Multicasting With Localized Control In Wireless Ad-Hoc Networks

Jun Peng, *IEEE Member*, Biplab Sikdar, *IEEE Member*, Liang Cheng, *IEEE Member*

Abstract— **This paper investigates how to support multicasting in wireless ad hoc networks without throttling the dominant unicast flows. Unicast flows are usually congestion-controlled with protocols like TCP. However, there are no such protocols for multicast flows in wireless ad hoc networks and multicast flows can therefore cause severe congestion and throttle TCP-like flows in these environments. Based on a cross-layer approach, this paper proposes a completely-localized scheme to prevent multicast flows from causing severe congestion and the associated deleterious effects on other flows in wireless ad hoc networks. The proposed scheme combines the layered multicast concept with the routing-based congestion avoidance idea to reduce the aggregated rate of multicast flows when they use excessive bandwidth on a wireless link. Our analysis and extensive simulations show that the fully-localized scheme proposed in this paper is effective in ensuring the fairness of bandwidth sharing between multicast and unicast flows in wireless ad hoc networks.**

Keywords: Multicast, Wireless Ad Hoc Networks, Mobile Ad Hoc Networks, Localized Control

I. Introduction

Wireless ad hoc networks such as mobile ad hoc networks and wireless mesh networks are self-organized and usually without centralized control. Protocols in such networks are also required to be distributed for robustness and scalability. If a distributed protocol only relies on local information and local actions for fulfilling its functionality, then the protocol is also localized. In the sense of using only local resources, a localized protocol is usually efficient and scalable, which are the basic characteristics required for protocols in wireless ad hoc networks. This paper investigates localized mechanisms to support multicasting in wireless ad hoc networks without throttling unicast flows.

One of the basic elements required for multicasting in wireless ad hoc networks is multicast routing. Similar to a unicast packet, a multicast packet relies on the underlying routing protocol to reach its destinations. Existing routing protocols for multicasting in wireless ad hoc networks such as MAODV [1] and ODMRP[2], like unicast routing protocols, only set up routing information in nodes but do not have other controls over flows, such as congestion control. Although there have been many efforts at creating multicast transport protocols, no mature protocols have emerged yet, even for wireline networks, due to the difficulties posed by the multiple-receiver characteristic of multicasting. Existing multicast congestion control schemes largely fall into two categories: single-rate and multi-rate. Single-rate schemes such as [3], [4], [5], [6], [7] have poor performance in intra-session fairness (i.e., the fairness between receivers in the same session) as compared to multi-rate schemes. This is because in single-rate schemes, the transmission rate of a multicast session is usually decided by the receiver with the lowest path rate. Multi-rate schemes such as [8], [9], [10], [11], [12], [13], [14], [15] can achieve much better intra-session fairness because with these schemes, each receiver has some freedom to choose a rate appropriate for itself.

Existing multi-rate protocols, such as Receiver-driven Layered Multicast (RLM), cannot ensure fairness with TCP [16], [17], even in wireline networks. To address the unfairness issue of RLM [16], RLC [9] adopts two strategies, namely synchronization points and probe bursts, to coordinate receivers in obtaining knowledge on path conditions. It has been shown, however, that both RLM and RLC have inherent limits in achieving fairness with TCP flows [17]. Another interesting protocol in this category is the Wave and Equation Based Rate Control (WEBRC) protocol [13]. WEBRC relies on round trip times and waves to approach fairness with TCP. It needs further investigation on how impreciseness of round trip times would impact the performance of the protocol. In addition, WEBRC introduces considerable (but lower than RLM and RLC) control traffic overhead in adjusting receivers' layers, which is a significant disadvantage in wireless ad hoc networks. There are also overlay-network multicast schemes such as [18] in the literature, whose impact on the network fairness needs further investigation.

In wireless ad hoc networks, the unfairness situation becomes more severe with existing multicast congestion control protocols. First, the wireless links of wireless ad hoc networks are prone to errors. High link-error rates usually interfere with congestion control. Second, wireless links can have much longer link delays than wireline links due to the difficulties of medium access control in wireless environments. Long link delays adversely impact multicast congestion control due to increased delays for control messages. Third, wireless links usually have low effective link bandwidth. Therefore, competition for bandwidth is more severe. For these reasons, it will be extremely difficult, if not impossible, to create an end-to-end congestion control scheme that is effective and TCP-friendly for multicasting in wireless ad hoc networks.

Instead of relying on end-to-end congestion control schemes, this paper proposes a fully localized scheme in the network layer to support multicasting in wireless ad hoc networks while maintaining fairness with unicast flows. The proposed scheme integrates layered multicast with routing-based congestion avoidance to achieve its rate control over multicast flows. The proposed scheme is fully localized. Each node acts based on locally-collected information and no additional interaction between nodes is required for the rate-control operations over multicast flows, except those required for standard multicasting service. With the proposed scheme, a multicast source encodes its signal into several layers of various priorities [8]. The source then sends each layer to a separate multicast group. Receivers of the multicast source subscribe to these multicast groups, and packets for all or some of these groups flow into the receivers. At the same time, each intermediate node in the wireless ad hoc network monitors its wireless link. When the link starts becoming congested, the node cuts the aggregated rate of multicast flows if the multicast flows are using excessive bandwidth on the link. The local rate-cut on multicast flows is possible because each multicast flow has multiple layers and layer-priority information is embedded in the multicast addresses of these layers. Our analysis and detailed simulation results show that the proposed scheme enables multicasting in wireless ad hoc networks and provides unicast flows their fair share of bandwidth.

The rest of the paper is organized as follows. Section II presents the proposed scheme in detail, and then Section III analyzes the proposed scheme for fairness between unicast flows and multicast flows. The results of scheme evaluation based on extensive simulations are shown in Section IV. Related work and discussion appear in Section V. Finally, Section VI concludes the paper.

II. The Proposed Scheme

A. Scheme Assumptions and Basic Approach

The proposed scheme imposes no direct control over any unicast flows and assumes that each unicast flow is controlled by TCP or a similar protocol without the assistance of active queue management. A multicast source encodes its signal into multiple layers and then sends each layer to a separate multicast group[1]. After a source chooses its layer size, it does not change the size for the rest of the multicast session. The intended receivers of the multicast source try to subscribe to all these groups. Packets for all or some of these groups then flow into each individual receiver.

All the multicast packets traversing a link and originating from the same multicast source are called a "multicast flow" on the link in this paper. When multicast flows traverse the wireless link of a node, the node observes the output queue of its link at regular intervals $I_{CheckQu}$ (congestion events are always immediately reported irrespective of the observation intervals). When the number of packets in the queue, N_{QuPkt}, exceeds a threshold, $QuThresh_2$, some layers of multicast flows will be blocked from entering the link. However, when N_{QuPkt} is below another threshold, $QuThresh_1$, for a specified amount of time, some blocked layers of multicast flows will be released and allowed to traverse the link. In other cases, there are usually no layer adjustments over multicast flows. This is the basic approach of the proposed scheme and the rest of this section discusses in details the various aspects of the scheme.

Before introducing the scheme details, we first explain what happens when the data in a layer is blocked. If the multicast application, such as video multicasting, can tolerate losses, the data in a blocked layer is usually not recovered and receivers thus obtain information at a lower resolution, such as lower quality of received video. On the other hand, if the multicast application requires total reliability in data delivery, the source needs to use a technique such as the digital fountain technique introduced in [10]. In such a case, a receiver needs to receive enough packets before it can decode and obtain all the data from the source. A blocked layer therefore introduces latency in data delivery in this case.

Finally, we need to clarify that the proposed scheme is mainly designed to effectively relieve congestion at bottlenecks with multicast traffic. In addition, it is designed to maintain general fairness in bandwidth sharing among the competing flows at a bottleneck. In particular, when a given bottleneck limits the rates of all flows that pass through it, all flows receive similar bandwidth shares at the bottleneck. However, in scenarios where the rates of some flows may be restricted by other bottlenecks, these flows may receive a lower share of the bandwidth at the given bottleneck. Finally, since the proposed scheme is not centralized, we do not expect it to meet the requirements of fairness criteria other than the one we consider.

