Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 52 (2008) 1583–1602

Computer Networks

www.elsevier.com/locate/comnet

A link layer adaptive pacing scheme for improving throughput of transport protocols in wireless mesh networks

A. Antony Franklin [*], C. Siva Ram Murthy

Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600 036, India

Received 16 July 2007; received in revised form 28 January 2008; accepted 5 February 2008
Available online 14 February 2008

Responsible Editor: L. Lenzini

Abstract

In this paper, we study the performance of a static multihop wireless network, specifically that of the backhaul network of a two-tier Wireless Mesh Network (WMN) operating on IEEE 802.11 Medium Access Control (MAC) protocol. The performance of an IEEE 802.11 based backhaul network is greatly affected by the MAC contention and congestion in the network. If the sources pump data into the network than can be supported, loss rate increases due to MAC contention and congestion in the network. This also leads to the problem of unfairness among flows. In this paper, we propose a Link Layer Adaptive Pacing (LLAP) scheme that adaptively controls the offered load into the network. This improves the performance of higher layer protocols without any modifications to them. Our LLAP scheme estimates the four hop transmission delay in the network path without incurring any additional overhead (Control packets) and accordingly paces the packet transmissions to reduce MAC contentions in the network. We implement the LLAP scheme in ns-2.29 network simulator and extensively study its performance for both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) traffic in different network scenarios. In all the cases, our scheme shows a significant improvement in the performance of both UDP and TCP traffic.

Keywords: Wireless mesh networks; Medium access control; IEEE 802.11 standard; TCP and UDP traffic; Congestion and contention

1. Introduction

Wireless mesh networking has emerged as a promising technology to meet the challenges such as providing flexible, adaptive, and reconfigurable architecture while offering cost-effective solutions to service providers, in next generation wireless networks [1]. In a Wireless Mesh Network (WMN), the backbone mesh network formed by a set of mesh nodes is a static multihop wireless network. The clients (for example, in a community WMN) are connected to the edge mesh nodes of the backbone network. Some mesh nodes, called gateway nodes in the backbone network, provide Internet connectivity to the clients of the WMN. The multihop backbone wireless network has to be utilized efficiently in order to improve the overall performance of the network. In a two-tier WMN architecture, the

[*] Corresponding author. Tel.: +91 4422574361.
E-mail addresses: antony@cse.iitm.ernet.in (A.A. Franklin), murthy@iitm.ac.in (C.S.R. Murthy).

communication of a client with a mesh node and that between mesh nodes is carried out using either a different technology or different channels so that both are independent. Most of the Internet based applications use Transmission Control Protocol (TCP) as a transport protocol, since it provides end-to-end reliable transmission of data. Many applications such as audio and video streaming use User Datagram Protocol (UDP) as a transport protocol, since they require faster delivery of data rather than reliable transmission. As WMN is used as backbone network for accessing the Internet as well as for community networking, the traffic in WMNs is from numerous applications which use different transport protocols (both reliable and unreliable transport protocols).

As applications such as audio and video streaming coexist with TCP traffic, improving only the performance of TCP protocol may not improve the overall performance of the WMNs. As the performance of TCP is greatly affected by the packet losses in the network, reducing the losses will improve its performance considerably. In wired networks, packet loss is mainly due to the buffer overflow at the intermediate routers. But in the case of multihop wireless networks, packet loss could also occur due to erroneous wireless channels, MAC contention, unstable network conditions, and mobility of nodes (in Mobile Ad hoc Networks). Bursty nature of the traffic also increases MAC contention and packet losses in the network due to self contention (packets of the same flow collide with each other) thereby affecting the performance of flows in a multihop wireless network. As TCP uses window based congestion control, it generates bursty traffic and leads to self contention.

It is shown in the literature that, in IEEE 802.11 based multihop wireless networks, if the interference range is twice the transmission range, the contention in the network path can be reduced by evenly spacing the packet transmissions with 4-hop transmission delay (FHD) at the source node [2]. The FHD is defined as the time for transmitting a packet from a node to the 4th hop node on the downstream path. The spacing between packet transmissions can be done at the transport layer by properly estimating the FHD of the network path. Although, this may provide a separation (delay) between successive transmission of packets for each flow at the higher layer, due to very high contention in the network or traffic from the multiple flows, there may not be such a separation at the Medium Access Control (MAC) layer in reality.

In WMNs, a number of clients can generate TCP and UDP traffic which goes in the same multihop path from an edge node to another edge node or from an edge node to the gateway node. Hence, all the packets going in a path have to be scheduled with an interval of FHD to reduce the contention between the packets, thereby achieving better channel spatial reuse. This spacing of packet transmissions is required in multihop wireless networks irrespective of higher layer protocols used when the flows are running for more than four hops. In this work, we aim to reduce the MAC layer contention by using an adaptive pacing mechanism at the link layer. Our proposed Link Layer Adaptive Pacing (LLAP) scheme tries to reduce the contention in the network by properly scheduling the packets at edge nodes thereby increasing the channel spatial reuse in the network.

We use a cross-layer approach for scheduling of packets and estimation of FHD in a path. Our approach estimates the FHD in a path by measuring the queuing and *transmission delay*[1] incurred at the bottleneck node in a distributed manner. The main contributions of this paper are as follows:

- A performance study of multihop wireless networks for both UDP and TCP traffic.
- The new LLAP scheme to reduce the MAC contention in the network for achieving better channel spatial reuse.
- The estimation of FHD in a path in a distributed manner without additional control packet exchanges between the edge nodes.

The rest of the paper is organized as follows. Section 2 discusses the related work in the literature and provides the motivation for our work. Section 3 gives a class of networks considered for our study. Section 4 describes the problem with multihop wireless networks for both UDP and TCP traffic. In Section 5, we describe the design and implementation of our LLAP scheme. In Section 6, we demonstrate the responsiveness of our LLAP to congestion in the network. In Section 7, we evaluate the LLAP scheme for both UDP and TCP traffic in different network scenarios. Finally, we conclude the paper in Section 8.

[1] Transmission delay is the sum of channel access time and transmission time.

2. Related work and motivation

There are several solutions in the literature to improve the performance of transport layer protocol over IEEE 802.11 based multihop wireless networks. But as explained in the previous section, the traffic in WMNs consists of traffic from different applications that use different transport protocols. We now briefly describe the various solutions available in the literature for improving the performance of transport protocol over multihop wireless networks.

In [3], the authors analyzed the performance of TCP over a multihop wireless channel. TCP achieves maximum throughput by maximum spatial reuse of the shared wireless channel. TCP identifies the overload of the network (congestion) by packet losses. But compared to a wired network where the network overload is indicated by the packet drop at the queue of an intermediate router, in a multihop wireless network, the network overload is also indicated by link layer contention losses. There will be some packet losses due to the lossy nature of the wireless medium which can be recovered by the MAC retransmission mechanism. IEEE 802.11 has a retransmission count of 7 to recover the losses due to channel errors. As the offered load increases in the network, the loss probability due to link contention also increases. To improve the performance of TCP over multihop wireless networks, several variants of TCP such as TCP ELFN [4] and TCP-Feedback [5] were proposed. These protocols try to distinguish between congestion and non-congestion losses in the network and appropriately take actions to achieve better performance. The rate based protocols such as ATP (Ad hoc Transport Protocol) [6], RBCC (Rate Based end-to-end Congestion Control) [7], and AR-TCP (Loss-Aware Adaptive Rate based TCP) [8] estimate the available bandwidth between the source and destination, and transmit the packets at the estimated rate. In these protocols, each node estimates the available bandwidth at that node and appends the information into the packets passing through it. Upon receiving the packets, the receiver estimates the available bandwidth on the path and sends it to the source through the acknowledgment packets. In ATP, the available bandwidth at a node is estimated by averaging the sum of queuing and transmission delay experienced by each packet at the node. In RBCC, the sending rate of each flow is determined by the channel utilization status at the bottleneck node. The channel utilization status of each node is measured using a new metric called *Channel Busyness Ratio* which is the ratio of the total length of the busy period to the total time during a time interval. In AR-TCP, the channel quality is considered for estimating the available bandwidth at each node. The link quality is measured using the signal strength received from each of its neighbors.

There are other solutions such as Distributed Link RED (LRED) [9] and Neighborhood RED [10] that incorporate Random Early Detection (RED) mechanism in queues to improve the TCP performance. LRED improves the performance of TCP flows by implementing RED at the queues and reacting to continuous packet collisions by increasing the MAC backoff time by one packet transmission time.

Several researchers identified that the poor performance of TCP in IEEE 802.11 based multihop wireless networks is due to the underlying routing and MAC layer protocols [11]. Due to the broadcast nature of wireless channel, the neighboring nodes in the network can not transmit simultaneously. So the packets of multihop flow contend with each other for the channel at successive hops, if the data arrived at the source is bursty in nature. It is well known that TCP generates bursty traffic based on the current congestion window size. This leads to self contention and increases the chances of dropping the packets. Even in static wireless networks, the routing protocol cannot distinguish packet loss due to congestion and packet loss due to broken route. So the packet losses in the network may lead to route failure which takes considerable amount of time to establish the route again. During this time, the TCP would have timed out thereby affecting the performance of TCP.

