To understand the underlying reasons, we plot the *priority trajectory (PT)* of a node. The priority trajectory at a

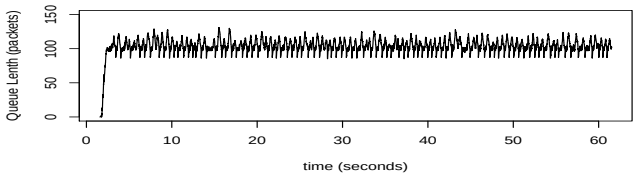

Fig. 1. Queue length evolution with backpressure scheduling in the 2-hop path. Node 2's queue fluctuates from 90 to 120.

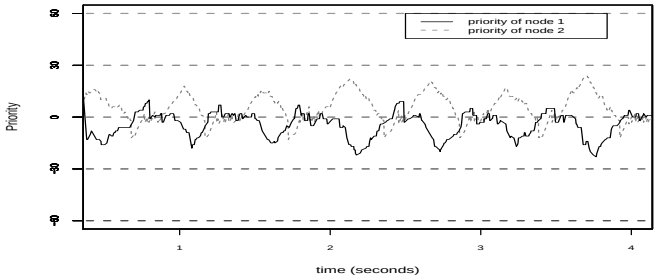

Fig. 2. Priority trajectories of Node 1 and Node 2 up to 4 seconds for original backpressure ($\beta = 1$ and $\gamma = 1$).

node shows, how the backpressure values of Eq. (4), which are quantized at the MAC layer ("priority" values on the y-axis and the quantization is shown with straight dotted lines in Fig. 2), evolve over time. By looking at the PT of Node 1 and Node 2 in Fig. 2, we find two reasons for fluctuation: an increase in priority due to up-drift and a decrease due to delayed link schedule activation (discussed in Section III). For instance, before $t = 2.2$ s, the fluctuation comes from the up-drift. Specifically, around $t = 2$ s, Node 1 and Node 2 share the similar priority, and then suddenly the priority of Node 2 increases (i.e., the queue length of Node 2 increases). However, the information about the increase in Node 2's queue length is delivered to Node 1 with some delay. Hence, the priority of Node 1 decreases with the delay around 0.1 s. Essentially, due to this delay, Node 1 and Node 2 reside in different quantized priorities (different regions divided by straight dotted lines). Due to these two effects, the queue length of Node 2 keeps fluctuating. This leads to a high packet inter-arrival variance.

C. The Effect of β and γ

Observing the fluctuations with backpressure scheduling, we next explore whether setting β and γ different than 1 has an effect on performance. As an example, we show the case where $\beta = 1.0$ and $\gamma = 0.2$. We observe that with these parameter settings, the fluctuations in Node 2's queue length

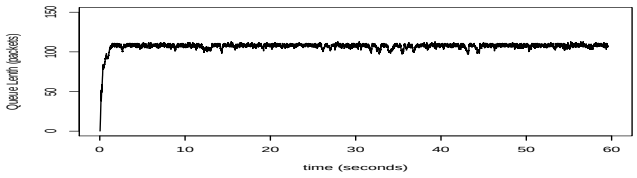

Fig. 3. Queue length evolution of Node 1 in 2-hop case over 60 seconds ($\beta = 1$ and $\gamma = 0.2$).

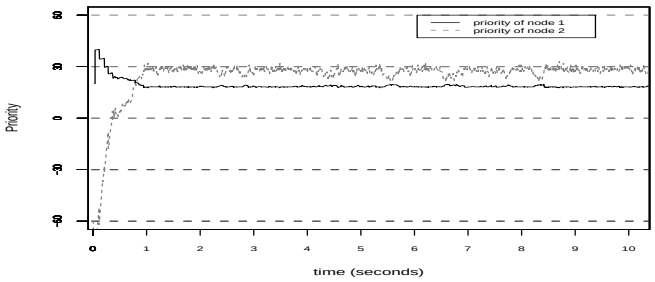

Fig. 4. Quantized priority space with two trajectories of Node 1 and 2 up to 10 seconds ($\beta = 1$ and $\gamma = 0.2$).