B. Retrieving Flow Information

With the proposed scheme, a node collects flow information about the traffic traversing its link to assist its congestion-control operation. Specifically, the number of TCP flows ($N_{TcpFlow}$), number of multicast flows ($N_{MctFlow}$), number of layers of each multicast flow ($N^i_{LiveLayer},\ 0 < i \leq N_{MctFlow}$), average per-flow rate of TCP flows (R_{TcpAvg}) and the average per-flow rate of multicast flows (R_{MctAvg}) are the information retrieved. The source and destination addresses and port numbers are used to identify the TCP flow or the layer of a specific multicast flow to which a packet belongs. The number of layers ($N^i_{LiveLayer}$) that the i^{th} multicast flow has on a link is obtained by observing the number of different multicast addresses used by the packets of the flow traversing the link. For the average per-flow rates, the proposed scheme does not need the absolute values.

[1]Multicast groups are not the only means to distinguish layers, although they are used in the proposed scheme.

Instead, the average per-flow rates of TCP flows and of multicast flows are measured in the following way. The total number of TCP packets (N_{TcpPkt}) and the total number of multicast packets (N_{MctPkt}) traversing the link in specified intervals (T_{count}) are counted[2], and then divided, respectively, by the number of TCP flows and the number of multicast flows traversing the link. The results are the measured average per-flow rates of TCP flows and multicast flows: $R_{TcpAvg} = N_{TcpPkt}/N_{TcpFlow}$; $R_{MctAvg} = N_{MctPkt}/N_{MctFlow}$.

The flow information retrieved from the traffic traversing a link is the basis for the operation of the proposed scheme on the link. The proposed scheme assumes that the number of layers that a multicast flow possesses on a link can reflect its relative data rate among the multicast flows traversing the same link. Particularly, a multicast flow with more layers has a higher data rate than a multicast flow with less layers on the same link[3].

C. Embedding Layer Priority Information

In some applications such as streaming media, different layers of a multicast flow have different priorities. In general, a lower layer has higher priority and the packets of such a layer are more useful for a receiver to get its wanted information. For example, when a piece of video is encoded into L layers, usually the packets of the m^{th} layer can be useful in decoding only if the packets from all lower layers $(1, 2, 3, ..., m-1)$ are available. In such a case, when a layer of a flow needs to be blocked on a link, the layer with the lowest priority should be blocked. In the proposed scheme, the layer priority information in a multicast flow is embedded in the multicast addresses used by the multicast flow.

With the proposed scheme, when a multicast source applies for multicast addresses, it is assigned a block of addresses. The multicast source allocates lower addresses to its higher-priority layers and higher addresses to its lower-priority layers. Therefore, a node in the network can determine the priority of a layer in a multicast flow traversing its link by comparing the address of the layer with the addresses of other layers in the same multicast flow.

This implicitly embedded priority information is important for the proposed scheme because it eliminates the need for defining new fields in the packet header for carrying packet priority information. If new fields had to be added to packet headers, the information in these fields would have to be retrieved separately upon the arrival of a packet. In such a case, extra cost would be introduced.

D. Flow Initialization and Receiver's Roles

At the beginning of a multicast session, each receiver adds layers gradually by subscribing to those multicast groups that are employed by the multicast source for delivering its layered signal. After adding a layer, a receiver waits for a specified period of time before adding another layer. If the added layer is not blocked in the network and its packets are flowing into the receiver, the receiver adds another layer. This process continues until an empty layer is obtained by the receiver. An empty layer is a layer whose packets are not flowing into the receiver because of being blocked in the network.

Apart from the initialization process, receivers also play a small role at other times for layer adjustment in the proposed scheme. Particularly, each receiver is responsible for maintaining a single empty layer. If a receiver has more than one empty layer, it drops all but the lowest one. On the other hand, if the packets of the maintained empty layer of a receiver start to flow into the receiver, the receiver adds another layer after a specified period of time. There are two purposes for a receiver to maintain a single empty layer. One is to prevent multiple unused layers from flowing into the network or a section of the network. The other is to maintain quick response to link state changes, since an empty layer may be released quickly when free bandwidth becomes available at a bottleneck.

E. Layer Block and Layer Release

Layer-block is the modification of the multicast routing table to stop a layer from entering a congested link (packets are actually dropped before entering the queue). On the contrary, layer-release is the modification of the routing table to allow a blocked layer to traverse a link. When a layer-block is necessary on a link, the multicast flow with the maximum number of layers on the link is selected to be blocked a layer (if there are ties, one of them is selected randomly). Within this selected flow, the layer with the lowest priority is blocked. This selection process is expressed formally below.

$$\begin{gathered}\text{If } N^m_{LiveLayer} \geq N^n_{LiveLayer} \text{ for } \forall n \in \\ (0, N_{MctFlow}], \text{ choose the } m^{th} \text{ flow.} \\ \text{Then, if } P_{L^m_i} \leq P_{L^m_j} \text{ for } \forall j \in \\ (0, N^m_{LiveLayer}], \text{ block the } i^{th} \text{ layer.}\end{gathered}$$

where $P_{L^m_i}$ denotes the priority of the i^{th} layer of the m^{th} session. On the other hand, when a layer-release is needed, the multicast flow with the minimum number of layers is selected to release a layer[4]. Within this selected flow, the layer with the highest priority among the blocked layers is released. This process of selecting a layer to release is expressed formally below.

$$\begin{gathered}\text{If } N^m_{LiveLayer} \leq N^n_{LiveLayer} \text{ for } \forall n \in \\ (0, N_{MctFlow}], \text{ choose the } m^{th} \text{ flow.} \\ \text{Then, if } P_{L^m_i} \geq P_{L^m_j} \text{ for } \forall j \in \\ (N^m_{LiveLayer}, N^m_{layer}], \text{ release the } i^{th} \text{ layer.}\end{gathered}$$

[2]In addition, the counting process is reset and restarted whenever congestion occurs at the bottleneck.

[3]Note that this assumption may not be satisfied in reality, which may then lower the fairness obtained by the proposed scheme.

[4]If this flow has no empty layer to release, then the flow with the next minimum number of layers is selected. This selection process continues until a flow is selected or there is no flow left.

These layer selection procedures ensure that competing multicast flows share the bandwidth available to them fairly. They also ensure that priorities among the layers of the same multicast flow are taken care of. The following subsections present how the proposed scheme ensures that multicast flows as a whole do not consume excessive bandwidth on a link.

F. Adjusting Multicast Layers

This subsection introduces the most important part of the proposed scheme, which is the adjustment of the number of multicast layers (N_{layer}) traversing a link that is congested, where $N_{layer} = \sum_{i=1}^{N_{MctFlow}} N^i_{LiveLayer}$. A link becomes congested when the total rate of the traversing traffic becomes greater than the effective bandwidth of the link. The effective bandwidth of a wireless link is determined by the physical bandwidth of the link and the radio competitions from other nodes. The effective bandwidth of a wireless link is smaller when there are more competing nodes.

The proposed scheme blocks or releases multicast layers on a link according to the state of the associated node's output queue. A queue is classified into three phases in the proposed scheme, which are phase-1, phase-2 and phase-3. The phase of a queue is determined by the number of packets in the queue, N_{QuPkt}, and two specified thresholds, $QuThresh_1$ and $QuThresh_2$. The thresholds depend on two scheme parameters, $F_{QuThresh1}$ and $F_{QuThresh2}$: $QuThresh_1 = QuSize \times F_{QuThresh1}$; $QuThresh_2 = QuSize \times F_{QuThresh2}$, where $QuSize$ is the size of the queue. The phases of a queue are classified as:

$$\text{QuPhase} = \begin{cases} 1 & \text{if } N_{QuPkt} \leq QuThresh_1 \\ 2 & \text{if } QuThresh_1 < N_{QuPkt} \leq QuThresh_2 \\ 3 & \text{if } N_{QuPkt} > QuThresh_2 \end{cases}$$

A phase-1 queue indicates that free bandwidth may be available on the link while a phase-3 queue usually implies existing or impending congestion on the link. To fully utilize available bandwidth and effectively deal with congestion, different actions are required in different queue phases.