A solution to improve the performance of TCP over multihop wireless networks which spreads the packet transmissions with FHD at the transport layer is TCP with Adaptive Pacing (TCP-AP) [12]. The main objective of TCP-AP is to spread the transmission of packets from the source so as to avoid self contention. It is well known that the packets in multihop wireless networks do not collide with the transmission of the packets four hops away. Hence, in a multihop path, if packet transmissions from the source are spaced in time equal to FHD, self contention can be eliminated completely. TCP-AP calculates the FHD from the round trip time (RTT) estimated at the source and transmits the packets of the current window with spacing equal to the calculated FHD. Pacing at the transport layer makes the packets in the MAC layer evenly spaced with FHD interval for that particular flow. This improves the performance of TCP if the source and destination has only one flow. The estimate of FHD assumes that all the intermediate links in the path use same data rate. This is always not true as multi-rate transmission selects the transmission rate based on the quality of the channel. Further, if the number of flows in a path increases, TCP-AP does not achieve perfect scheduling at the MAC layer. As the estimation of FHD by TCP-AP is by the RTT of the TCP packets, it works well when both the source and sink are in the WMNs. If the TCP flows are between a mobile client and the Internet, the FHD estimation by RTT will fail. In [13], a cross-layer approach is used at the gateway node to pace the packets in multihop wireless networks. Here, the FHD estimation is done by tapping the TCP data and ack packets at the gateway node.

The main problem with an IEEE 802.11 based multihop wireless network such as the backbone network in a WMN is that, if each edge node pushes data into the network, congestion and MAC contention increase and the overall utilization of the network reduces significantly. To efficiently use the radio resources in the network, each edge node in the network should inject data that can be supported by the network. MAC contention on a multihop path can be reduced by evenly spacing all packets on that path. This reduces the contention loss on that path. This spacing of packet transmissions at the MAC is essential for all higher layer protocols, if the traffic is bursty in nature or the network is congested. Hence, we propose an adaptive pacing scheme at the link layer in IEEE 802.11 based multihop wireless networks. By perfectly scheduling the transmission of packets (i.e., pacing packets with FHD), self contention of packets belonging to a flow can be reduced and maximum transmission rate of $R_{\max} = 1/\text{FHD}$ [2] can be achieved.

The available capacity on a particular path of the network can be estimated by using/transmitting additional control packets. But this consumes some amount of bandwidth and reduces the available capacity of the network. In our work, we use the broadcast property of the wireless channel that allows overhearing of transmission of neighboring nodes, to measure the FHD in a path. Our estimation algorithm is based on the following assumptions: 1. The FHD in the path is at most four times the queuing and transmission delay incurred at the bottleneck node, 2. The maximum data rate achievable on the path depends on the queuing and transmission delay of the bottleneck node in the path, 3. Increasing the queuing delay in all the upstream nodes of the bottleneck node does not reduce the achievable data rate, and 4. The estimation of amount of time that a packet spent in a node includes the queuing delay, channel access time, and transmission time. As discussed above, in IEEE 802.11 based multihop wireless networks, congestion in the network is not only due to queuing of packets at the intermediate nodes but also due to the channel contention at the intermediate nodes in the path.

There exist numerous research proposals in the literature which focus on other issues such as resolving of hidden and exposed node problems, fairness to the competing flows, efficient utilization of bandwidth in the network and battery power at mobile nodes, and efficient error handling techniques to deal with erroneous wireless channels, for improving the performance of multihop wireless networks by making modifications/improvements in the underlying IEEE 802.11 MAC. An excellent survey of these proposals can be found in [14].

3. Two-tier wireless mesh networks

We consider an IEEE 802.11 based static multihop wireless network as the backbone network of WMN for serving the clients. Each node in the backbone network is called a mesh node. Some of the mesh nodes in the backbone network, called gateway nodes, connect to the Internet through wired link. Such a type of network is quite common in a number of applications with WMNs. The clients utilize the mesh backbone network by getting connected to the edge nodes of the WMNs. In [15] and [16], the authors presented methods for deployment of two-tier WMNs. Here, all the intermediate mesh nodes are used as relaying nodes. This is shown in Fig. 1 with edge mesh nodes and core mesh nodes with different colours. The client nodes get connected to the edge mesh nodes with a different networking technology or different channel as used in mesh nodes. Hence, the communication of client nodes with the edge nodes does not interfere

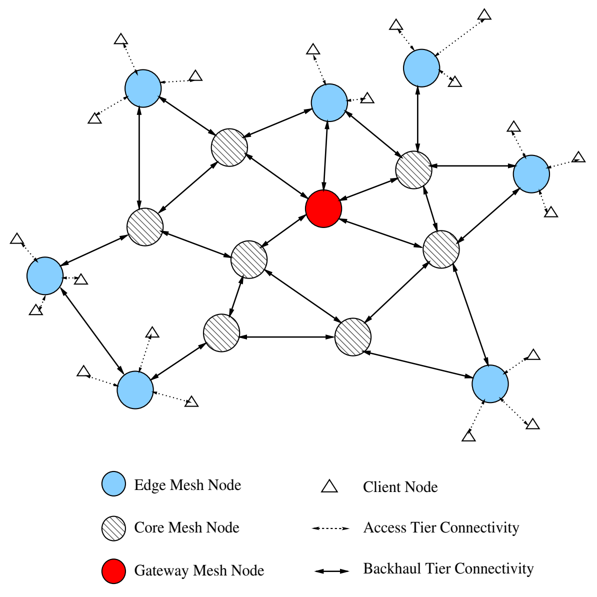

Fig. 1. An example two-tier WMN with edge and core mesh nodes and one gateway node.

with the communication between the mesh nodes. The client nodes connect to one of the edge mesh nodes in a single hop. This is called a two-tier architecture of WMNs. In this type of networks, the backbone network alone can be considered as an independent network running its own routing protocol such as AODV [17]. This network can be considered as an unplanned, single-radio WMN (for, e.g. a community network), in which some mesh nodes called gateway nodes have wired connectivity to the Internet. The main advantage of WMNs in community networking is that the traffic between the clients need not go through the Internet. Instead, it goes through the mesh nodes. The traffic between the client nodes goes through the backbone network, without using the Internet. This improves the overall performance of the community networking. Traffic to the Internet reaches a gateway node through other mesh nodes in a multihop path. In large scale networks, the average hop length of flows could be more than four and many TCP and UDP flows might exist between any two edge nodes. At the core nodes, flows from different edge nodes join and flows to different edge nodes split. The bottleneck node for any flow depends on the traffic pattern and the path that the routing protocol chooses. So the available bandwidth between any two edge nodes depends on the traffic pattern in the network. The mesh nodes use separate routing protocol to route packets in the backbone network. For each flow, the edge node through which it enters the backbone network is called the ingress node and the edge node through which it leaves the backbone network is called the egress node.

4. Performance problem with IEEE 802.11 multihop networks

The performance of a multihop wireless network degrades if the sources push more traffic than the capacity of the network. This is due to the fact that the packets get collided in the network if more packets are pumped into the network. For our performance study, we used the ns-2.29 network simulator. In this section, we consider a 10-hop chain network with each node separated by 200 m. The transmission and interference ranges are 250 m and 550 m, respectively. All nodes use 802.11 DCF with RTS/CTS enabled unless otherwise specified. The data rate and basic rate are 11 Mbps and 1 Mbps, respectively. Ad hoc On-Demand Distance Vector (AODV) is used as the routing protocol and two-ray ground propagation model is used for the simulations. We consider one flow between the end nodes of the network (from Node 0 to Node 9) as shown in Fig. 2. In this section, we study the performance of both UDP and TCP flows on this 10-hop chain topology. To study the performance of UDP, we use a CBR traffic with varying data rate for different packet sizes. For TCP, we use FTP traffic with fixed packet size of 1460 bytes.

4.1. Single UDP flow

We estimate the achieved throughput of a UDP flow in a chain topology by varying the offered load in the network. By varying the interval between the successive packets of the CBR traffic, we measure the end-to-end achieved throughput with and without enabling RTS/CTS mechanism of MAC. In both the cases when the offered load is initially low, the achieved end-to-end throughput increases linearly but reaches a maximum and starts decreasing with

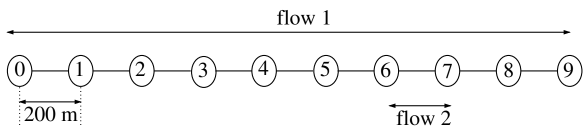

Fig. 2. Chain topology with two flows.

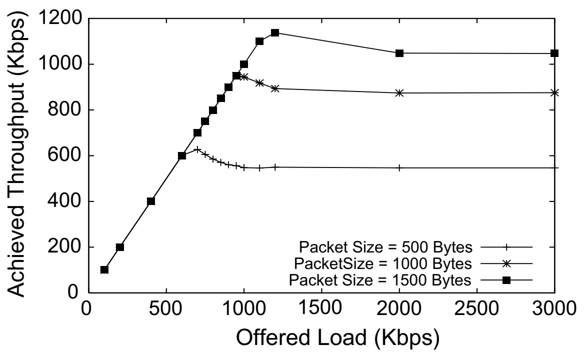

Fig. 3. Achieved throughput vs offered load with RTS/CTS disabled.

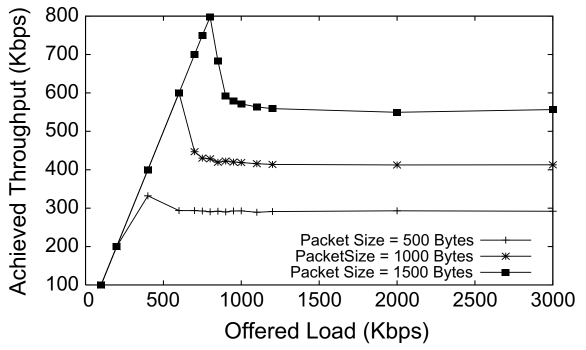

Fig. 4. Achieved throughput vs offered load with RTS/CTS enabled.

increasing offered load. The results are shown in Figs. 3 and 4. We show the end-to-end throughput for three different packet sizes in the CBR traffic. The results clearly show that if the offered load is more than the available capacity on the multihop path, the end-to-end throughput decreases significantly. This is due to the fact that the packets in the multihop path collide with the packets from the upstream node due to the well known hidden terminal problem in 802.11 DCF. Enabling the RTS/CTS mechanism also could not reduce the collisions in the network and the throughput of the UDP flow decreases drastically with increase in the offered load above a certain point.