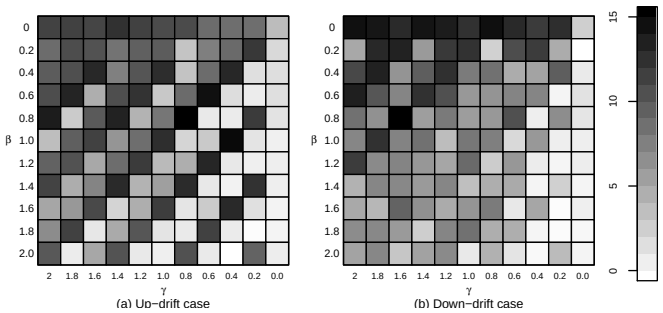

Fig. 5. Comparison of queue length standard deviation at middle node in 2-hop case. We vary β and γ from 0 to 2.0 with the step size of 0.2.

can be avoided (see Fig. 3). The corresponding PT is plotted in Fig. 4. In this case, Node 2 controls its priority based on its own queue length (the destination queue length is constant based on the assumption of backpressure scheduling [2]) and there is no delayed information. Hence, using a higher β value, Node 2 starts reacting to changes fast and aggressively in its queue size.

Next, we present the overall results of varying β and γ from 0 to 2 with a step size of 0.2. For each value of β and γ, we present an average of 5 experiments, where each experiment lasts 60 seconds. To capture the overall queue fluctuation, we plot the standard deviation of each β and γ pairs by using a grid intensity plot (see Fig. 5) where dark and white colors indicate high and low deviations, respectively. Generally, we observe that $\beta > \gamma$ shows better deviation results as a high γ value means slow but aggressive reaction at the senders, which consequently leads to fluctuations in sending rate. Also, with high γ values, the error in neighbor queue length information due to monitoring delay is amplified. Similar trend of "$\beta > \gamma$ shows less fluctuation" also holds as we increase the hop count.

However, even with $\beta > \gamma$, due to the quantization of backpressure values, in some cases a node might switch to using another priority queue faster than its next hop node. Due to delays in monitoring queue lengths and link scheduling, this triggers fluctuations in the queue length, which explains the dark points in some cases for $\beta > \gamma$ in Fig. 5. Therefore, depending on the current network conditions, it is desirable to adapt β and γ parameters to avoid queue fluctuations.

Finally, when there is no backpressure scheduling ($\beta = 0$

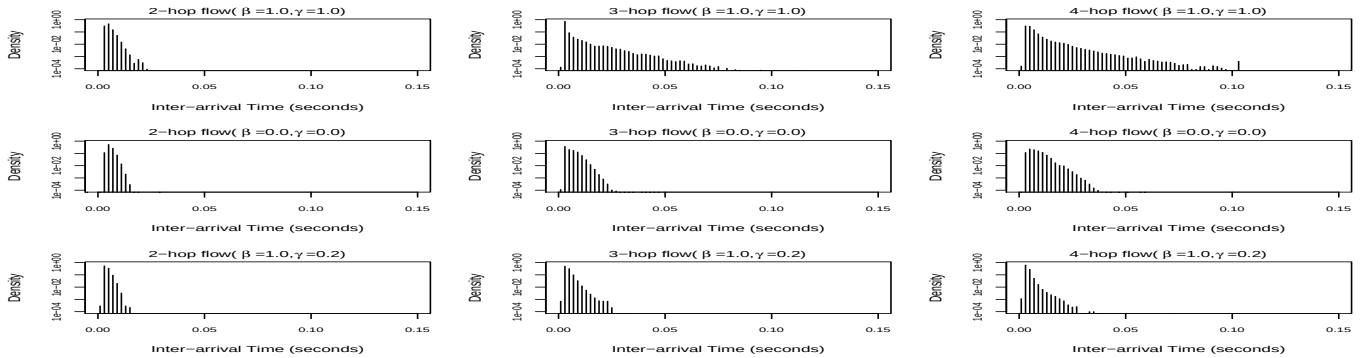

Fig. 6. Density of packet inter-arrival time at the destination queue. Comparison is made among (from the top) pure backpressure scheduling, no scheduling, and WBS ($\beta = 1$ and $\gamma = 0.2$) traffic over 2, 3 and 4-hop path.