Phase-1 is designed for multicast flows to claim free bandwidth on the link. However, when the queue is in phase-1, a multicast layer is not necessarily released. The reason is that a phase-1 queue does not necessarily ensure that free bandwidth is available on the link. TCP flows have their famous sawtooth-like rate fluctuations due to their Additive Increase and Multiplicative Decrease (AIMD) congestion control mechanism. Upon congestion, a TCP flow cuts its rate multiplicatively to relax the congested link, then it additively builds up its rate to probe for free bandwidth on its path. Therefore, a phase-1 queue may just indicate that the TCP flows traversing the link cut their rates multiplicatively a moment ago and are building up their rates now. In such a case, no multicast layer should be released. Otherwise, TCP flows may be deprived of their share of bandwidth. However, if the queue stays in phase 1 for a relatively long time, then it is almost certain that free bandwidth is available on the link. Therefore, in the proposed scheme, a multicast layer is released on a link only if the link output queue has stayed in phase-1 for a specified amount of time ($T_{Phase1Thresh}$).

When the link output queue is in phase-2, usually no action is taken for multicast flows. A kind of balance is achieved when the queue stays in phase-2 most of the time. However, this does not necessarily mean that good fairness is also achieved between TCP flows and multicast flows. For example, if a multicast flow leaves, TCP flows may increase their rates and keep the queue in phase-2 most of the time (i.e., to multicast flows, there is no free bandwidth on the link). In such a case, the multicast flows left behind by the departing multicast flow can not get a share of the bandwidth released by the departing multicast flow. This is because without additional precautions, no multicast layer will be released unless the queue stays in phase-1 for a duration longer than $T_{Phase1Thresh}$. To avoid this kind of unfairness situation, the average per-flow rate of TCP flows (R_{TcpAvg}) and the average per-flow rate of multicast flows (R_{MctAvg}) are checked in phase-2. If R_{MctAvg} is less than R_{TcpAvg}, a multicast layer is released. Otherwise, no action is taken.

The purpose of phase-3 is to detect congestion on the link. When the link output queue is in phase-3, a multicast layer may be blocked instantly. This is because if the queue stays in phase-3 for a significant amount of time, then TCP flows may be throttled from frequent packet losses. However, there is a special situation to consider if a multicast layer is blocked instantly each time when the queue enters phase-3. The rate fluctuations of TCP flows may cause the queue to visit phase-3 from time to time. In such a case, multicast layers may frequently be blocked in phase-3 and then be released in phase-2 after the TCP flows back off due to congestion. The number of layers of multicast flows traversing the link thus fluctuates in such a case. To avoid this problem, the average per-flow rate of TCP flows (R_{TcpAvg}) and the average per-flow rate of multicast flows (R_{MctAvg}) are also checked in phase-3. Only if R_{MctAvg} is higher than R_{TcpAvg}, a multicast layer is blocked in phase-3. This procedure stabilizes multicast traffic and ensures fairness among competing flows.

G. Scheme Adaptation

This subsection introduces several scheme adaptation procedures for stabilizing multicast traffic on a link. The main cause of instability of multicast traffic on a link is that the rate adjustment units for multicast flows are layers and a multicast layer usually has a significant size.

It is possible that a layer adjustment on a link may cause queue phase and layer fluctuation due to the significant size (or equivalently, rate) of a multicast layer. For example, a layer release in phase 2 may force the

queue to go to phase-3 because of the significantly increased traffic on the link. However, in phase-3, the layer may be blocked instantly, which may cause the queue to fall back to phase-2. In phase-2, the layer may be released again. These queue movements therefore cause phase and layer fluctuation. In general, the number of multicast layers on a link should be kept as stable as possible for good bandwidth utilization. The proposed scheme adopts several procedures to alleviate layer and queue phase fluctuation.

First, when multicast traffic needs to be increased continuously on a link, the rate of increase is decreased each time after a layer is released. Particularly, the observation time ($T_{Observe}$) (i.e., the time of observation before a layer adjustment is made) for the next layer release is increased by a factor $F_{SlowDown}$ ($F_{SlowDown} > 1$): $T_{Observe} \longleftarrow T_{Observe} \times F_{SlowDown}$. Second, when a layer is blocked right after a layer is released, the observation time ($T_{Observe}$) for the next layer release is increased by another factor $F_{BackOff}$ ($F_{BackOff} > 1$): $T_{Observe} \longleftarrow T_{Observe} \times F_{BackOff}$. $T_{Observe}$ is reset to its initial value after the queue has been in phase-1 for a specified period of time. Third, a layer is blocked in phase 3 only if the average per-flow rate of multicast flows is greater than the average per-flow rate of TCP flows by a ratio threshold (RT_{Block}): $(R_{MctAvg} - R_{TcpAvg})/R_{TcpAvg} > RT_{Block}$. The last procedure specifically reduces phase alternations between phase 2 and phase 3.

H. Scheme Parameter Settings

Our experiments show that the scheme is not sensitive to its parameter settings if some basic rules are followed in assigning the values. For the queue threshold factors, the higher one, $F_{QuThresh2}$, should be near to 1.0, while the lower one, $F_{QuThresh1}$, should be close to 0.0. This is because phase 3 is to detect potential congestion conditions while phase 1 is to detect free bandwidth on the link. For the two back-off parameters, $F_{SlowDown}$ and $F_{BackOff}$, the latter should be larger than the former, since a layer block right after a layer release indicates a high probability of another layer block if a layer is released again in a short time. The choice of RT_{Block} should be one to a couple of tens percent for avoiding layer alternation between phase 2 and phase 3. $I_{CheckQu}$ determines how frequently the queue is checked for its status. Too long intervals mean slow response, while too short intervals introduce excessive cost. In general, higher link speed requires shorter check intervals because higher-rate traffic change the queue status quickly. $T_{Observe}$ is for stabilizing layer adjustment. Since this parameter is adjusted dynamically by the scheme, the initial value should be set to a small value. $T_{Phase1Thresh}$ is the time threshold in phase 1 for releasing a layer. It should be set to a relatively high value for ensuring that free bandwidth is indeed available when a layer is released. Finally, T_{count}, the interval to measure the average per-flow rates of TCP flows and multicast flows, should be set to a relatively large value as compared to the expected RTTs of the flows in the network. We implemented the protocol in our simulations with a set of parameter values that follow these above rules. Our results show that these same parameter setting works well in all our simulation scenarios.

III. Convergence to Fairness

In this section we analytically prove that the proposed scheme shares bandwidth fairly with an arbitrary number of competing TCP flows. We use Jain's fairness index [19] to quantify a notion of fairness. Consider a set of n flows where the window (or equivalently the rate) of the i-th flow is given by $x_i(t)$. The Jain's fairness index $F(t)$ at time t is then given by

$$F(t) = \frac{\left(\sum_{i=1}^{n} x_i(t)\right)^2}{n \sum_{i=1}^{n} x_i(t)^2} \tag{1}$$

which attains the value of 1 when the allocation is totally fair ($x_1(t) = x_2(t) = \cdots = x_n(t)$).

