Measurement-Based Short-Term Performance Prediction in Wireless Mesh Networks

Hua Yu Kai Zeng Prasant Mohapatra
Department of Computer Science
University of California, Davis
Email: {huayu, kaizeng, pmohapatra}@ucdavis.edu

***Abstract*—Traditionally, the performance of wireless mesh networks (WMNs) is measured by long-term averaged metrics, such as long-term averaged packet delivery ratio or throughput. However, due to the dynamic nature of the wireless networks, long-term averaged metrics cannot reflect the short-term behaviors of the network. In the meanwhile, the users may require a sustained performance for a certain period of time in many real-time applications. Prediction of network performance of WMNs at the level of individual flows in a small time granularity becomes very important, but is missing in the literature. In this paper, we propose a measurement-based model to predict the short-term performance of both goodput and packet loss for individual flows in WMNs. This model captures the complex dependencies among the different queues in the system, traffic demand, and wireless interference in the network. We developed two tools *Rater* and *CalMedium* to calculate the probabilities of packet loss along all the layers in the protocol stack at each node. Real-world experiments on an indoor mesh network testbed demonstrate that our prediction method can achieve accurate performance prediction under both single-flow and multiple-flow scenarios. We also discuss two application examples of utilizing our prediction model: finding the bottleneck rate of a flow and admission control.**

I. Introduction

Wireless mesh networks (WMNs) are popular for extending last-mile Internet access due to its low cost and easy deployment using off-the-shelf equipments without relying on existing infrastructure. There are many important and popular civilian applications over WMNs [1], such as broadband access in residential communities, disaster relief, remote data access, and intelligent transportation systems, which require reliable, robust and resilient network. However, unlike wired networks, WMNs have to endure open-air shared communication medium, dynamic and time-varying operating environment, and limited resources. Consequently, it is desirable and necessary to develop efficient technologies capable of predicting the network performance for the deployment and maintenance of large-scale practical systems.

Due to the proliferation of real-time applications, such as real-time e-healthcare, VoIP and video conferencing, prediction of network performance of WMNs at the level of individual flows in small time granularities becomes very important, but is missing in the literature. Specific applications may require a sustained performance for a certain period of time. For example, in real-time e-healthcare, the transportation of real-time medical information or monitoring data requires a packet loss rate smaller than 2% at each second for a period of three minutes. However, the commonly used long-term averaged metric cannot reflect the short-term behaviors of the network, due to the dynamic nature of the wireless networks and the varying network traffic condition. By predicting the network performance in a short term, we can make a timely adjustment (such as routing or traffic control) in the network in order to meet the performance requirement of real-time applications.

In this paper, our goal is to enable systematic prediction of network performance of WMNs at the level of individual flows for each short time interval during a period of time. We will mainly focus on predicting the goodput and packet loss. With our work, we will answer the following questions that are not easily solved before: What will be the short-term network performance under the current or a planned traffic load for the network?

We propose a novel measurement-based model-driven approach to predict the short-term network performance for WMNs. We focus on static 802.11-based multi-hop networks, though we believe that the general methodology is applicable to other scenarios. To the best of our knowledge, our work is one of the first to present a systematical model to predict the performance in WMNs at a short time scale.

The cornerstone of our approach is a new model that captures the complex dependency among traffic demand, device's transmission queue, CPU's receiving queue, socket queue, wireless interference, and MAC layer settings in the network. These dependencies contribute to the complexity in evaluating the network performance for a short time period. Accurately capturing them without complicated calculation of the inter-dependency factors is the fundamental challenge in fulfilling our goal. To achieve this, we develop two tools *Rater* and *CalMedium* to calculate the probabilities of packet loss along all the layers in the protocol stack under different experiment settings. The actual packet loss probabilities used in the model under the current network condition is mapped accordingly and plugged into the model to determine the goodput and packet loss at each small time interval.