4.2. Single TCP flow

The performance of the TCP flow is affected greatly by the packet loss in the network. Due to the burstiness of the packets in the multihop path, packets collide in the network which increases packet loss. Due to this packet loss, the performance of the TCP degrades significantly. We measured the goodput, number of TCP timeouts, number of transmitted packets, number of retransmitted packets and loss percentage of a single TCP flow over the 10-hop chain topology (shown in Fig. 2) with 2 Mbps channel data rate. Table 1 shows these values for a TCP flow running for 250 s and Fig. 5 shows the progression of TCP window with time. For a period of 250 s, we found large number of timeouts and high packet loss percentage. The packet losses here are not due to congestion or wireless loss but due to the MAC contention from the upstream packets in the path. Due to these packet losses in the network, the window progression is not smooth and there are larger number of timeouts in TCP. Due to this self contention, TCP window size goes to unity most of the time by these timeouts. This reduces the throughput of the TCP flow significantly. Enabling RTS/CTS in IEEE 802.11 DCF does not help in improving the performance of TCP over multihop wireless networks. Though RTS/CTS mechanism reduces the hidden node collisions to some extent, it does not avoid the packet losses due to MAC contention completely.

The main problem of packet loss on the multihop path for a TCP connection is due to the burstiness of packet transmission by the TCP connection. It is well known that on a multihop path, if the transmission of packets are spaced in time with four hop transmission delay, the overall performance of the multihop flow improves. This spacing can be realized by properly scheduling the packet transmission at the source such that each packet transmitted reaches the fourth hop before transmitting the next packet. This reduces the MAC contention in the network due to spatial channel reuse. TCP-AP [12] does this by estimating the four hop transmission delay using the packet Round Trip Time (RTT) and schedules the packets with interval of four hop transmission delay. Here we show that, TCP-

Table 1
Performance of single TCP flow on a chain topology

Protocol	Goodput (Kbps)	Timeout (count)	Packet loss (%)
NewReno (RTS/CTS Disabled)	96.64	27.33	3.11
NewReno (RTS/CTS Enabled)	88.50	53.8	7.93
TCP-AP (RTS/CTS Disabled)	244.56	9.56	0.18

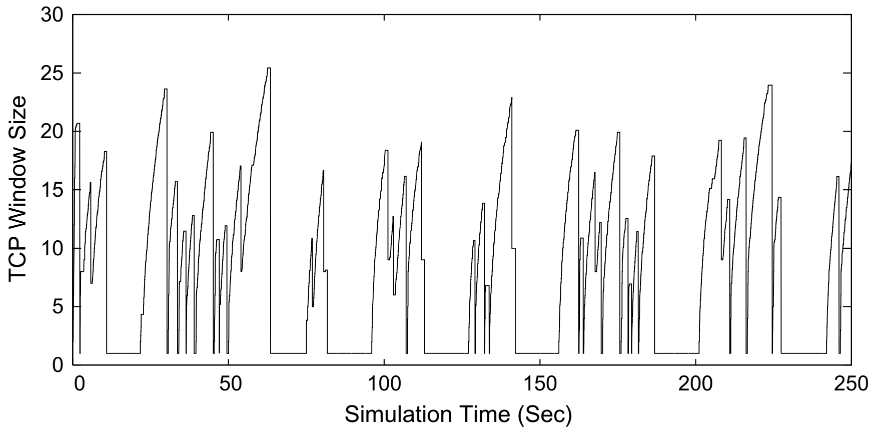

Fig. 5. Progression of window growth with time of a TCP NewReno flow over a multihop chain.

AP improves the performance of the single TCP flow over a multihop path with all nodes operating at same data rate. This can be seen from the results in Table 1. In a two-tier architecture proposed in Section 3, modifications at the transport layer are not feasible as the source and destination nodes are not in the backbone network. The performance of a flow is greatly affected in the multihop wireless network operating on IEEE 802.11 MAC protocol in the mesh backbone network rather than in the wired counterpart of the flow. In this work, we estimate the four hop transmission delay between each pair of edge nodes of the backbone network and control the packet transmission in the network. This improves the performance of the transport protocols over two-tier WMNs.

5. The link layer adaptive pacing scheme

The implementation of Link Layer Adaptive Pacing (LLAP) scheme has two phases (components). One is the estimation of FHD on the path between each ingress and egress pair and the other one is sending packets into the network with pacing delay (interval between adjacent packet transmission) of FHD. The FHD estimation on the path between ingress and egress pair is done by propagating the congestion information of the bottleneck node to the ingress node in a distributed manner. In the absence of congestion in the network, the FHD of a path is four times the one-hop transmission delay in the path. But if the network is congested, it should be calculated as four times the sum of queuing delay, channel access time, and transmission time at the bottleneck node.

5.1. Node architecture and distributed scheduler

In this section, we provide a node architecture and a distributed scheduler at the link layer for estimation of FHD for each ingress and egress pair. It is assumed that all core nodes in the backbone network maintain a separate queue called *Input Queue* for the packets destined to the same egress node and a *Transmission Queue* which contains the packets that are ready for transmission. The scheduler moves the packets from *Input Queue* to *Transmission Queue*. Both these queues serve the packets in First-In-First-Out (FIFO) fashion. The key idea here is to introduce delay between arrival and departure of packets that are destined for a particular egress node, if required, to propagate the congestion information of the downstream node to the ingress node. All the incoming packets are placed into the corresponding *Input Queue* based on the egress node that a packet has to reach. This information will be obtained from the routing layer. Each node maintains the average time that the packets belonging to a particular egress node (d) spent at this node (HT^d) and at the downstream node (NHT^d). The scheduler moves a packet from *Input Queue* to the *Transmission Queue* based on the values of HT^d and NHT^d. If NHT^d at a node is greater than HT^d for egress node d, it adds additional delay to move the packet from its *Input Queue* to the *Transmission Queue*. At the ingress node the delay incurred in moving the packet from *Input Queue* to *Transmission Queue* is the estimated value of FHD^d for that egress node d. An illustration of node architecture that consists of queues and the scheduler is shown in Fig. 6.

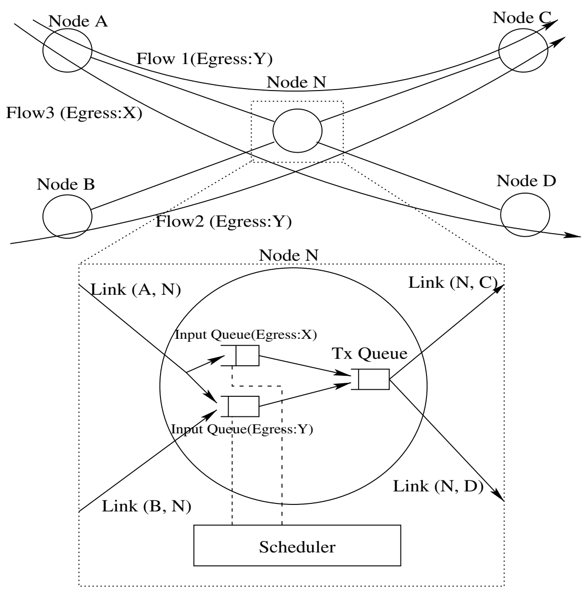

Fig. 6. An illustration of node architecture.

5.2. Estimation of 4-hop transmission delay (FHD)

In our LLAP scheme, the bottleneck node's queuing and transmission delay is propagated to the ingress node by introducing additional delay to the transmission of packets at all the intermediate core nodes such that all upstream nodes make the queuing and transmission delay equal to that of the bottleneck node. Each node can measure NHT^d by overhearing the transmission of the downstream node except the node preceding the egress node. If the downstream node's queuing delay and transmission delay is more than the node's queuing and transmission delay, the node introduces additional delay which makes the queuing and transmission delay of this node equal to that of the downstream node. By doing this, it is indirectly notifying its immediate upstream node about the congestion at the bottleneck node. This process is repeated at each upstream node and finally the ingress node finds the bottleneck node's queuing delay and transmission delay by the estimate of NHT^d. Once the ingress node gets the congestion level at the bottleneck node, it estimates the FHD for egress node d (FHD^d).

5.3. Propagation of congestion information

The distributed scheduler discussed above propagates the congestion information in the network to the ingress node in a distributed fashion. The HT^d and NHT^d are estimated as follows. Whenever a packet is transmitted from a node, the amount of time that the packet spent in this node is calculated by difference between the time of arrival of the packet to this node and the time of completion of packet transmission in the MAC layer. Let us denote this by HT^d_{current}. HT^d is calculated using Weighted Moving Average (WMA) method with weight α as shown below

$$HT^d = HT^d_{\text{old}} \times \alpha + HT^d_{\text{current}} \times (1 - \alpha).$$

The amount of time the packet spent in the downstream node is the difference between the time at which the packet is transmitted from this node and the time at which the node overheard the transmission of the same packet from the downstream node. Let us denote this by NHT^d_{current}. NHT^d is calculated using WMA with weight α as shown below

$$NHT^d = NHT^d_{\text{old}} \times \alpha + NHT^d_{\text{current}} \times (1 - \alpha).$$

The pacing delay to be introduced for each packet belonging to egress node d is PD^d and calculated as given below with initial value of PD^d as 0

$$PD^d = PD^d + (NHT^d - HT^d).$$

5.4. Pacing at the ingress node

As discussed earlier, each upstream node in the path from the bottleneck node ensures that the amount of time the packets belonging to a particular egress node spent in it is equal to the amount of time the packets spent at the bottleneck node. Now, the ingress node can estimate the FHD^d by multiplying the estimate of NHT^d at this node by a constant k depending upon the hop length of the path. If the hop length is less than four then k is equal to hop length of the path, and four otherwise. Hence, the pacing delay at the ingress node for a given egress node d is calculated as follows:

$$PD^d = k \times NHT^d$$

The scheduler moves the packets from the *Input Queue* to the *Transmission Queue* with a pacing delay (PD^d) which makes the delay between successive transmission of packets for that particular egress node to be FHD^d.