We consider a scenario where a multicast flow shares a bottleneck with an arbitrary number, denoted by n, of TCP flows. For ease of analysis, we assume that all the TCP flows have the same RTT. The service capacity of the bottleneck is C bits per second. The size of a multicast layer is denoted by g and we assume that $g \gg 1$ since the size of a multicast layer is expected to be quite large. Note that $g \geq 2$ suffices for our proof. The action of the two protocols involved can be described in terms of their response to congestion or the absence thereof, as dictated by their increase ($\mathcal{I}$) and decrease ($\mathcal{D}$) policies. Let each application of the increase or decrease policy be separated by R time units (we assume that these are the same for both TCP and the multicast session for ease of analysis). The behavior of the multicast flow with rate $x_m(t)$ at time t, with the corresponding rate of the i^{th} TCP flow being $x_i(t)$, $1 \leq i \leq n$, is then given by

$$\mathcal{I}: \quad x_m(t+R) \longleftarrow \begin{cases} x_m(t) + g & \text{if } x_m(t) < x_{tcp}(t) \\ x_m(t) & \text{otherwise} \end{cases} \tag{2}$$

$$\mathcal{D}: \quad x_m(t+R) \longleftarrow \begin{cases} x_m(t) & \text{if } x_m(t) < x_{tcp}(t) \\ x_m(t) - g & \text{otherwise} \end{cases} \tag{3}$$

where $x_{tcp}(t) = \sum_{i=1}^{n} x_i(t)/n$ is the average rate of TCP flows. Similarly, the behavior of the TCP flows can be modeled as

$$\mathcal{I}: \quad x_i(t+R) \longleftarrow x_i(t) + \alpha \tag{4}$$

$$\mathcal{D}: \quad x_i(t+R) \longleftarrow x_i(t) - \beta x_i(t) \tag{5}$$

with standard values of α and β being 1 and $1/2$ respectively.

For the purposes of this proof, we consider the operation of the network in terms of "rounds". The length of a round is not fixed. At the beginning of each round, the rates of the TCP flows and the multicast flow are increased as specified by Equations (2) and (4) above. If

the queue at the bottleneck does not overflow as a result of this increase, the current round ends and the next round begins with another application of the increase policies. On the other hand, if the application of the increase policy overflows the queue, then the decrease policies as specified in Equations (3) and (5) are applied to the flows. If this does not alleviate the overflow in the queue, multiple applications of the decrease policy may follow. The round ends when the overflow at the bottleneck is alleviated. The next round again begins with the application of the increase policies since each round begins with a stable system.

In the rest of this section we drop the time index t from the notation and use: x_m, x_{tcp} and x_i to denote the rates of the multicast flow, the average rate of the TCP flows and the rate of the i-th TCP flow, $1 \leq i \leq n$, respectively, at the beginning of the current round. Note that $x_{tcp} = \frac{\sum_{i=1}^{n} x_i}{n}$. We use x'_m, x'_{tcp} and x'_i to denote these rates after the increase policy has been applied once the round begins. Also, since each round begins with a stable system, we have $x_m + \sum x_i \leq C$. Finally, for ease of notation, we drop the limits from the summations and all sums are from $i = 1, \cdots, n$ unless specifically mentioned.

To prove that the system of flows converges to fairness, we use the following result, which is Theorem 4, page 334, of [20] and is reproduced below for convenience:

Result 1: (Theorem 4, page 334, [20]): m flows with windows (or rates) $y_1, y_2, \cdots, y_m$ converge to fairness if the following condition is satisfied over any small period of time

$$\sum_{i=1}^{m} y_i^2 \sum_{i=1}^{m} \Delta y_i \geq \sum_{i=1}^{m} y_i \sum_{i=1}^{m} y_i \Delta y_i \tag{6}$$

where Δy_i represents the change in the window (or rate) of the i-th flow over the period of time and at least one of the $\mathcal{I}$ or $\mathcal{D}$ policies results in a strict inequality.

In addition to Result 1 we have the following results which will be used in the proof of our claim of convergence to fairness.

Result 2: If $x_{tcp} \leq x_m + g$ then $\frac{x_{tcp}}{2} \leq x_m$.

Proof: The rate of the multicast flow, x_m is an integral multiple of g. Say $x_m = rg$, for some $r \geq 1$. Thus $x_{tcp} \leq x_m + g = (r+1)g$ implies

$$\frac{x_{tcp}}{2} \leq \frac{r+1}{2} g \leq rg = x_m$$

for $r \geq 1$. Note that if x_{tcp} is strictly less than $x_m + g$, i.e., $x_{tcp} < x_m + g$, then $\frac{x_{tcp}}{2} < x_m$. ■

Result 3: If $x_m < x_{tcp}$ then $x_m^2 + \sum x_i^2 > x_m + \sum x_i$.

Proof: We first note that since $x_m \geq g \gg 1$ (or $x_m \geq 2$), $x_m^2 > x_m$. Also, since $x_{tcp} > x_m$ we have $\frac{\sum x_i}{n} > x_m \geq g \gg 1$ and thus $\sum x_i \gg n$. Let $\sum x_i = \alpha$. Then

$$\begin{aligned} \sum x_i = \alpha \gg n \quad &\Rightarrow \quad (\sum x_i)^2 = \alpha^2 \gg n^2 \\ &\Rightarrow \quad n \sum x_i^2 \geq (\sum x_i)^2 = \alpha^2 \gg n^2 \quad (7) \\ &\Rightarrow \quad \sum x_i^2 \geq \frac{\alpha^2}{n} > \alpha = \sum x_i \end{aligned}$$

where in Equation (7) we have used Cauchy's inequality ($n \sum x_i^2 \geq (\sum x_i)^2$). Thus we have $x_m^2 + \sum x_i^2 > x_m + \sum x_i$. ■

We now formally state the claim of convergence to fairness for our scheme and prove it.

Claim 1: A system comprising of a multicast flow sharing a bottleneck with an arbitrary number of TCP flows converges to fairness.

Proof: We break the proof in two parts by considering the two possibilities: (1) $x_m < x_{tcp}$ and (2) $x_m \geq x_{tcp}$, each of which has four sub-cases. We show that for the cases that involve only applications of the increase policy, Eqn. (6) is satisfied with a strict inequality.

Case 1: $x_m < x_{tcp}$: In this case, since the rate of the multicast flow is lower than that of the TCP flows and the queue is not overfull when the round begins, both the multicast and TCP flows increase their rates, as per Equations (2) and (4). Thus we have $x'_m = x_m + g$, $x'_{tcp} = x_{tcp} + 1$ and $x'_i = x_i + 1$. Based on this, we have four subcases:

subcase 1.a $x'_m \leq x'_{tcp}$: no overflow ($x_m + g + \sum x_i + n \leq C$)
subcase 1.b $x'_m \leq x'_{tcp}$: overflow ($x_m + g + \sum x_i + n > C$)
subcase 1.c $x'_m > x'_{tcp}$: no overflow ($(n+1)(x_m + g) \leq C$)
subcase 1.d $x'_m > x'_{tcp}$: overflow ($(n+1)(x_m + g) > C$)

Here we prove subcase 1.a. The other cases can be proved in a similar way.

subcase 1.a ($x'_m \leq x'_{tcp}$ and $x_m + g + \sum x_i + n \leq C$): Since we have $x'_m = x_m + g$, $x'_{tcp} = x_{tcp} + 1$ and $x'_i = x_i + 1$, the total rate of traffic flow into the bottleneck queue is $x_m + g + \sum x_i + n$. Since this not more than the queue's service capacity C, the queue does not overflow. Also, $\Delta x_m = (x_m + g) - x_m = g$ and $\Delta x_i = (x_i + 1) - x_i = 1$. To show that Equation (6) holds in this case we need to demonstrate

$$\begin{aligned} & (x_m^2 + \sum x_i^2)(g + \sum 1) \geq (x_m + \sum x_i) \\ & \times (g x_m + \sum 1 \cdot x_i) \\ \text{s.i.,} \quad & g \sum x_i^2 + n x_m^2 + n \sum x_i^2 \geq \\ & g x_m \sum x_i + x_m \sum x_i + (\sum x_i)^2 \\ \text{s.i.,} \quad & g \sum x_i^2 + n x_m^2 \geq \\ & g x_m \sum x_i + x_m \sum x_i \end{aligned} \tag{8}$$

where s.i. represents "which is satisfied if". To prove the relation above, we note that for any x_i, $1 \leq i \leq n$

$$(x_i - x_m)^2 \geq 0 \quad \Rightarrow \quad x_i^2 + x_m^2 \geq 2x_m x_i \tag{9}$$

Adding over all i, i.e. over all the TCP flows and multiplying by g, we have

$$\begin{aligned} & g\sum x_i^2 + ngx_m^2 \geq 2gx_m \sum x_i \\ \Rightarrow \quad & g\sum x_i^2 + nx_m^2 \geq (g+1)x_m \sum x_i + \\ & \left[(g-1)x_m \sum x_i - n(g-1)x_m^2\right] \\ \Rightarrow \quad & g\sum x_i^2 + nx_m^2 > (g+1)x_m \sum x_i \end{aligned}$$

since $x_m < x_{tcp} \Rightarrow nx_m < \sum x_i \Rightarrow n(g-1)x_m^2 < (g-1)x_m \sum x_i \Rightarrow 0 < [(g-1)x_m \sum x_i - n(g-1)x_m^2]$. This shows that Eqn. (8) holds and consequently Eqn. (6) holds for this subcase with a strict inequality.