Our evaluation shows that the predicted results have a good match with those obtained from our real-world mesh network testbed for both single-flow and multiple-flow scenarios. We also demonstrate how our work can help in the admission control decision and in finding the bottleneck rate for a flow. The main contributions of the work include:

- We propose a new concept of characterizing the network performance for WMNs. Instead of giving just a metric measured in a long-term period, we provide time-varying short-term metrics (i.e., a series of the goodput and packet loss values along time) to better reflect the wireless network conditions.
- We propose a general measurement-based model to com-

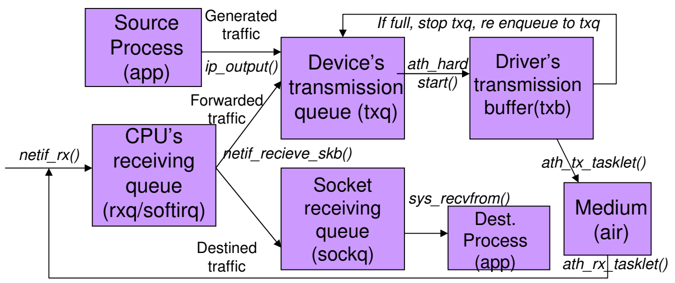


Fig. 1. Queues and related functions

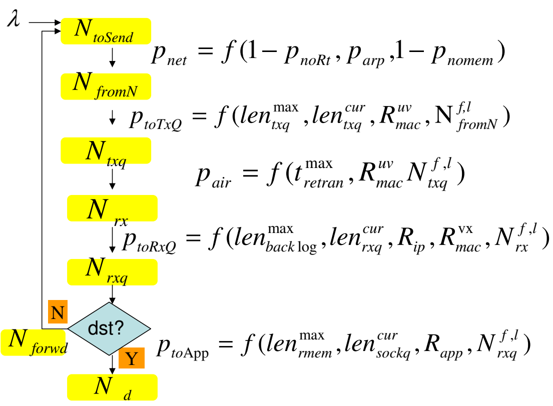

λ
N_{toSend}
$p_{net} = f(1 - p_{noRt}, p_{arp}, 1 - p_{nomem})$
N_{fromN}
$p_{toTxQ} = f(len_{txq}^{\max}, len_{txq}^{cur}, R_{mac}^{uv}, N_{fromN}^{f,l})$
N_{txq}
$p_{air} = f(t_{retran}^{\max}, R_{mac}^{uv} N_{txq}^{f,l})$
N_{rx}
$p_{toRxQ} = f(len_{back\log}^{\max}, len_{rxq}^{cur}, R_{ip}, R_{mac}^{vx}, N_{rx}^{f,l})$
N_{rxq}
N
dst?
Y
N_{forwd}
$p_{toApp} = f(len_{rmem}^{\max}, len_{sockq}^{cur}, R_{app}, N_{rxq}^{f,l})$
N_d

Fig. 2. General model

pute the network performance at a short time scale. The parameters of the model are obtained from a profiling phase under different experiment settings through our developed tool *Rater* and *CalMedium*. The mapping of the parameters according to the current network condition is the key to avoid the complicated calculation and consideration of the inter-dependency among the traffic load, different queue processing speed, MAC layer settings, and wireless interference.

- We conduct real-world experiments to validate our prediction model.

II. Related Works

A large number of measurement based studies have been carried out to study the source of packet loss in indoor large scale 802.11 deployments [2]. Aguayo et al. [3] identified interference as a main source of packet loss for an outdoor 802.11 mesh deployment. Sheth et al. [4] characterized the packet losses for a WiFi-Based Long Distance Network. Most of the experimental works only consider the impact of the wireless link. We focused on evaluating the impact of different factors from both the wireless medium and different queues in the protocol stack on the network performance.

The problems of throughput prediction have been studied thoroughly in the wireline domain. Banerjee et. al [5] propose a model to estimate the optimal rate for the network. They calculate the packet loss probability as a function of the packet-arrival rate, which, however, is difficult to derive in wireless networks. Computing achievable throughput in wireless networks is a widely-studied area. Mirza et al. [6] address the problem of TCP throughput prediction for opportunistic WLAN networks with a focus on file transfer application. Li et. al [7] predict the throughput of individual flows and optimize the network for fairness and throughput. They mainly consider wireless interference in the network in their model. However, they do not consider the packet loss occurring in the queues of the system protocol stack, nor do they consider short time metrics.

III. Measurement Based Model

In this section, we develop a model for deriving a time-based series of network performance values. The model is based on 802.11 DCF MAC. We assume that the probability of node failure is zero and that the flow has CBR traffic with fixed packet size.