5.5. Implementation

We implemented the LLAP scheme in ns-2.29 network simulator. We implemented the node architecture proposed (shown in Fig. 6) in Section 5.1 in Interface Queue (IFQ) of ns-2.29. At the ingress node all the incoming packets will be from clients that are connected in one-hop to that ingress node. At the egress node, all packets will be destined to clients that are connected to that egress node. But at the core nodes, packets from different ingress nodes will join and packets to different egress nodes split. We implemented an interface queue with the architecture proposed and the scheduler moves the packets from the *Input Queue* to the *Transmission Queue*. Algorithm 1 shows the scheduling algorithm implemented at the IFQ to realize the LLAP. The delay between the adjacent packets moving from each *Input Queue* to the *Transmission Queue* is *PD* and is calculated at the core node and at the ingress node as explained before. The propagation of congestion information requires the estimation of *HT* and *NHT* at each node. Algorithm 2 describes the estimation procedure of *HT* and *NHT*.

Algorithm 1. Scheduling Algorithm at the Interface Queue.

1. When a packet is received, depending on the destination of the packet, find the egress node to which the destination client is connected and put the packet in the corresponding *Input Queue*. If the *Input Queue* for that egress node does not exist, create one and put it.
2. Estimate the *NHT* for each egress node passively by overhearing the packet transmissions at the MAC layer by adjacent node.
3. Create timers for each *Input Queue* with timer value equal to *PD* at both core node and ingress node with respective values.
4. Whenever the timer belonging to any *Input Queue* expires, move a packet from the corresponding *Input Queue* to *Transmission Queue* and restart the timer with the current estimated value of *PD*.
5. After each successful transmission by the MAC layer, get the next packet from the *Transmission Queue* to transmit.

Algorithm 2. *HT* and *NHT* Estimation Algorithm at the MAC Layer.

1. When a packet arrives at a node from the upstream node, update the time of arrival of the packet ($T_{a(n)}$).
2. Whenever a packet is successfully transmitted by the MAC layer corresponding to an egress node, get the current time that gives the time of arrival of the packet at the downstream node ($T_{a(n+1)}$) and subtract the packet transmission time to get the time of departure of the packet from this node ($T_{d(n)}$).
3. *HT* sample is calculated as $T_{d(n)} - T_{a(n)}$. Find the average using WMA method.
4. Keep the identification of the packet to compare when overhearing the packet transmission by downstream node.
5. Whenever a packet transmission is overheard, get the packet and if the packet identification is available, get the current time which gives the time of successful transmission of the packet at the downstream node ($T_{d(n+1)}$).
6. Remove the identification of the packet from the node.
7. The *NHT* sample is calculated as $T_{d(n+1)} - T_{a(n+1)}$. The average value is calculated using WMA method.
8. If the packet is not overheard for a period of $4 \times NHT$, the packet might have been dropped at the downstream node or the packets are queued up in the downstream node for longer time or the node is busy during that time. In that case, the identification of the packet will be removed from the node and the *NHT* sample value is taken as $4 \times NHT$.

The implementation of LLAP incurs both memory and computational overhead at each node in the backbone network due to the maintenance of state information of packets. Each node has to buffer information about the packet belonging to a particular egress node for comparison when overhearing the packets. The amount of memory required at each node for maintaining the state information is proportional to the number of egress nodes in the network i.e., the number of edge nodes in the network. For the identification of the packet, a randomly generated sequence number (2–4 bytes) can be added to the packet header by the ingress node before transmitting it into the network. This will be an additional overhead in the packet. As buffering the whole packet requires lot of memory, we tried to compare the packets at the node without storing the packet as such. We used the Frame Check Sequence (FCS) of the data content (4 bytes) of the packet for comparison. The IEEE 802.11 MAC calculates the

FCS of the whole packet including header for error detection. We can calculate the FCS of the data content alone along with the FCS calculation of the packet for error detection to identify the packet. Apart from the packet identification, the WMA values of *HT* and *NHT* for each egress node are also maintained. As the FCS calculation done by MAC for each packet is received even for the overheard packet, there is no computational overhead. The overall memory overhead is $k \times N$, where N is the number of edge nodes in the network and k is the number bytes per egress node entry. The computational overhead is only searching the entry for each egress node. The problem with this approach is that if the data content of the packets of different egress is same and a node overhears the transmission and identifies wrongly, that sample for *NHT* estimation goes wrong. This will not have significant effect on the performance of the protocol as samples are averaged using WMA and the probability of two packets having same data and meeting at a downstream node at the same time is very low.

The propagation of congestion information to ingress node is done by estimating the congestion at the downstream node by overhearing the packet transmission at every node in the network. In 802.11 based network, a node may not overhear all the packets transmitted by the downstream node due to hidden terminal problem. This may happen when the node transmits/receives packet from some other node other than the downstream node at the time of the transmission of the downstream node. Though CSMA/CA mechanism tries to avoid two nodes transmitting simultaneously when they are in transmission range of each other, there is a probability of two nodes transmitting simultaneously when the load in the network is high. This leads to missing of overhearing from downstream node. This may also happen when the packet is dropped at the queue of the downstream node or stayed in the downstream node for a long time. Whenever it happens, the node that estimates the NHT will miss one sample for Weighted Moving Average (WMA). As we are buffering only one packet at the upstream node for comparison, we have to remove the entry whenever we are missing the overhearing. For this purpose, whenever a packet belonging to a particular egress is transmitted, a timer (*Overhear Timer*) will be started. If the *Overhear Timer* expires, the entry of the buffered packet will be removed and the next packet transmitted for that egress will be buffered. As the LLAP scheme reduces congestion in the network, the probability of missing the overhearing will not go high even though offered load is high. Further we validate the effect of this problem in the next section.

6. Congestion identification, parameter tuning, and responsiveness

6.1. Simulation environment

For simulations, we considered only the backbone network of the two-tier WMNs operating with IEEE 802.11 MAC protocol. As clients are connected to the edge nodes of the WMN in single hop and do not interfere with the transmission of the backbone network, we do not put the clients in the simulations. We generate all the traffic at the edge node itself. For the simulation setup, the transmission range and carrier sense range are set to 250 m and 550 m, respectively. We enabled RTS/CTS handshake for MAC transmission. The channel capacity is set to 11 Mbps unless otherwise stated. We assume the queue size of all nodes to be 25 and it will be shared by all *Input Queues* and *Transmission Queue*. Unless otherwise stated, in all considered topologies, each node is 200 m apart from each of its adjacent nodes. AODV [17] is used as a routing protocol.

In all our experiments, we run the simulation for 500 s and ignore the behavior for the first 50 s for the TCP traffic to neglect the initial transients unless otherwise mentioned. All the results presented in this paper are averaged over 30 simulation runs with different random seeds. The goodput of each TCP flow is measured using the formula ((*number of bytes received* − *number of bytes retransmitted*)/*duration of the flow*). The fairness index between the TCP flows is calculated using Jain's fairness index $\frac{(\sum_{i=0}^{n} x_i)^2}{n \times \sum_{i=1}^{n} x_i^2}$ where x_i is the goodput of ith TCP flow and n is the number of TCP flows. For UDP traffic the achieved data rate is measured using the formula (*number of bytes received*/*duration of the flow*).

6.2. Identification of congestion

To validate the effectiveness of our proposed scheme, we compared the FHD estimated at the ingress node and actual FHD obtained from the simulation trace in the following scenario. We took a 10-hop chain network and generated two CBR flows: *flow1* from node 0 to node 9 and *flow2* from node 6 to node 7 as shown in Fig. 2. We assumed the channel capacity to be 11 Mbps. The data rate of *flow1* is set to 1000 Kbps and varied the data rate of *flow2*. We ran the simulation for 90 s by running *flow1* throughout the simulation time and *flow2* from $T_1 = 30$ s to $T_2 = 60$ s. In the absence of *flow2* the estimated FHD at node 0 is around 0.006 s, but when *flow2* started at T_1 the FHD estimate increases significantly and reduces the transmission rate of *flow1*. When *flow2* terminates at T_2, the FHD estimate again comes down to 0.006 s and there is an increase in the transmission rate. The FHD estimation at the ingress node for data rates of 500 Kbps, 1000 Kbps, and 2000 Kbps for *flow2* are shown in Figs. 7–9, respectively. This experiment shows that the congestion in the path can be identified at the ingress node by implementing the distributed scheduler discussed earlier.