TABLE I
2-HOP THROUGHPUT RESULTS OF WBS ($\beta = 1.0$, $\gamma = 0.2$) AND UDP ($\beta = 0$, $\gamma = 0$).

	avg. throughput (Mbps)	avg. packet loss per sec
Up-drift UDP	2.24	46.47
Down-drift UDP	2.50	5.99
Up-drift WBS	2.52	0.02
Down-drift WBS	2.52	0.00

and $\gamma = 0$), there is relatively low fluctuation but the average queue length is not controllable. For instance, in the up-drift case, the queue over-runs and in the down-drift case the queue under-runs, which consequently leads to network under-utilization. By letting $\beta = 0$ and $\gamma > 0$ or $\beta > 0$ and $\gamma = 0$, we can observe the results of scheduling policies that only consider the queue of the node itself [7] or the queue of next-hop node [5]. In our results, scheduling policies that only consider the queue of the node itself ($\beta > 0$ and $\gamma = 0$) show better queue stability in both up and down drift cases.

D. Performance Effects of Queue Instability

In this section, we present how the queue instability affects performance in terms of jitter and throughput. Fig. 6 shows that one of the outcomes of the queue instability is the high packet arrival time variance (i.e., jitter). While the results for backpressure scheduling are unacceptable [13], by using $\beta = 1$ and $\gamma = 0.2$, it is possible to decrease the inter-arrival distribution range by a factor of 2 compared to pure backpressure scheduling. If we compare this result to UDP case without scheduling, we observe that the inter-arrival distribution is more regular due to less fluctuations in the queues. Finally, the throughput results of UDP ($\beta = 0$ and $\gamma = 0$) and WBS with $\beta = 1$ and $\gamma = 0.2$ over 2-hop path is shown in Table I. When there is up-drift, UDP is not able to control the queue of Node 2, and packet drops occur due to queue over-run. These packet drops lead to bandwidth waste and low throughput. On the other hand, in the down-drift case, UDP just suffers from queue under-run and no packet drops occur. Thus, for all cases, the resulting throughput is similar.

VI. CONCLUSION

We show that backpressure scheduling can lead to the queue instability when implemented in IEEE 802.11 WMNs. This behavior mainly stems from the different interactions between packet and link scheduling components of backpressure scheduling implementations. To analyze the effects further we expand the parameter space of the backpressure scheduling with two new weights, β and γ. From the analysis, we conclude that $\beta > \gamma$ generally leads to network queue stabilization in a line topology. For future work, we will explore the effect of β and γ with other topologies and design an algorithm to adaptively tune β and γ for optimal operation.

VII. ACKNOWLEDGMENT

This work was partially supported by Mid-career Research Program through NRF grant funded by the MEST (No. 2009-0086222).

REFERENCES

[1] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE Wireless Communications*, vol. 10, no. 5, pp. 8–14, 2003.
[2] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 236–250, Feb. 1995.
[3] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," in *Proc. of IEEE INFOCOM*, Apr. 2009.
[4] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: Balancing TCP over multiple paths in wireless mesh network," in *Proc. of ACM MobiCom*, Sep. 2008.
[5] A. Aziz, S. David, P. Thiran, and E. F. Alaseddine, "EZ-Flow: Removing turbulence in IEEE 802.11 wireless mesh networks without message passing," in *ACM CoNEXT*, Dec. 2009.
[6] *Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE Computer Society Std., 2005.
[7] A. Proutiere, Y. Yi, and M. Chiang, "Throughput of random access without message passing," in *Proc. of IEEE CISS*, March 2008.
[8] M. Al-Bado, A. Feldmann, T. Fischer, T. Huehn, R. Merz, H. Schioeberg, J. S. Zander, C. Sengul, and B. Vahl, "Automated online reconfigurations in an outdoor live wireless mesh networks," Demonstration at ACM SIGCOMM, Aug. 2009.
[9] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.
[10] "Multiband Atheros driver for WiFi," http://madwifi-project.org/.
[11] "Click modular router," http://read.cs.ucla.edu/click/.
[12] J. Y. Yoo, T. Huehn, and J. Kim, "Active capture of wireless traces: Overcome the lack in protocol analysis," in *Proc. of WinTECH Workshop at Mobicom*, Sep. 2009, pp. 41–48.
[13] P. Calya and C. G. Lee, "Characterizing voice and video traffic behavior over the Internet," in *Proc. of ISCIS*, Oct. 2005.