A. Approach

We first present a brief background about the socket and MadWifi [8] driver implementation on the mesh node. Assuming UDP is used as the transport layer protocol, the flow of packets through different queues along the protocol stack and the related key functions are shown in Fig. 1. At the transport layer, the message is copied into the socket sender buffer and passed to Function *ip_output* after adding the IP header. At the network layer, a processed IP datagram is put into the device's transmission queue (*txq*) after adding the frame header. The MAC layer calls the driver's transmission function *ath_hardstart* to put the frame into the driver's transmission buffer for sending. On the reception side, a tasklet is triggered by the reception interrupt and Function *ath_rx_tasklet* is called to process the received frame and to hand it over to the 802.11 protocol layer. The frame will be decapsulated and Function *netif_rx* is called to put the packet into the CPU's receiving queue (*rxq/softirq queue*) and to triger a software interrupt to inform the upper IP layer. Function *netif_receive_skb* is called to process the IP packets. The traffic destined to the node is put into the socket receiving queue (*sockq*) waiting for the application to read. Otherwise, the forwarded traffic will be finally passed to Function *ip_output* and then put into the device's transmission queue (*txq*) for transmitting.

We consider an end-to-end flow f_{sdk}, the k^{th} flow from a source node s to a destination node d. At each hop, in a time interval τ, the following factors which may cause packet dropping are considered: 1) no route or ARP (Address Resolution Protocol) error, 2) device's transmission queue (*txq*) overflow, 3) random medium error and interference collision, and 4) CPU's receiving queue (*rxq*) overflow. At the destination node, the packet dropping is mainly due to the socket receiving queue (*sockq*) overflow. This model accurately captures the node-, channel-, and interference-induced packet drops, which are the main causes of the packet losses summarized in our previous work [9]. We use our developed tools to obtain the probabilities of the packets passing through these points, and predict the goodput and packet loss hop by hop for each flow at each time interval.

B. Model Description

Fig. 2 shows the model for the flow of packets through the layers in the protocol stack to determine how many packets can be successfully passed through hops from the source to the destination. The notations are explained as below:

λ	external traffic demand rate
N_{toSend}	pkts # from the application layer or previous hop
p_{net}	probability of the pkt passing to the network layer
N_{fromN}	pkts # sent from the network layer
p_{toTxq}	probability of the pkt passing the *txq* queue
N_{txq}	pkts # accepted at the *txq* queue
p_{air}	probability of the pkt successfully received from air
N_{rx}	pkts # received from the medium
p_{toRxq}	probability of the pkt passing the *rxq* queue
N_{rxq}	pkts # accepted at the *rxq* queue
N_{forwd}	pkts # forwarded to the next hop
p_{toApp}	probability of the pkt accepted by the application
N_d	pkts # received at the application of destination node d

Now we describe the model used to calculate the goodput $g(t)$ at any time t for a considered flow f_{sdk}. The goodput $g(t)$ is defined as the number of bits received during time interval [t-τ, t]. Suppose $N_d^{f_{sdk}}$ packets can be correctly received at the destination in time interval τ. Eqn. (1) shows the calculation of $g(t)$,

$$g(t) = \sum_{p=1}^{N_d^{f_{sdk}}} S_p/\tau, \tag{1}$$

where S_p is the size of each received packet p. Then computing $g(t)$ is converted to calculate $N_d^{f_{sdk}}$ as follows. Extending the notation of N_x mentioned above, we use $N_x^{f_{sdk},l_{uv}}$ to denote the number of packets that belong to the flow f_{sdk} and will be routed over link l_{uv} at the considered point x. In the routing matrix R, $R(f_{sdk}, l_{uv}) = 1$ denotes that packets of f_{sdk} is routed via l_{uv}. $\forall f, l \in \{R(f_{sdk}, l_{uv}) = 1\}$, we have:

$$N_{toSend}^{f_{sdk},l_{uv}} = \lambda^{f_{sdk}}\tau|\{u = s\} + \sum_{w\in R(f_{sdk},l_{wu})=1} N_{forwd}^{f_{sdk},l_{wu}} \tag{2}$$

$$N_{fromN}^{f_{sdk},l_{uv}} = N_{toSend}^{f_{sdk},l_{uv}} p_{net}, \tag{3}$$

$$N_{txq}^{f_{sdk},l_{uv}} = N_{fromN}^{f_{sdk},l_{uv}} p_{toTxq}, \tag{4}$$

$$N_{rx}^{f_{sdk},l_{uv}} = N_{txq}^{f_{sdk},l_{uv}} p_{air}, \tag{5}$$

$$N_{rxq}^{f_{sdk},l_{uv}} = N_{rx}^{f_{sdk},l_{uv}} p_{toRxq}, \tag{6}$$

$$N_{forwd}^{f_{sdk},l_{uv}} = N_{rxq}^{f_{sdk},l_{uv}}, \tag{7}$$

Eqn. (2) indicates that the number of packets $N_{toSend}^{f_{sdk},l_{uv}}$ arrived at node u to be sent to node v, i.e., the traffic of the flow f_{sdk}, is the summation of two components: external traffic generated from node u and the forwarded traffic to node u from the previous hop node w. For a given flow f_{sdk}, one of the components is zero. For single-hop routing, the second component consists of at most one item as there is at most one node w satisfying the routing matrix.