Case 2: $x_m \geq x_{tcp}$**:** In this case, since the rate of the multicast flow is greater than that of the TCP flows and the queue is not overfull when the round begins, only the TCP flows increase their rates, as per Equations (2) and (4). Thus we have $x'_m = x_m$, $x'_{tcp} = x_{tcp} + 1$ and $x'_i = x_i + 1$. Based on this, we have four subcases:

subcase 2.a $x'_m \geq x'_{tcp}$: no overflow ($x_m + \sum x_i + n \leq C$)
subcase 2.b $x'_m \geq x'_{tcp}$: overflow ($x_m + \sum x_i + n > C$)
subcase 2.c $x'_m < x'_{tcp}$: no overflow ($x_m + g + \sum x_i + ng \leq C$)
subcase 2.d $x'_m < x'_{tcp}$: overflow ($x_m + g + \sum x_i + ng > C$)

Here we prove subcase 2.a. The other cases can be proved in a similar way.

subcase 2.a ($x'_m \geq x'_{tcp}$ and $x_m + \sum x_i + n \leq C$): At the end of the round, the change in the rates of the flows is $\Delta x_m = x_m - x_m = 0$ and $\Delta x_i = (x_i + 1) - x_i = 1$. Also, $\Delta x_m = 0$, $\Delta x_i = 1$ and $x'_m \geq x'_{tcp}$ implies that $x_m > x_{tcp}$. To show Equation (6) holds in this case, we then need to show

$$\begin{aligned} & (x_m^2 + \sum x_i^2)\left(0 + \sum 1\right) \geq \\ & (x_m + \sum x_i)\left(0 \cdot x_m + \sum x_i \cdot 1\right) \\ \text{s.i.,} \quad & nx_m^2 + n\sum x_i^2 \geq x_m \sum x_i + (\sum x_i)^2 \end{aligned} \tag{10}$$

From Cauchy's inequality, $n\sum x_i^2 \geq (\sum x_i)^2$. Also, $x_m > x_{tcp} \Rightarrow nx_m > \sum x_i \Rightarrow nx_m^2 > x_m \sum x_i$. Thus Equation (10) and consequently Equation (6) holds in this case with a strict inequality.

■

This proves that irrespective of the current rate of the flows and the current level of fairness in the system, there exists a time interval after which the fairness of the system becomes greater than the current level. While the level of fairness may oscillate over a period of time, the system converges towards fairness over time, even though the oscillations persist.

IV. Scheme Evaluation

This section presents the evaluation results for the proposed scheme. Our evaluation was done with extensive simulations on Network Simulator 2 [21]. The values of main scheme parameters used in our simulations are given in Table I. Our simulation configurations focus on generating a single typical bottleneck in a wireless ad hoc network that bears both multicast and unicast flows. The proposed scheme in this paper adopts a fully localized strategy for controlling multicast flows. With the proposed scheme, congestion in the network is dealt with locally and cooperation between intermediate nodes is not required, except those demanded for standard multicasting. It is therefore general enough to investigate how the proposed scheme performs on a single typical bottleneck in a wireless ad hoc network.

In the wireless ad hoc network configured in our simulations, the propagation model is two-ray ground; the MAC protocol is IEEE 802.11; the ad-hoc routing protocol is DSR[5]; and the link queue size is 50 packets. The wireless ad hoc network is in an area of 670m by 670m and the transmission distance of each node is 100m. Two nodes at a distance of 100m form a shared wireless bottleneck. The senders of the competing flows are randomly placed within 100m around one node of the shared bottleneck and the receivers of the competing flows are randomly placed within 100m around the other node of the shared bottleneck. It is ensured though that all the senders must be more than 100m away from all the receivers so that all competing flows traverse the shared bottleneck, which has a radio bandwidth of 1Mbps. There are 5 competing flows in our simulations, 2 multicast flows and 3 TCP flows, and there is a maximum of 14 nodes in the simulations. Each multicast flow has 15 layers and the size of each layer is 10Kb/s[6].

In addition, a multicast source in our simulations can either be a Constant Bit Rate (CBR) source or Variable Bit Rate (VBR) source. In general, CBR multicast sources are easier to deal with than VBR multicast sources because the latter introduce traffic fluctuation on links. VBR multicast sources pose challenges for existing multicast congestion control schemes [16], [17]. However, our simulation results show that the proposed scheme can tolerate significant fluctuation of multicast traffic.

A. CBR Multicast Sources

This subsection shows the simulation results for CBR multicast sources. We used three scenarios to test the proposed scheme. In the first scenario, there is no node mobility; in the second scenario, one secession moves away from the bottleneck; in the third scenario, all nodes follow random waypoint movement.

[5]We did not simulate the details of an existing multicast routing protocol because the performance of the proposed protocol at a given bottleneck does not depend on the details of such a protocol.

[6]Existing CODECs may need redesign or modification to generate equal-size layers in reality.

TABLE I
MAIN SCHEME PARAMETER VALUES

$F_{QuThresh1}$	$F_{QuThresh2}$	$F_{SlowDown}$	$F_{BackOff}$	RT_{Block}	$I_{CheckQu}$	$T_{Observe}$	$T_{Phase1Thresh}$	T_{count}
0.0	0.8	1.5	4.0	0.2	0.02	0.1	1.0	0.02

1) Static Network Scenario: We first consider scenarios in which nodes have no mobility. In this scenario, all flows start at the 100th second of the simulation (for leaving some establishment time for the wireless ad hoc network) and stop at the 1500th second. The simulation results are shown in Fig. 1 and Fig. 2. Fig. 1 shows the number of layers and the throughput of each multicast flow, while Fig. 2 presents the throughput of each individual TCP flow and the average per-flow throughput of TCP flows. As shown in these two figures, each flow, multicast or unicast, gets a throughput close to 4 KBytes/s. Moreover, after the initial adjustment, the number of layers of each multicast flow is fairly stable[7].

For reference, in this section we show the simulation results for RLM [8], which is one of the most well-known multi-layer multicast congestion control schemes in the literature. Due to limited space, we only show the results for the least challenging case from RLM's perspective, in which the multicast traffic is CBR traffic and all nodes are static. RLM usually uses exponentially increasing sizes for layers, which is for multiplicatively reducing traffic rate in case of congestion. In our simulations, RLM uses 5 layers and the size of each layer is 10, 20, 40, 80, and 160 kb/s, respectively, from layer 1 to layer 5. The simulation results are shown in Fig. 3 and Fig. 4. By comparing Fig. 3 and Fig. 4 with Fig. 1 and Fig. 2, respectively, we can find that the proposed scheme achieves higher stability in both numbers of layers for multicast sessions and throughput for all sessions. This demonstrates that the congestion on the bottleneck is better controlled with the proposed scheme. In addition, these figures show that the proposed scheme achieves better fairness as compared to RLM; the Jain's fairness index, on average, is 0.89 and 0.73 for the proposed scheme and RLM, respectively.