The effectiveness of FHD estimation depends mainly on the propagation of congestion information to the ingress node by the distributed scheduler. The distributed scheduler propagates the congestion information by the estimation of NHT at each node in the path. This is done by overhearing the transmission of the downstream node. As mentioned in the previous section, all the packets transmitted by downstream node may not be overheard and to find the effectiveness of the NHT estimation, we measure the rate of expiry of *Overhear Timer* at all the nodes in the network. In the single chain shown in Fig. 2, the *Overhear Timer* never expires at any node in the path for varying rate of *flow*1 (*flow*2 is not present). This is because there will not be any hidden node

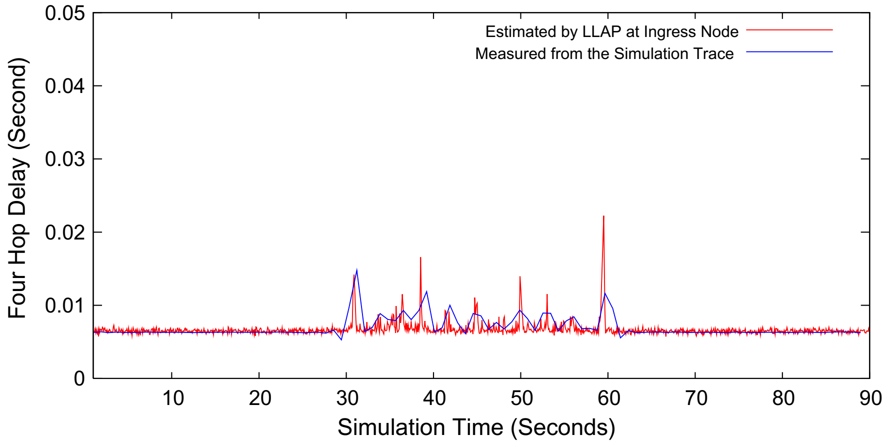

Fig. 7. FHD estimation with *flow2* operating at 500 Kbps.

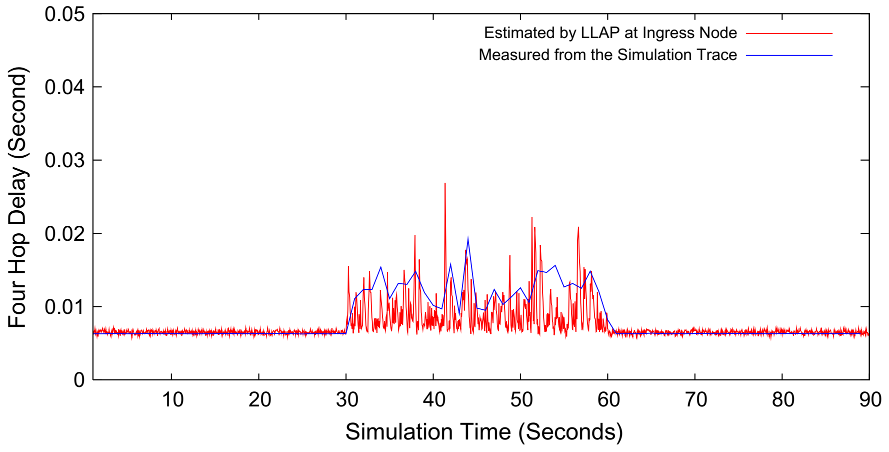

Fig. 8. FHD estimation with *flow2* operating at 1000 Kbps.

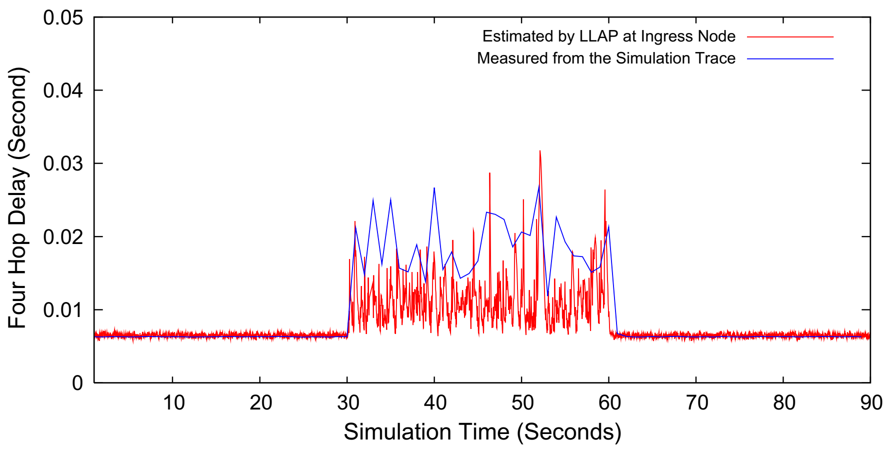

Fig. 9. FHD estimation with *flow2* operating at 2000 Kbps.

collisions as the packets are pushed at the ingress node with a time interval of FHD. So, we took a 5 × 5 grid with 6 flows as shown in Fig. 10 and measured the rate of expiry of *Overhear Timer*. We varied the rate of all the 6 flows simultaneously from 10 Kbps to 300 Kbps and measured the rate of expiry of *Overhear Timer* in each node. Fig. 11 shows the average rate of expiry of *Overhear Timer* with total offered load into the network. The rate of timer expiry is less than 1 per second per node even though the offered load is more than 1200 Kbps (1200 pkts/s).

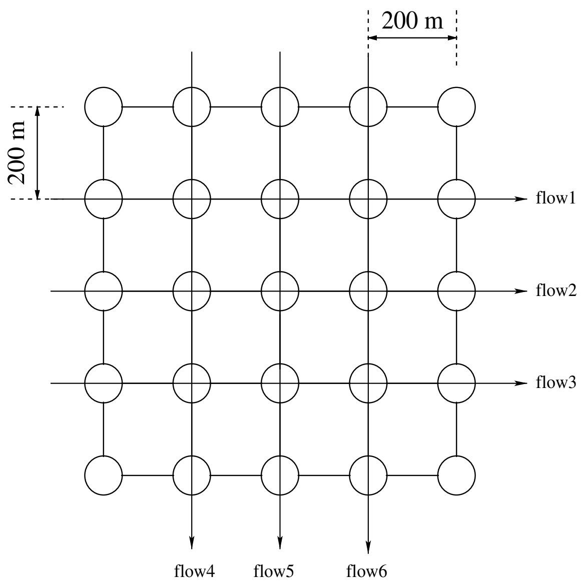

Fig. 10. 5 × 5 grid topology with 6 flows.

6.3. Parameter tuning

The value of α used in WMA smoothens the estimates of both NHT^d and HT^d with changes in the network conditions. These two values change significantly with the MAC contention in the network. The value of α contributes significantly to the performance of the network. To find the best value of α, we conduct an experiment with two competing TCP NewReno flows in the cross topology as shown in Fig. 12. We measure the aggregate goodput of the flows and Jain's Fairness Index between the flows. Due to symmetry of the two flows, the Fairness Index reaches unity with long simulation time. So we measure the short term unfairness between the two TCP flows with the metric Time To Fairness (TTF – time to achieve the desired fairness index by considering the goodput achieved by each flow). We choose an instant T_0 and measure the time ($T_0 + \mathrm{TTF}$) at which the Jain's Fairness Index reaches the desired fairness. We choose T_0 as 20 s and measured TTF for Fairness Index of 0.95. The aggregate goodput is measured over a simulation run of 500 s. The measured aggregate goodput and TTF between the two flows are shown in Fig. 13. One can observe from the figure that $\alpha = 0.9$ gives better aggregate goodput and TTF. We choose the value of α as 0.9 for the remaining simulation experiments.

6.4. Responsiveness

The responsiveness denotes how quickly our scheme adapts to the changes in network congestion. To illustrate the transient behavior of our pro-

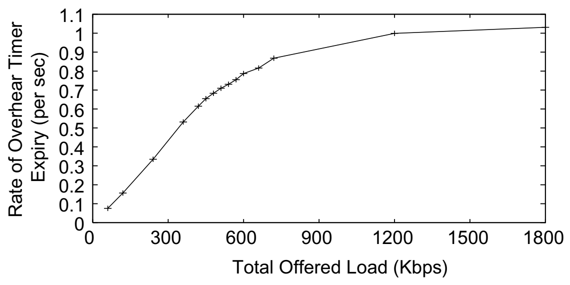

Fig. 11. Rate of expiry of the *Overhear Timer* with offered load.

posed scheme, we measure the goodput of two TCP flows in the same cross topology as shown in Fig. 12. Here the two TCP NewReno flows compete for the channel access. For this experiment, *flow1* runs from beginning of the simulation to $T_2 = 60$ s

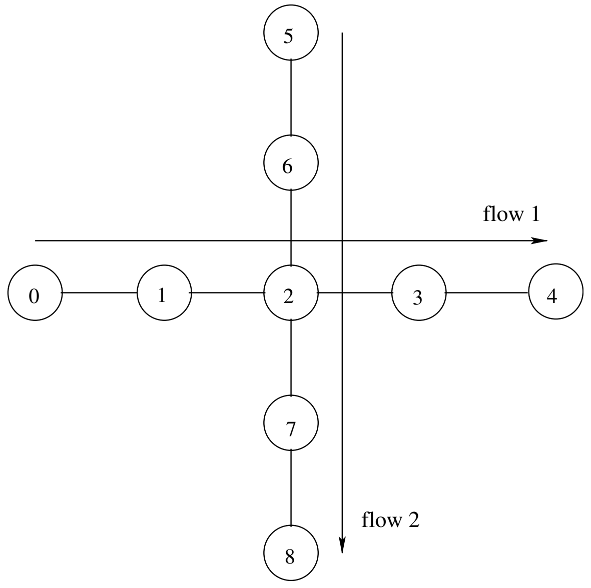

Fig. 12. Two flows on a cross topology.

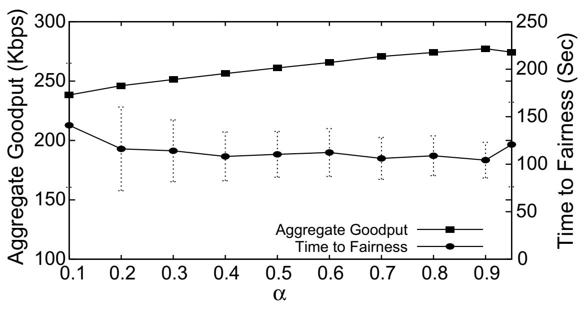

Fig. 13. Time To Fairness (TTF) for varying weight factor α in cross topology.

and *flow2* runs from $T_1 = 30$ s to end of simulation $T_3 = 90$ s. Figs. 14 and 15 plot the goodput vs simulation time for both flows with and without LLAP, respectively. With LLAP both TCP NewReno flows utilize the available bandwidth when there is no competing flow and share the bandwidth fairly between them when they compete for the bandwidth of the channel. But without LLAP, *flow1* captures the channel and utilizes almost the entire bandwidth even after *flow2* starts. The other observation here is that when congestion occurs due to the competing flows the propagation of congestion information to the ingress node causes the flows to respond quickly.