Eqns. (3) to (6) calculate $N_{rxq}^{f_{sdk},l_{uv}}$, the number of packets of flow f_{sdk} received at the *rxq* queue of node v from node u. These packets pass through the network layer, the device's transmission queue of node u, the link between node u and v, and the CPU's receive queue with the corresponding probability p_{net}, p_{toTxq}, p_{air}, and p_{toRxq}. Eqn. (7) says that $N_{rxq}^{f_{sdk},l_{uv}}$ consists of the forwarded traffic at node v for the next hop, which will be used in Eqn. (2). Lastly, we have Eqn. (8) for node $d \in R(f_{sdk}, l_{ud}) = 1$ to get $N_d^{f_{sdk}}$ packets through the achieved $N_{rxq}^{f_{sdk},l_{ud}}$ from the above equations hop by hop:

$$N_d^{f_{sdk}} = N_{rxq}^{f_{sdk},l_{ud}} p_{toApp}. \tag{8}$$

Packet losses are updated at each time interval according to the current queue length and the probabilities calculated. Due to the limited length of the queues and different flow durations, the packet loss can be different in different short time intervals. When multiple flows are considered, we assume the same packet loss rate applies to all the flows passing a specific queue. This assumption is reasonable as the packets from each flow in the queues are usually uniformly interleaved when FIFO (first in, first out) is applied. We do not show the packet loss in the model, but we include the prediction of the short-term packet losses in the algorithm and the evaluation results sections. Fig. 3 is an example topology with the above model for multiple flows, i.e., flow $f1(S_1, A, B, E, D_1)$, $f2(S_2, A, B, C, D_2)$, $f3(A, B)$. For each flow, the line color and style, and the color of notations are labeled differently.

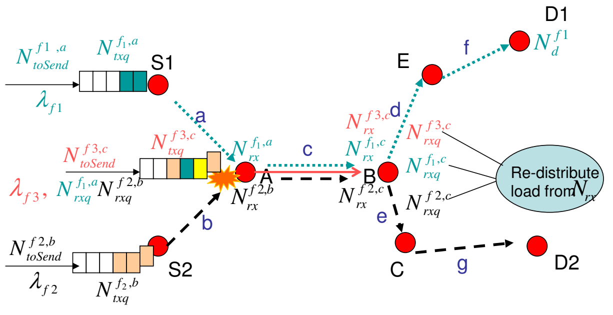


Fig. 3. Example Topology

C. Calculation of Probabilities in the Model

Now we describe the queuing model and how to calculate the probabilities in Eqns. (3) to (8). We model the CPU's receiving queue (*rxq*), socket receiving queue (*sockq*), device's transmission queue (*txq*) and driver's transmission buffer as an M/M[N]/1/K bulk-service finite-capacity queue. The M/M[N]/1/K queue is used for determining the packet-loss probability due to buffer overflow, as a function of the arrival data rate and service rate. The finite capacity is due to the limited size of the queue of buffer.

In practical systems, the arrival and service processes have the bulk feature. Moreover, a single CPU is responsible for both the arrival and service processes. It is hard (if not impossible) to get a closed form solution of the service rate. The probabilities indicated in Fig. 2 depend on the traffic load, queue service rate, MAC layer conditions, and many other factors. It is not easy to get a closed form solution of the probabilities, either. Thus, we take a measurement based approach to collect the corresponding service rates μ which may be suitable in different scenarios for various arrival rates λ. In order to achieve this, we developed a tool, *Rater*, by taking advantage of kprobe [10], a dynamic kernel instrumentation system. *Rater* calculates the number of packets arrived at or removed from a queue within a certain time interval by time-stamping and counting the key functions related to the change of the queue size under current network condition.