Another thing that we examined in this scenario is what happens when the TCP flows do not have the same RTT. In the examination, the sender of the first TCP flow moves out of the 100m range of the shared bottleneck so that the TCP flow has four instead of three hops on its path. The simulation results are shown in Fig. 5 and Fig. 6. As shown in Fig. 6, the first TCP flow obtains less bandwidth at the bottleneck than the other TCP flows due to its longer RTT. Howerver, the average TCP flow rate is still similar to that shown in Fig. 2.

Effect of Interfering Flows: We now consider the presence of other nodes in the vicinity of the bottleneck whose flows do not necessarily flow through the nodes forming the bottleneck. However, their transmissions cause interference and contend for the same channel as the bottleneck thereby reducing the effective bandwidth at the bottleneck and causing stochastic variations in the channel availability. To observe the impact of such flows on the performance of our proposed protocol, we introduced an interfering flow with a constant rate of 40 kb/s (the rate of 4 multicast layers) in the neighborhood of the bottleneck. This was achieved by adding two more wireless nodes within the transmission radius of the nodes forming the bottleneck link. The CBR flow transfers packets between these nodes and in the process, competes for the channel (or bandwidth) with the flows traversing the bottleneck. The simulation results for this case are shown in Figs. 7 and 8. As shown in these figures, the bandwidth available to each competing flow on average is reduced when the effective bandwidth of the bottleneck is reduced. However, the fairness is still maintained on the bottleneck.

2) One Session Moving-away Scenario: In this scenario we show how the proposed scheme behaves when one competing flow leaves the shared bottleneck. We first show the simulation results for the case in which a TCP source is moving away from the common bottleneck link. The simulation results are shown in Fig. 9 and Fig. 10. As shown in Fig. 10, when the source of TCP1 starts moving away at the 500th second of the simulation, the throughput of TCP1 decreases and finally the TCP flow disappears from the bottleneck link at about the 700th second. As shown in Fig. 9 and Fig. 10, the bandwidth released by the departing TCP flow is fairly shared among the four flows that are left behind by the departing TCP flow.

3) Random Node Movement Scenario: This scenario shows the performance of the proposed scheme when nodes follow random waypoint movement. As introduced earlier, there is a single wireless link connecting two parts of the wireless ad hoc network in our simulations (the ad hoc network is 500 by 500 square meters). Source nodes are located in one part, while destination nodes are located in the other part. This configuration provides the shared bottleneck as the ideal observation point for scheme behaviors. Both source nodes and destination nodes follow random waypoint movement in their parts of the network. The maximum node speed is five meters per second, while the average pause time is five seconds. The simulation results for this scenario are shown in Fig. 11 and Fig. 12.

As shown in Fig. 11, the two multicast flows do not have a stable number of layers anymore as random movement is introduced to nodes in the network. This is because when nodes move, link quality and actual

[7]There is a one-layer difference between the two multicast flows in some cases. One-layer difference between multicast flows is possible with layered multicast congestion control because the units of rate adjustment for multicast flows are layers.

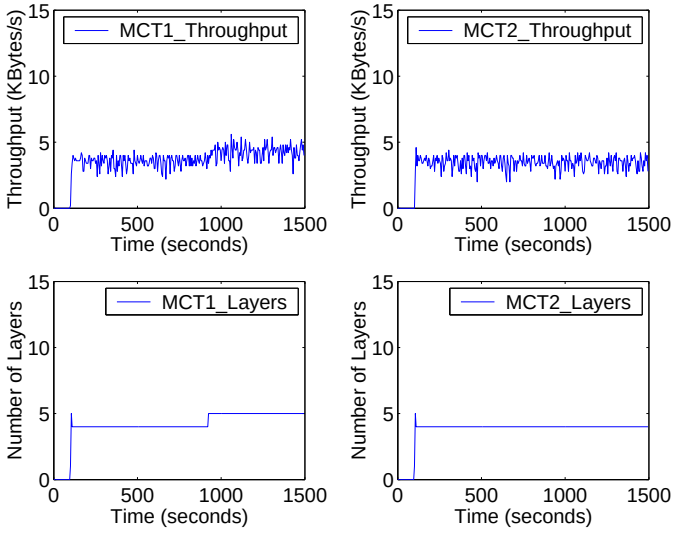

Fig. 1. Individual Multicast Throughput and Number of Layers

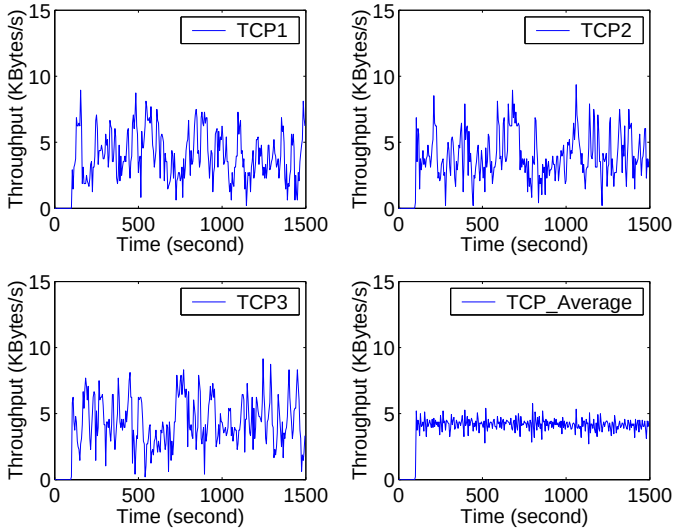

Fig. 2. Individual TCP Throughput and Average Per-flow TCP Throughput

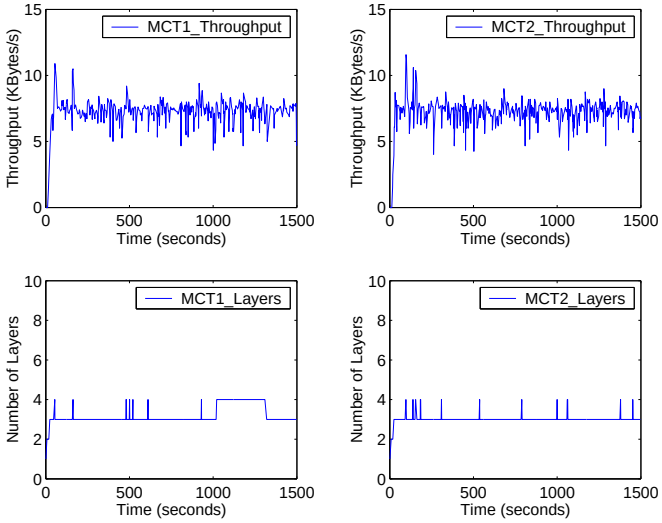

Fig. 3. Individual Multicast Throughput and Number of Layers (RLM Case)

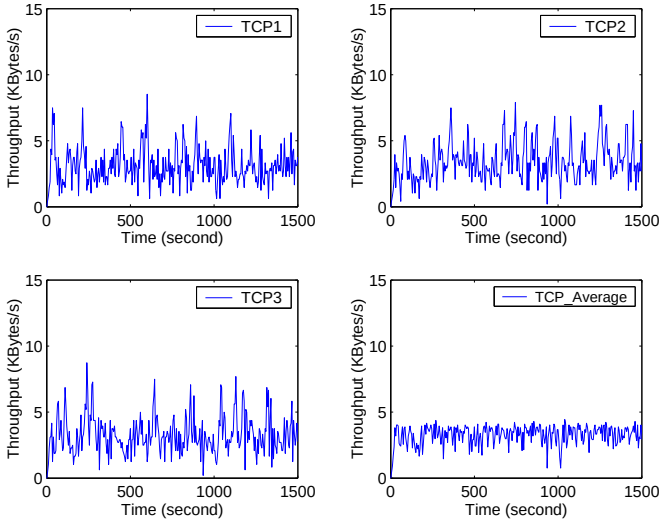

Fig. 4. Individual TCP Throughput and Average Per-flow TCP Throughput (RLM Case)

capacities change in the network. In addition, existing paths may even disappear in the network. However, on average, the two multicast flows have similar numbers of layers when both of them have traffic on the shared bottleneck (the second multicast flow loses traffic on the bottleneck a couple of times due to node movement). The behavior of the three TCP flows is shown in Fig. 12 and we note that due to the random movement, they do not have similar shares of the bandwidth on the bottleneck. This is because the TCP flows here have different delays and link quality on their paths. However, multicast flows still show fairness with TCP flows in this random movement scenario; TCP flows, on average, obtain a fair share of bandwidth on the bottleneck, as shown in Fig. 12.