7. Performance measurement and simulation results

In this section, we study the performance of LLAP scheme in different network scenarios for different types of traffic. First, we study its performance over a single path between an ingress and egress pair. We study the performance of different types of traffic over this single path. Second, to study the performance of this scheme over a mesh backbone with many number of core and edge nodes, we took a grid topology and measured the performance under different traffic patterns. In all our simulation experiments we took the traffic between two edge nodes. But in real scenarios of two-tier WMNs, the traffic sources are between two mobile clients or between a mobile client and a node in the Internet. As LLAP estimates FHD between the edge nodes of the network, it improves the performance of higher layer protocol irrespective of where the source is located. The performance of TCP improves significantly even when either TCP source or sink is located in the Internet. The simulation setup is same as that mentioned in Section 6.

7.1. Chain topology

We measured the performance of both UDP and TCP traffic over the chain topology shown in Fig. 2 in this section. For UDP traffic, we considered both CBR and self similar traffic. For CBR traffic, we measured the achieved throughput, end-to-end delay, and end-to-end delay variation with and without LLAP by varying the data rate of the CBR traffic from 20 Kbps to 3000 Kbps. The increase in data rate at the source increases the achieved throughput at the receiver linearly as long as the time interval between packet arrivals at the

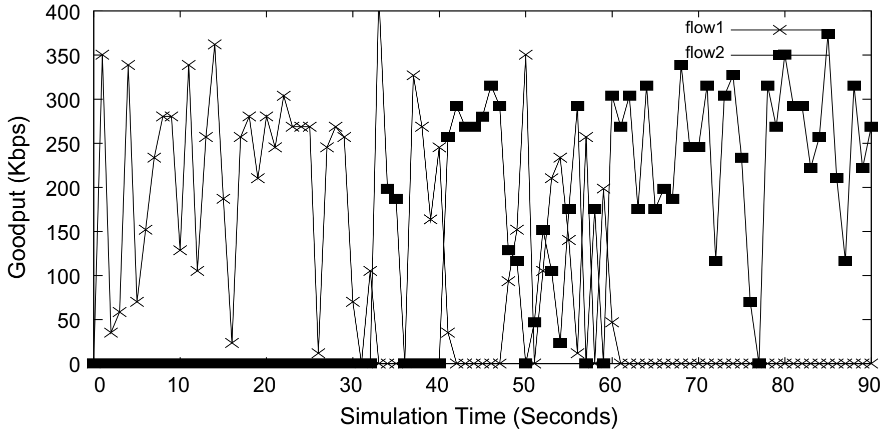

Fig. 14. Responsiveness of TCP NewReno without LLAP.

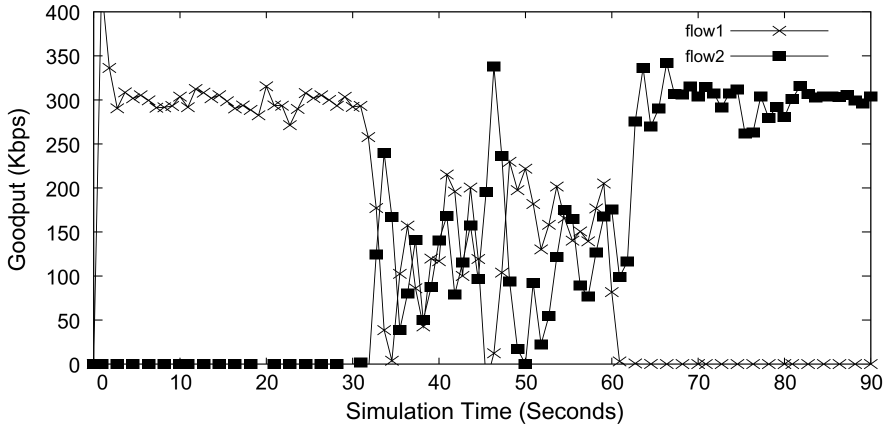

Fig. 15. Responsiveness of TCP NewReno with LLAP.

CBR source is less than FHD in the path for both the cases. Without LLAP, further increase in data rate of the source decreases the achieved throughput at the receiver as it increases the contention in the path. But in the case of LLAP, though the packet inter arrival time decreases below FHD, the ingress node pushes the packets into the network with an interval of FHD. So, the throughput at the receiver remains constant and does not reduce with increase in data rate of the CBR source. This can be observed in Fig. 16.

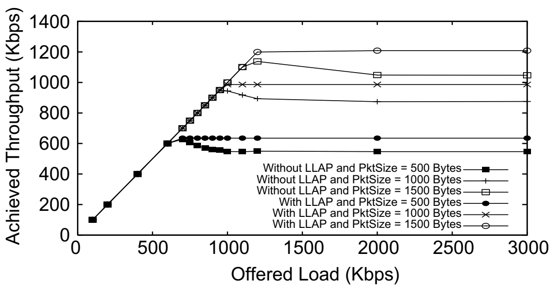

Fig. 16. Throughput vs offered load with and without LLAP for different packet sizes.

The increase in data rate of the source decreases the time interval between the consecutive packets from the source. As long as the interval between consecutive packets is less than the FHD, end-to-end delay will not change without applying LLAP. We can also see the same trend in the case of LLAP as LLAP will not incur artificial delay if the interval between consecutive packets is more than FHD. If the data rate of source increases further such that the interval between consecutive packets is less than FHD, the end-to-end delay increases with data rate and saturates. Without LLAP, the increase in end-

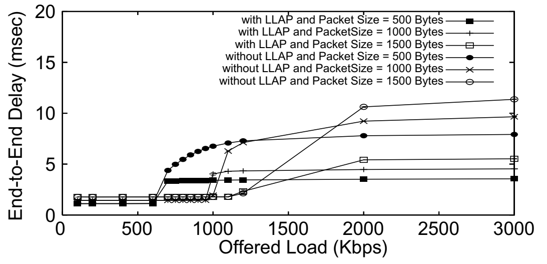

Fig. 17. End-to-end delay vs offered load with and without LLAP for different packet sizes.

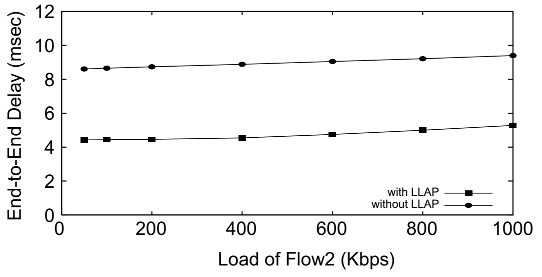

Fig. 19. End-to-end delay (flow1) vs congestion load (flow2) with and without LLAP.

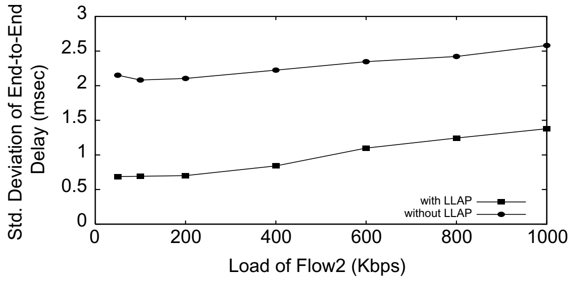

Fig. 20. Standard deviation of end-to-end delay (flow1) vs congestion load (flow2) with and without LLAP.

to-end delay is due to the fact that, packet queuing delay and the contention delay in the path increases with data rate of the source. In the case of LLAP, the ingress node pushes packets into the path with interval of FHD between consecutive packets. So, the increase in end-to-end delay is due to the packet queuing delay at the ingress node. The end-to-end delay saturates when the queue is full. This is shown in Fig. 17. From the figure, it is clear that introducing artificial delay at the ingress node does not increase the end-to-end delay of packets. The end-to-end delay is always less than the end-to-end delay when LLAP is not used. One more advantage of LLAP is that variation in end-to-end delay is reduced as shown in Fig. 18.

To show the end-to-end delay performance of LLAP in the presence of congestion, we varied the data rate of flow2 in Fig. 2 and measured the end-to-end delay of flow1. The results are shown in Figs. 19 and 20. Even in the presence of congestion, LLAP reduces both the end-to-end delay and end-to-end delay variation of the CBR flow.

In the case of self similar traffic, the average data rate of the source is varied and the achieved throughput is measured. We find that the LLAP improves the achieved data rate if we increase the average data rate of the self similar traffic as seen in Fig. 21.

In Section 4, we showed that the performance of TCP is greatly affected by self contention of TCP packets on the multihop path due to burstiness of the TCP flow. LLAP spreads the transmission of the packets in the multihop path in such a way that it reduces self contention on the path. We measured the goodput, number of TCP timeouts, number of transmitted packets, and number of retransmitted packets of a single TCP flow over the 10-hop chain topology (shown in Fig. 2) with 2 Mbps channel data rate. Table 2 shows these values for a TCP flow

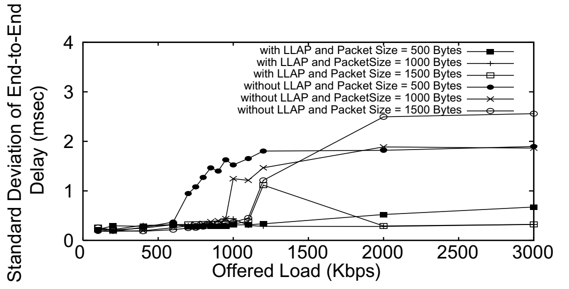

Fig. 18. Standard deviation of end-to-end delay vs offered load with and without LLAP for different packet sizes.