Note that in reality the discrete/bulky processing may put the packet into the queue first and then the service process/kernel will remove them later. We make the calculation easier by assuming all the packets in a short time interval have the same behavior. We take an average value for both the arrival rate

TABLE I
SAMPLE RATER OUTPUT

node	λ_{rxq}	μ_{rxq}	λ_{txq}	μ_{txq}	μ_{txb}	λ_{sockq}	μ_d	rate(pkt/s)	size(Byte)	l_{rxq}	l_{txq}	l_{sockq}
n_{src}	0.00	-1.00	1502.78	1503.78	1518.81	0.00	0.00	1500	256	0	1	1
n_{fwd}	1516.80	352.91	350.89	352.91	349.87	0.00	0.00	1500	256	1001	1	2
n_{dst}	369.34	384.29	-1.00	-1.00	14.94	384.29	380.02	1500	256	0	1	0

and service rate of a queue over all the packets during the interval. When the arrival rate λ is smaller than or equal to the service rate μ, no queue is built up and queued packets can be processed and removed. On the other hand, when the arrival rate is greater, a queue will be built up with a speed of $\lambda - \mu$ and then start dropping packets with such speed depending on the current queue length. This packet drop is part of the total packet losses for a flow.

A sample output of *Rater* obtained from a real-world mesh network measurement is shown in Table I. The three lines depict the outputs collected at a source node (n_{src}), a forwarding node (n_{fwd}) and a destination node (n_{dst}). From left to right on each row, the values denote the arrival and service rates at different related queues, the generation rate and packet size of a flow, and the length of each queue at the end of the interval. A non-positive value of the arrival/service rate means that there is no traffic through the corresponding queues and that the rate information at these queues is not needed for this type of nodes.

In the profiling phase, we run *Rater* under different levels of packet generation rate and packet size combination (low, medium and high levels for both parameters) for flows of one to three hops to output a table with a format as Table I. This process considers the complicacy between the system and network such as queue processing, inter-flow interference, traffic load and MAC factors, while this measurement based approach achieves simpleness and accuracy at the same time. Then the probabilities p_{toTxq}, p_{toRxq}, and p_{toApp} can be obtained by looking up such a table to find out the mapped service rate at each corresponding queue according to the current traffic and the number of transmitters within two hops at the same time. If the number of transmitters is more than two, we use the table for flows of three hops, otherwise, we use the table for flows of one or two hops depending on whether there is one or two simultaneous transmitters. From our measurement results, we found very few number of packet losses for static routing when the packets are passing to the network layer. In our simulation, we set p_{net} as 100%. One point that deserves to be mentioned is that there are no packet losses at the transmission buffer because the packet will be re-enqueued into the device's transmission queue if the transmission buffer is temporarily full.

The medium transmission probability p_{air} can be obtained by *CalMedium* tool, which is modified based on our previously developed tool *FlowPac* [9]. After collecting the flow-based statistics at different points in the system through *FlowPac*, *CalMedium* calculates the probability of successful transmission of a link based on the number of packets sent to the medium and received at the receiver by the device driver. For each link $l(s, r)$, we run *CalMedium* and store the results in a table. p_{air} can then be mapped later according to the current network traffic condition. Due to the CSMA/CA, the packet loss directly due to the collision is rare. However, there is random medium loss and the service rate at each queue will be decreased due to the wireless interference, which is reflected in the table from *Rater* output. This approach captures the random medium loss, interference and underlying MAC factors such as transmission rate and retransmission times.

IV. Prediction Algorithm

This section describes our short-term goodput and packet loss prediction algorithm. Alg. 1 is the main algorithm for prediction. It loops for each time interval. At each time interval (*curT*), it checks each link (*curL*) that carries flows to see whether it can be predicted or not (Alg. 2). If it is the case (Line 8), it predicts the different counts on the link. Each loop ends until all the links that carry flows have been predicted (Line 5). The time complexity is proportional to the number of links which carry flows during the prediction time.

Algorithm 1 Main

```
1: procedure PREDICT($T$, $\tau$, $l$, $p$)                ▷ $T$:monitor_time
2:     for $curT \leftarrow 1, floor(T/\tau)$ do
3:         $curL \leftarrow tLinks(curT)$
4:         $p(l) \leftarrow -1, \forall l \in curL$          ▷ $p$: dependency vector
5:         while do $\exists\ l,\ s.t.\ p(l) == -1 || p(l) > 0$
6:             $[y, r, g, fd] \leftarrow$ can_be_predicted$(curT, curL)$;
7:             $p(l) \leftarrow y$              ▷ $y$: number of dependent Links
8:             if $y == 0$ then
9:                 predict_the_link$(curT, curL, g, fd, r)$;
10:            end if
11:            if $p(l)$ is the same as that in the last loop then
12:                estimate_the_rate$(curT, curL)$;
13:            end if
14:        end while
15:    end for
16: end procedure
```

The function *can_be_predicted* checks all the flows passing through the sender node of the link. If there are receiving flows forwarded by or destined to this node (Line 23) and the number of receiving packets at the sender node N_{rx} is not known (Line 24), the dependency (y) is increased by one (Line 25). Only if there is no dependency for the sender node, i.e., a source node or the previous hop has been predicted, this link can be predicted. During this process of examination, the number of packets destined to, forwarded by or generated from this node (r, fd, and g) are recorded to help with the probability calculation (Line 26 to 33).