B. VBR Multicast Sources

VBR multicast sources pose more serious challenges for existing multicast congestion control schemes [16], [17]. This subsection shows how the proposed scheme behaves with VBR multicast sources. For generating the VBR multicast sources, we used the exponential traffic model in NS-2. The simulated VBR traffic has exponentially distributed burst and idle times. We set the average idle time to 100ms and varied the average burst time of the multicast traffic to test the proposed scheme with different traffic burstiness, while the average flow rate is kept the same. Due to limited space, we only show the

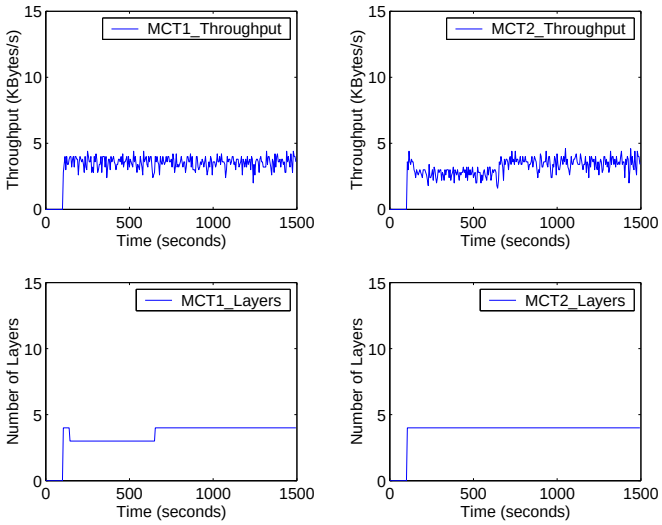

Fig. 5. Individual Multicast Throughput and Number of Layers (Various TCP RTTs)

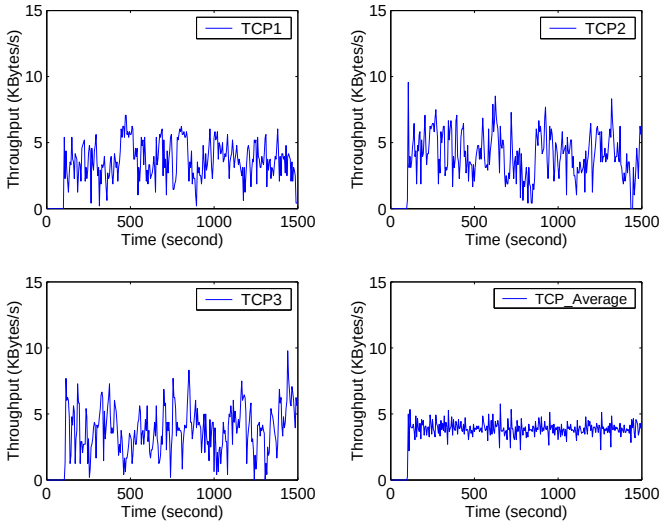

Fig. 6. Individual TCP Throughput and Average Per-flow TCP Throughput (Various TCP RTTs)

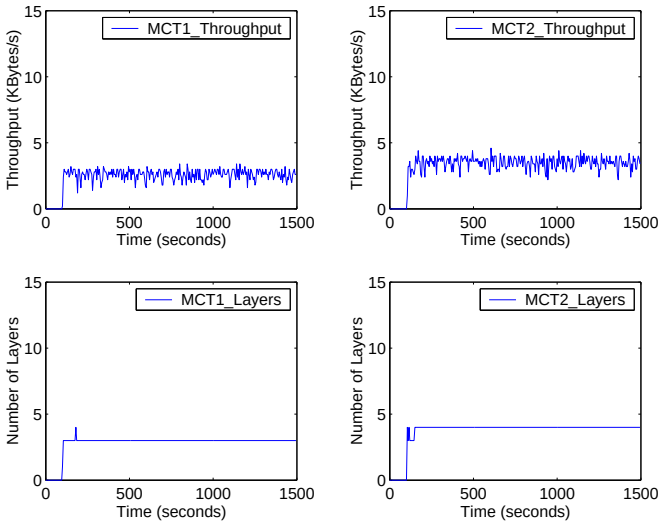

Fig. 7. Individual Multicast Throughput and Number of Layers (Reduced Effective Bandwidth Case)

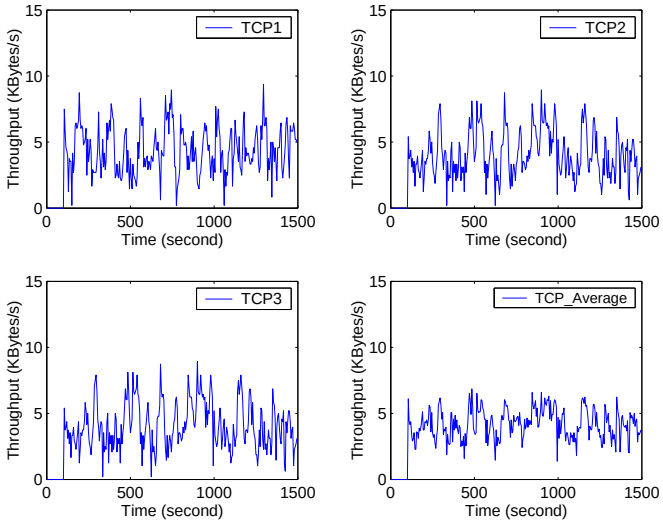

Fig. 8. Individual TCP Throughput and Average Per-flow TCP Throughput (Reduced Effective Bandwidth Case)

results for the static network case, with five competing flows. The results for the cases with node mobility follow the same trends as in the results for CBR sources.

The simulation results are shown in Fig. 13 and and Fig. 14 for the case in which the average burst time of the multicast traffic is 1000ms. As shown in these two figures, both multicast flows still have a fairly stable number of layers. In addition, fairness is still achieved among the competing flows. However, as compared with the simulation results shown in Fig. 1 and Fig. 2, the actual throughput on the common wireless link is reduced after the introduction of the VBR multicast traffic. This is because VBR traffic can cause more severe congestion on the bottleneck link than CBR traffic of the same average rate.

To increase the burstiness of the multicast traffic, we decreased the average burst time to 500ms (the burst rate is actually increased for keeping the average flow rate unchanged). The simulation results are shown in Fig. 15 and Fig. 16. As shown in these two figures, both multicast flows still maintain the stability of their number of layers. In addition, fairness is also maintained among the competing flows. To further increase the burstiness of the multicast traffic, we decreased the average burst time to 200ms. The simulation results are shown in Fig. 17 and Fig. 18. As shown in these two figures, both multicast flows show significant variation in their numbers of layers. However, the fairness among the competing flows is still maintained in such a case.