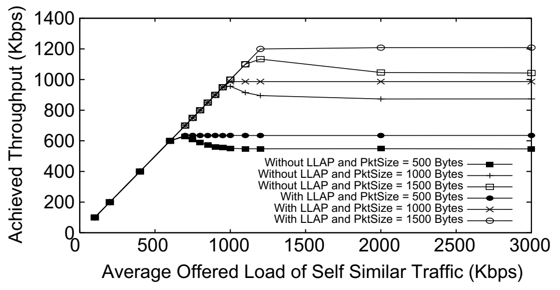

Fig. 21. Throughput vs average offered load of self similar traffic with and without LLAP for different packet sizes.

Table 2
Performance of single TCP NewReno flow on a chain topology

Protocol	Goodput (Kbps)	Timeout (count)	Packet loss percentage (%)
Without LLAP (RTS/CTS Disabled)	96.64	27.33	3.11
Without LLAP (RTS/CTS Enabled)	88.50	53.8	7.93
With LLAP (RTS/CTS Disabled)	248.75	2.66	0.83
With LLAP (RTS/CTS Enabled)	160.38	9.16	2.43

with and without LLAP running for 250 s and Fig. 22 shows the progression of TCP window with time. The progression of TCP window is smooth and the number of timeouts is reduced significantly by LLAP.

We considered three different cases for the study of performance of TCP with LLAP: 1. Measurement of goodput of one TCP NewReno flow by varying the hop length in a chain topology, 2. Measurement of aggregate goodput and fairness index of TCP NewReno flows by varying the number of flows between the ingress and egress pair of the chain, and 3. Measurement of goodput of a TCP NewReno flow over a 10-hop chain by choosing MAC data rate randomly for each node.

In the first experiment, by changing the length of the chain topology, we ran an FTP flow using TCP NewReno connection between end nodes of the chain and measured the goodput with and without LLAP. For comparative purpose, we measured the performance of TCP-AP without LLAP also. The

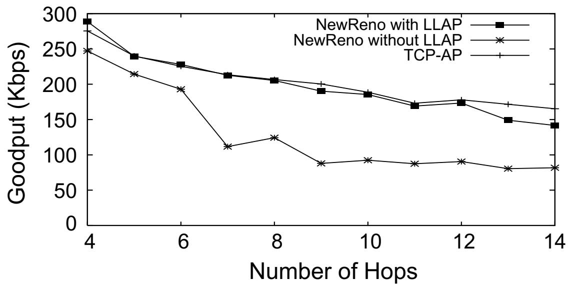

Fig. 23. TCP goodput vs hop length with and without LLAP.

results are shown in Fig. 23. The goodput of TCP NewReno over IEEE 802.11 MAC without LLAP reduces drastically when the number of hops increases from 4 to 7. This is due to the fact that the number of hidden terminal collisions are higher near the source node. But LLAP reduces the contention in the path and improves the goodput of TCP NewReno. We noted that TCP NewReno with LLAP achieves goodput comparable to that of TCP-AP. As only one TCP flow exists in the chain topology, pacing at the transport layer spreads the packet transmission at the MAC layer as well. Here, we found that our LLAP scheme gives goodput comparable to that of TCP-AP. TCP-AP estimates the FHD using the RTTs and it works well only if both TCP source and sink are in WMN. But our LLAP scheme estimates the FHD in a distributed manner at the MAC layer and it works well even if one of them is in the Internet.

In the second experiment, over a chain topology of fixed length (we took hop length of 10), we measured the aggregate goodput of TCP NewReno flows by varying the number flows between the edge

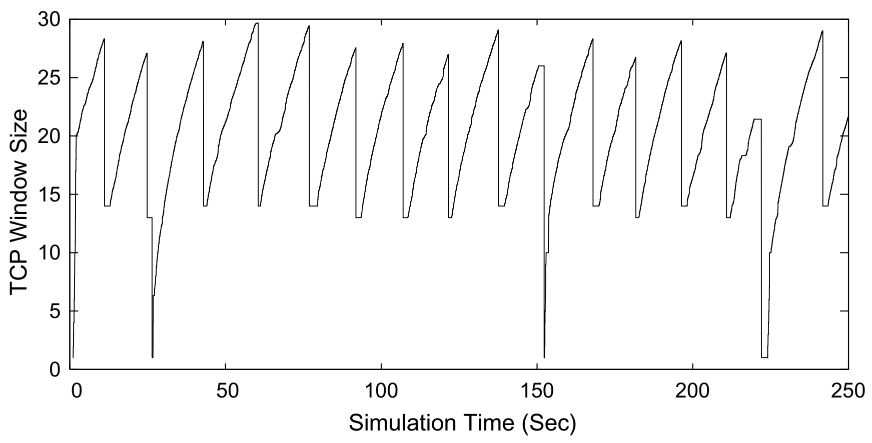

Fig. 22. Progression of window growth with time of a TCP NewReno flow over a multihop chain.

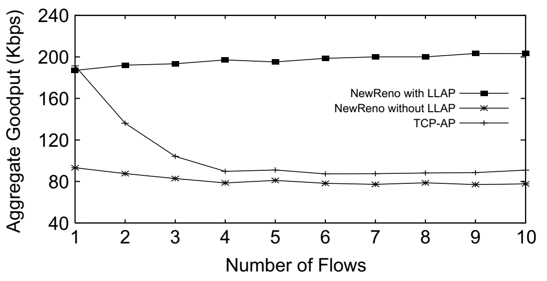

Fig. 24. Aggregate goodput of TCP NewReno flows with and without LLAP.

nodes. As discussed in Section 1, spreading the transmission of packets at the transport layer does not spread the transmission of packets at the MAC layer. Hence, TCP-AP does not perform well with multiple flows. As the number of TCP flows increases between the same source and destination, the aggregate goodput decreases drastically. But TCP NewReno with LLAP achieves maximum aggregate goodput as seen in Fig. 24. We analyzed the fairness between these flows using Jain's fairness index. We found that fairness index decreases with increase in number of flows either with or without LLAP. But the reduction in the fairness index is minimal in the case of LLAP. When compared with TCP-AP, there is a reduction in fairness index, but this reduction in fairness compensates the huge increase in aggregate goodput when compared with TCP NewReno and TCP-AP. The measured Jain's fairness index is shown in Fig. 25.

In the third experiment, we took a 10-hop chain network and chose the MAC data rate of each node randomly. We measured the goodput of TCP NewReno with and without LLAP for 30 different random assignment of data rates to the nodes. For each random assignment of data rate, we measured the goodput of TCP NewReno for 30 different runs.

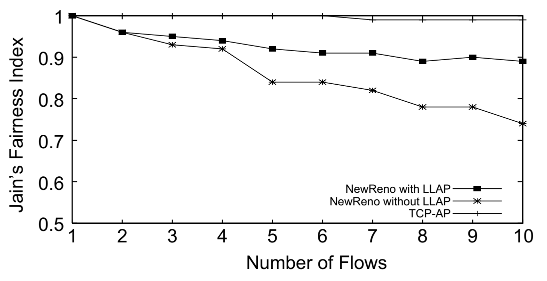

Fig. 25. Jain's fairness index of TCP NewReno flows with and without LLAP.

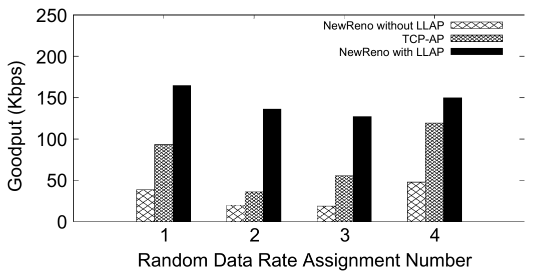

Fig. 26. Average goodput with random MAC data rate at nodes.

We found that TCP NewReno over LLAP performs better compared to TCP NewReno without LLAP and TCP-AP in most of the random assignments. This is because the LLAP estimates the bottleneck link's transmission delay at the ingress node and paces the packets with FHD of the bottleneck node. In Fig. 26, we show the goodput of TCP NewReno with and without LLAP and TCP-AP for four different random assignment of data rates.

7.2. Grid topology

To study the performance of LLAP in general network scenarios, we took a 10×10 grid topology with all edge nodes of the grid as ingress and egress nodes and all other nodes as core nodes. We measured the performance of both UDP and TCP traffic over this grid topology shown in Fig. 27. For UDP

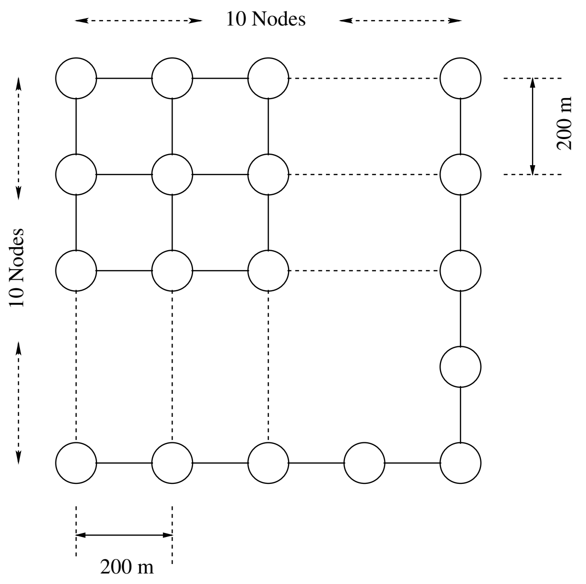

Fig. 27. 10×10 grid topology with 200 m between adjacent nodes.