If two links are waiting for each other to predict, a dead-lock occurs. This would happen when a link has flows with different directions. To break the dead-lock, the procedure *estimate_the_rate* picks up the link with the smallest dependency and estimates the receiving counts for each receiving flow toward the sender node of the selected link. The estimated value is the minimum value between N_{rx} in the previous time interval and the known N_{fromN} value at the nearest sender node of the receiving flow.

Algorithm 2 Link Check

```
17: function CAN_BE_PREDICTED($curT, curL$)
18:     $n_{send} = edges(curL)$                          ▷ sender node
19:     $F_n \leftarrow tNFlows(curT, n_{send})$          ▷ flows passing $n_{send}$
20:     $y \leftarrow 0$
21:     $g, fd, r \leftarrow 0$
22:     for all $f \in F_n$ do
23:         if $f < 0$ then                           ▷ receiving nodes for flow $f$
24:             if $N_{rx}(n_{send}) < 0$ then
25:                 $y \leftarrow y + 1$
26:             else if dst($f$) == $n_{send}$ then
27:                 $r \leftarrow N_{rx}(n_{send}) + r$
28:             else
29:                 $fd \leftarrow N_{rx}(n_{send}) + fd$
30:             end if
31:         else
32:             $g \leftarrow \lambda(f) * \tau + g$
33:         end if
34:     end for
35: end function
```

The procedure *predict_the_link* first calculates all the above-mentioned probabilities by mapping the table with the current traffic on the link. It then updates the packet counts remained after passing each point in the system, the queue length and the packet losses correspondingly. The packet losses include the packet drop at different queues, the network layer, and the medium. When the numbers of packets reaching a queue from different flows are known, the packet drop and queue length can be updated. When the service rate is lower than the arrival rate, there is potential loss at the queue. If it is greater than the current available queue space, there is packet drop. Otherwise, packets can be queued more or be removed from the queue until it reaches the saturated service rate, which is the largest rate without building any queues. This value could be mapped by checking the table output from our *Rater* tool.

V. Simulation And Experimental Results

A. Experimental Topology and Configuration

We perform our measurements on a mesh network testbed consisting of four mesh nodes (Node 1 to Node 4) with one client, the laptop (Node 5) shown in Figure 4. The mesh nodes are situated within a house and form a multi-hop network.

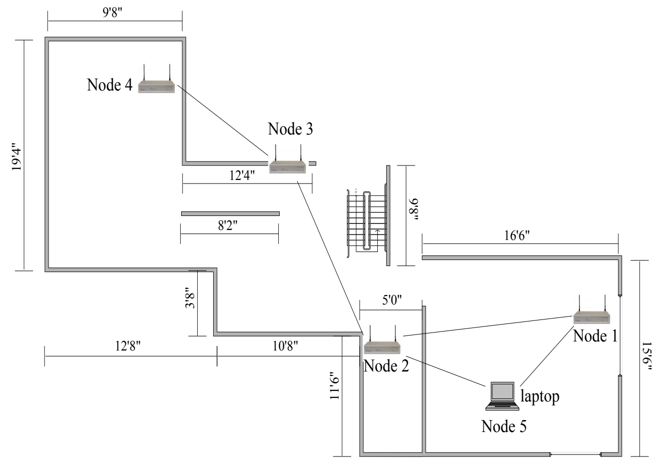
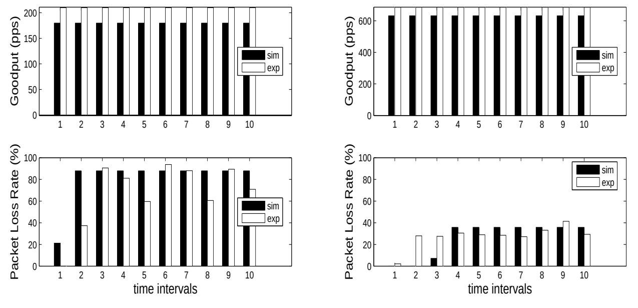


Fig. 4. Experimental Topology

Each mesh node is a 266 MHz x86 Soekris net4826 embedded device running a custom built Linux distribution using a 2.6.23 Linux kernel [11]. The mesh nodes have 128MB SDRAM main memory and 64MB compact flash for the OS and other storage. The client is an HP nc6000 laptop running with a Linux kernel of 2.6.25. We use Atheros 802.11 a/b/g radios in all of our devices. Each of the mesh nodes in our testbed is equipped with two radios. In the following experiments, we only use the ad-hoc mode radio. All the nodes are in the same interference conflict region.



(a) 1500*1280 (b) 1000*256

Fig. 5. Single flow under different Load

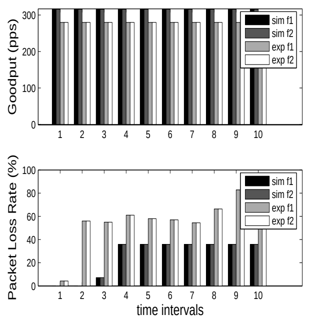
All of our experimental results are based on a 10-second constant bit rate (CBR) UDP traffic stream. The measurements of goodput and packet loss are reported using *ITGDec* command in D-ITG [12] tool. The time interval τ we chose is 1 second considering the trade-off between accuracy and computation complexity.

B. Evaluation for a Single Flow Under Different Loads

We evaluated the performance for a single flow from Node 1 to Node 4 via the route $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. Fig. 5(a) shows the experimental and simulation goodput and packet loss rate of the flow under the traffic demand of 15.36Mbps at a generation rate of 1500 packets per second (pps) with packet size of 1280 bytes. Fig. 5(b) shows the experimental and simulation results under the traffic demand of 2Mbps at a generation rate of 1000 pps with 256-byte packet size. We can see there is a relatively good match between the experimental and simulation results under both traffic demands. Some reasons contributing to the discrepancy include: 1) We assume the current queue length at the start of simulation is zero, while it may not be true in the experiment; 2) The simulation uses average values for each short time interval while the packets are processed discretely. 3) The goodput has a better match than the packet loss because the model is better at reflecting the rate. Our model achieves better results if the random loss of the wireless environment is relatively stable. We may include it as a future work to consider random process to achieve more accurate values. Fig. 5 also shows that the packet loss varies with the time while the goodput is relatively stable.

C. Evaluation for Multiple Flows Under Different Loads

Fig. 6(a) evaluates two identical flows with the same source and destination nodes, Node 1 to Node 4. Both flows have the same generation rate of 500 pps with 256-byte packet size and last for the same duration. In Fig. 6(b), Flow 1 ($f1$) starts at the $8th$ time interval with a generation rate of 499 pps and 214-byte packet size from Node 4 to Node 3, and Flow 2 ($f2$) starts from the beginning at a generation rate of 727 pps with


(a) flows with same src and dst nodes

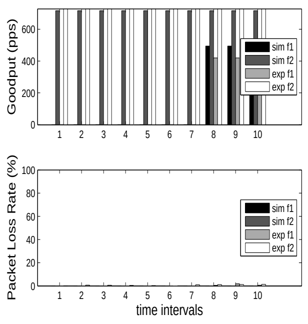
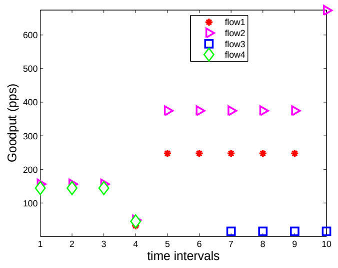

(b) flows with different src and dst nodes

Fig. 6. Multiple flows

TABLE II
MULTIPLE FLOWS

$flow$	T_{start}	T_{fin}	$rate$	size	s	d	route
1	3	9	452	44	5	1	[5 2 1]
2	0	10	684	93	1	3	[1 2 3]
3	6	12	16	17	2	1	[2 1]
4	0	4	632	718	5	3	[5 2 3]
5	4	7	154	676	4	1	[4 3 2 1]


(a) Four flows

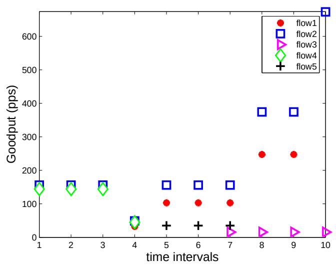

(b) Five flows

Fig. 7. Example of Admission Control: goodput

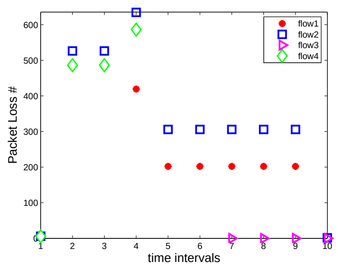
412-byte packet size from Node 3 to Node 2. We also found a good match for both scenarios. For Fig. 6(b), our prediction of packet loss is zero, however, in reality, a few packets may be lost due to the dynamic wireless environment.