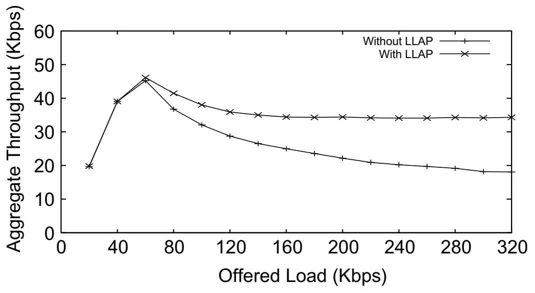

Fig. 28. Aggregate throughput for increasing offered load with 2 Mbps channel and packet size of 50 bytes.

traffic we considered 20 CBR flows and measured the aggregate throughput at the receiver for different data rates. We took packet size of 50 bytes (typical voice applications generate small UDP packets) and 1000 bytes. We setup flows from one edge node to other edge node in the opposite side. This is to make sure that hop length of all flows is more than four. We randomly picked five ingress and five egress nodes in each side of the grid and generated traffic between the ingress and egress nodes selected on opposite sides of the grid. We ran the simulations for 30 such different flow patterns and computed the average aggregate throughput. We ran experiments with and without LLAP by varying the data rate of CBR flows. For packet size of 50 bytes, we varied the data rate of all CBR traffic from 1 Kbps to 16 Kbps. Fig. 28 shows the aggregate throughput for varying offered load. Without LLAP, as the offered load increases above 60 Kbps (3 Kbps of each CBR flow), the aggregate throughput decreases drastically. But, with LLAP the reduction in throughput is less. Thus, the overall network throughput increases with LLAP. For the case of packet size of 1000 bytes, we used the channel data rate as 11 Mbps and varied the data rate of each

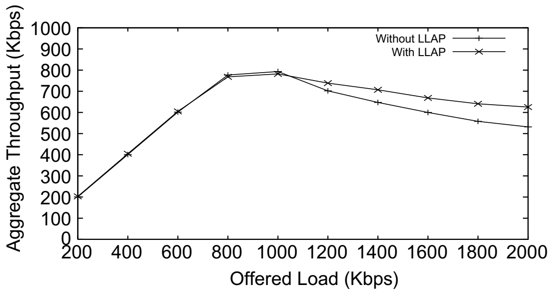

Fig. 29. Aggregate throughput for increasing offered load with 11 Mbps channel and packet size of 1000 bytes.

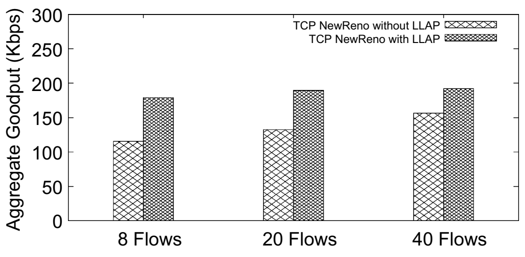

Fig. 30. Aggregate goodput achieved for different number of flows.

CBR flow from 10 Kbps to 100 Kbps. Even in this case, we found that the aggregate throughput improves with increasing offered load as shown in Fig. 29.

To study the performance of TCP traffic over grid topology, we varied the number of TCP flows from each side of the grid as 2, 5, and 10. The ingress and egress nodes are selected at random in each side and TCP connections are established between the ingress and egress nodes on the opposite sides of the grid. For the randomly generated TCP flows, the aggregate goodput is computed by averaging over 30 different such flow patterns. We measured the aggregate goodput of the TCP flows with and without LLAP which is shown in Fig. 30. The aggregate goodput of the TCP flows improves with LLAP.

8. Conclusion

In this paper, we analyzed the performance of multihop wireless backbone network in two-tier wireless mesh networks. If the ingress nodes inject data more than what the backbone network can handle, the performance of the backbone network reduces drastically. We proposed a Link Layer Adaptive Pacing (LLAP) scheme to control the flow of packets into the backbone network at all ingress nodes in order to effectively utilize the capacity of the backbone network. The available capacity of each path between the ingress and egress pair of nodes is found using a distributed way of estimating the four hop transmission time on a multihop path. This method does not result in any additional transmission overhead to the network. Through ns-2 simulations, we found that our LLAP improves the performance of both reliable and unreliable transport layer protocols such as TCP and UDP, respectively in different network scenarios.

Acknowledgements

The authors wish to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] I.F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (4) (2005) 445–487.
[2] K. Chen, Y. Xue, S.H. Shah, K. Nahrstedt, Understanding bandwidth-delay product in mobile ad hoc networks, Computer Communications Journal 27 (10) (2004) 923–934.
[3] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP performance, IEEE Transactions on Mobile Computing 4 (2) (2005) 209–221.
[4] G. Hallond, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, Wireless Networks 4 (2/3) (2002) 275–288.
[5] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, A feedback based scheme for improving TCP performance in ad hoc wireless networks, IEEE Personal Communications Magazine 8 (1) (2001) 34–39.
[6] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, R. Sivakumar, ATP: a reliable transport protocol for ad-hoc networks, in: Proceedings of ACM MOBIHOC, June 2003, pp. 64–75.
[7] H. Zhai, X. Chen, Y. Fang, Rate-based transport control for mobile ad hoc networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, March 2005, pp. 2264–2269.
[8] B.V. Ramana, B.S. Manoj, C.S.R. Murthy, AR-TCP: a loss-aware adaptive rate based TCP for ad hoc wireless networks, Journal of High Speed Networks 15 (1) (2006) 53–72.
[9] S. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, in: Proceedings of IEEE INFOCOM, March 2003, pp. 1744–1753.
[10] K. Xu, M. Gerla, L. Qi, Y. Shu, Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED, in: Proceedings of ACM MOBICOM, September 2003, pp. 16–28.
[11] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R.H. Katz, A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking 5 (6) (1997) 756–769.
[12] S.M. Eirakabawy, A. Klemm, C. Lindemann, TCP with adaptive pacing for multihop wireless networks, in: Proceedings of ACM MOBIHOC, May 2005, pp. 288–299.
[13] S.M. Elrakabawy, A. Klemm, C. Lindemann, Gateway adaptive pacing for TCP across multihop wireless networks and the internet, in: Proceedings of ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, 2006, pp. 173–182.
[14] C.S.R. Murthy, B.S. Manoj, Ad Hoc Wireless Networks – Architectures and Protocols, Prentice Hall PTR, 2004.
[15] J. Robinson, E. Knightly, A Performance study of deployment factors in wireless mesh networks, in: Proceedings of IEEE INFOCOM, May 2007, pp. 2054–2062.
[16] J. Camp, J. Robinson, C. Steger, E. Knightly, Measurement driven deployment of a two-tier urban mesh access network, in: Proceedings of ACM MOBISYS, June 2006, pp. 96–106.
[17] C. Perkins, E. Royar, S. Das, Ad Hoc On-Demand Distance Vector (AODV) Routing, IETF RFC 3561, 2003.

A. Antony Franklin received his B.E. degree in Electronics and Communication Engineering from Madurai Kamaraj University, India, in 2000 and M.E. degree in Computer Science and Engineering from Anna University, India, in 2002. After that he worked in K.L.N. College of Engineering, Madurai, India, as a lecturer for three years in the Department of Computer Science and Engineering where he was involved in research on VLSI design and embedded system development. He is currently pursuing his Ph.D., in the Department of Computer Science and Engineering, Indian Institute of Technology Madras, India. His research interests include wireless ad hoc, sensor, and mesh networks. His current research focus is towards developing protocols and architectures for wireless mesh networks.

C. Siva Ram Murthy earned his B.Tech. degree in Electronics and Communications Engineering from Regional Engineering College (now National Institute of Technology), Warangal, India, in 1982, M.Tech. degree in Computer Engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 1984, and Ph.D., degree in Computer Science from the Indian Institute of Science, Bangalore, India, in 1988.

He has been with the Department of Computer Science and Engineering at IIT Madras, since 1988, where he is currently a Professor. He has held visiting positions at the German National Research Centre for Information Technology (GMD), Germany, University of Stuttgart, Germany, University of Freiburg, Germany, Max Planck Institute for Software Systems, Germany, Swiss Federal Institute of Technology (EPFL), Switzerland, and University of Washington, Seattle, USA.

He is the co-author of the textbooks *Parallel Computers: Architecture and Programming*, (Prentice-Hall of India, New Delhi, India), *New Parallel Algorithms for Direct Solution of Linear Equations*, (John Wiley & Sons, Inc., NY, USA), *Resource Management in Real-time Systems and Networks*, (MIT Press, Cambridge, MA, USA), *WDM Optical Networks: Concepts, Design, and Algorithms*, (Prentice Hall, Upper Saddle River, NJ, USA), and *Ad Hoc Wireless Networks: Architectures and Protocols*, (Prentice Hall, Upper Saddle River, NJ, USA). His research interests include parallel and distributed computing, real-time systems, lightwave networks, and wireless networks. He has published more than 125 international journal and 125 international conference papers in these areas.

He is a recipient of Best Ph.D. Thesis Award from the Indian Institute of Science, Indian National Science Academy (INSA) Medal for Young Scientists, and Dr. Vikram Sarabhai Research Award. He is a co-recipient of Best Paper Awards from the 5th

IEEE International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), the 6th and 11th IEEE Annual International Conference on High Performance Computing (HiPC), and the 14th IEEE International Conference on Networks (ICON). He is a Fellow of the Indian National Academy of Engineering, an Associate Editor of IEEE Transactions on Computers, and a Subject Area Editor of Journal of Parallel and Distributed Computing.