D. Two applications

Now we give two application examples that our prediction tools can be used for.

1) Finding the Bottleneck Rate: We define the bottleneck rate as the largest rate at which a flow can be sent without incurring any packet loss or queuing. With our tools *Rater* and *CalMedium*, we applied a method similar to the binary search to find out the bottleneck rate. Under our experimental topology, we found that 700 pps with 1280-byte packet size for a three-hop flow can start to saturate the network.

2) Admission Control: Fig. 7(a) and Fig. 8(a) show the goodput and packet loss rate with four flows (Flows 1 to 4 described in Table II) in the experimental network. For example, Flow 1, from Node 5 to 1, has a duration of 6 seconds from start time 3 to end time 9 at a generation rate of 452 pps with 44-byte packet size, taking the route $5 \rightarrow 2 \rightarrow 1$. As shown in Fig. 7(b) and Fig. 8(b), the goodput decreases and the packet loss increases for the Flows 1 and 2 when Flow 5 starts at time 4 during the 5-th interval. The reverse happens when Flow 5 stops at time 7. At time interval 10, the goodput of Flow 2 increases a lot when Flow 1 stops at time 9. We


(a) Four flows

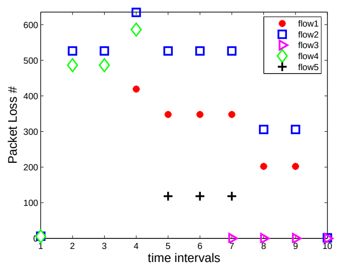

(b) Five flows

Fig. 8. Example of Admission Control: packet loss

can make the admission control decision according to the time-based performance (the goodput and packet loss). For example, if the degradation of the goodput or packet loss of the existing flows (e.g., Flows 1 and 2) is not within the acceptable range if Flow 5 comes in, Flow 5 will not be admitted.

VI. CONCLUSION

Prediction of network performance of WMNs at the level of individual flows in a small time granularity becomes very important due to the proliferation of various real-time applications. The short-term goodput and packet loss can be a good reflection of the network performance. In this work, we proposed a model to predict the short-term performance for a certain user's flow in WMNs. Our model achieves both the accuracy and simplicity based on the measurement results given by our tools *Rater* and *CalMedium*. It also avoids the complexity of the inter-dependency of interference and MAC-induced factors. Our predicted results have a good match with those obtained from our real-world mesh network testbed for both single-flow and multiple-flow scenarios. For the future work, we would like to consider different traffic patterns with varying packet sizes for each flow. Node failure probability can be another factor we may consider in the model. We would also like to include latency as a metric in the prediction of performance.

REFERENCES

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, March 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4F53V5H-2/2/9fa1587e47665f1fb3f7fb461461dd6b

[2] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, "Measurement-based characterization of 802.11 in a hotspot setting," in *E-WIND*, 2005.

[3] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *SIGCOMM*, 2004.

[4] A. Sheth, S. Nedevschi, R. Patra, S. Surana, E. Brewer, and L. Subramanian, "Packet Loss Characterization in WiFi-Based Long Distance Networks," in *InfoCom*, 2007.

[5] A. Banerjee, B. Mukherjee, and D. Ghosal, "Modeling and analysis to estimate the endsystem performance bottleneck rate for high-speed data transfer," in *PFLDnet*, 2007.

[6] M. Mirza, K. Springborn, S. Banerjee, P. Barford, M. Blodgett, and X. Zhu, "On the accuracy of TCP throughput prediction for opportunistic wireless networks," in *Secon*, 2009.

[7] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner, "Predictable performance optimization for wireless networks," *SIGCOMM*, vol. 38, no. 4, 2008.

[8] "Madwifi: multiband atheros driver for wifi," http://www.madwifi.org/.

[9] H. Yu, D. Wu, and P. Mohapatra, "Experimental anatomy of packet losses in wireless mesh networks," in *Secon*, 2009.

[10] "KPROBE." [Online]. Available: http://www.redhat.com/magazine/005mar05/features/kprobes/

[11] "Soekris engineering," http://www.soekris.com/.

[12] "D-itg, distributed internet traffic generator." [Online]. Available: http://www.grid.unina.it/software/ITG/