The simulation results in this subsection show that

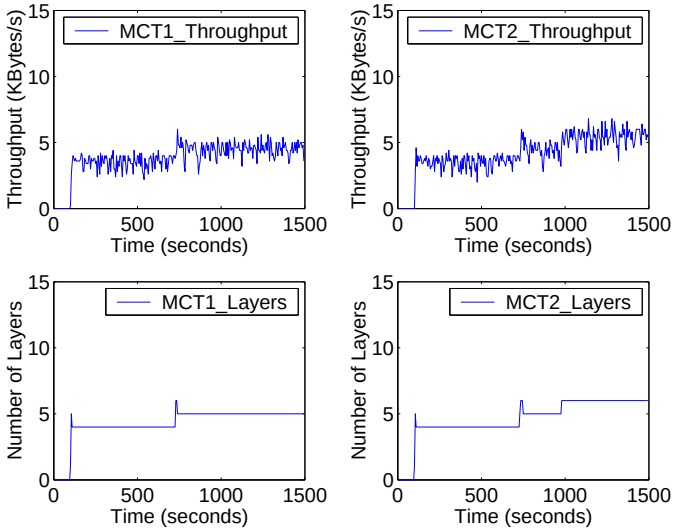

Fig. 9. Individual Multicast Throughput and Number of Layers (A TCP flow moves away from the common bottleneck link)

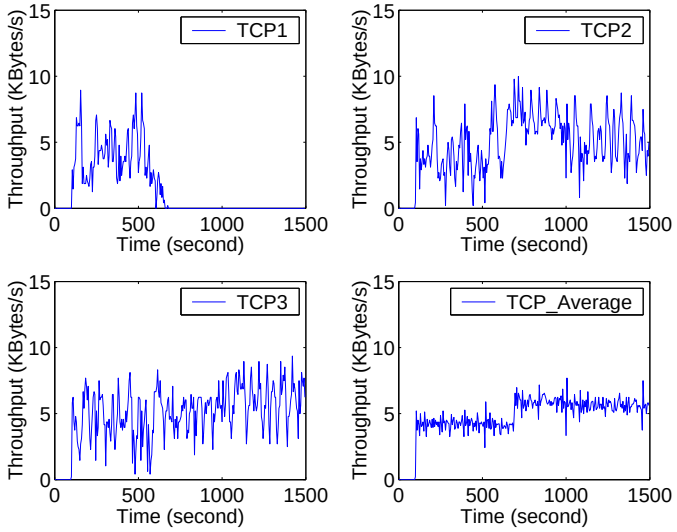

Fig. 10. Individual TCP Throughput and Average Per-flow TCP Throughput (A TCP flow moves away from the common bottleneck link)

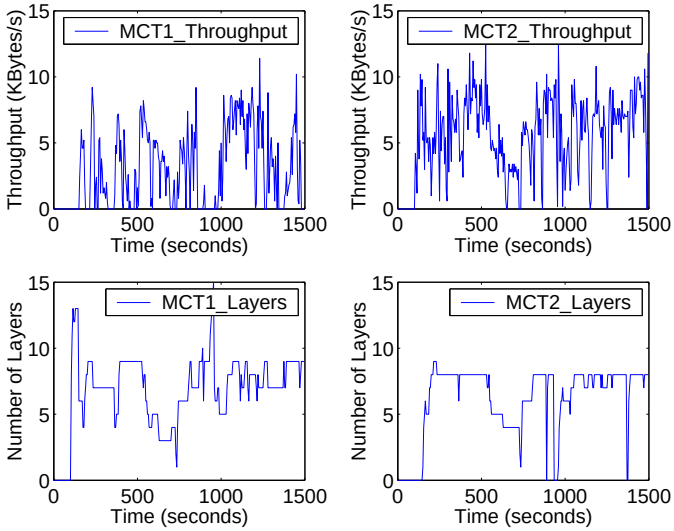

Fig. 11. Individual Multicast Throughput and Number of Layers (random waypoint movement)

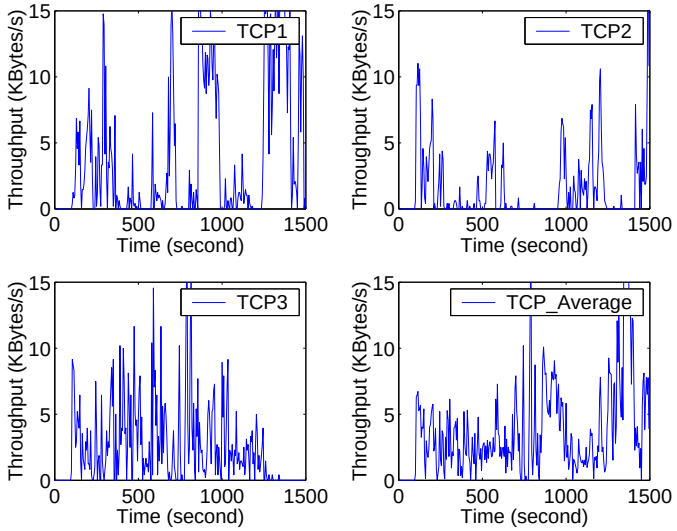

Fig. 12. Individual TCP Throughput and Average Per-flow TCP Throughput (random waypoint movement)

the proposed scheme is able to tolerate significant fluctuation of multicast traffic. With limited fluctuation in multicast traffic, the scheme still maintains the stability of the number of layers for each multicast flow while fairness is also maintained among the competing flows. As multicast traffic fluctuates severely, the multicast flows may have an unstable number of layers. However, the fairness between competing flows is still maintained. The instability of multicast layers in such a case is practically unavoidable with any layered congestion control schemes because the severely fluctuating multicast traffic causes frequent flash congestion on the bottleneck link and the congestion control scheme thus has to adjust multicast layers correspondingly.

V. Related Work and Discussion

Besides the purely end-to-end multicast congestion control schemes introduced in the first section of this paper, there are also some network-assisted congestion control schemes proposed for layered multicast in the literature, such as [22] [23] [24]. Sarkar et al. propose a new scheduling policy in [22] where flows are served in a round robin manner in each link based on feedback from immediate downstream links. When a flow is sampled, it is served only if at least one of the immediate downstream links is not congested and able to accept packets from the flow. Bajaj et al. analyze and compare

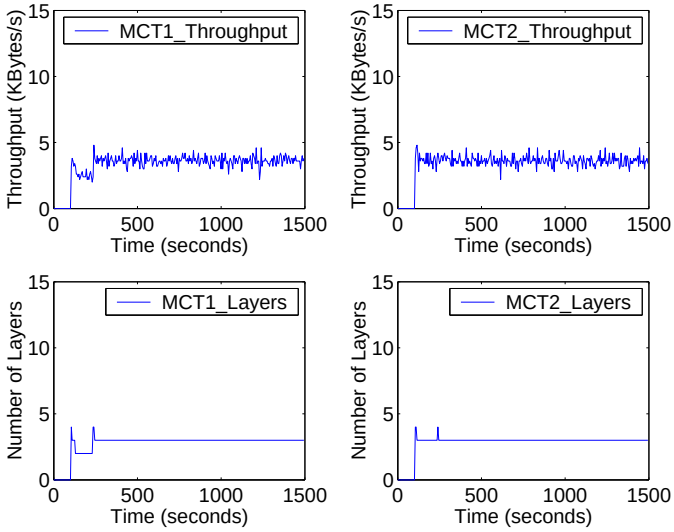

Fig. 13. Individual Multicast Throughput and Number of Layers (VBR multicast traffic with average burst time of 1000ms)

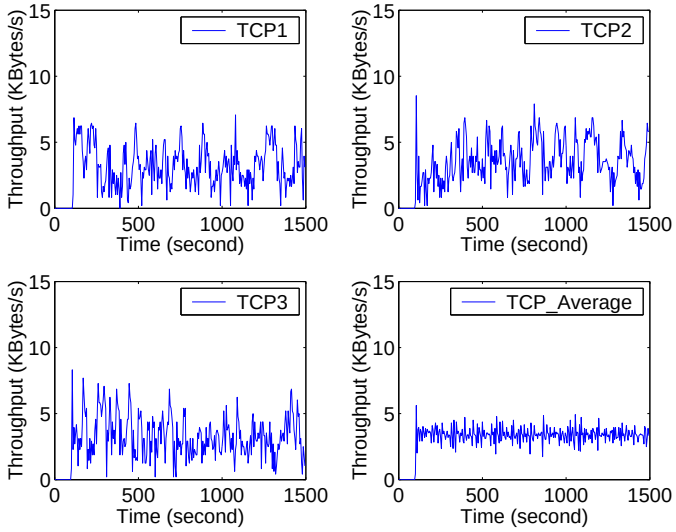

Fig. 14. Individual TCP Throughput and Average Per-flow TCP Throughput (VBR multicast traffic with average burst time of 1000ms)

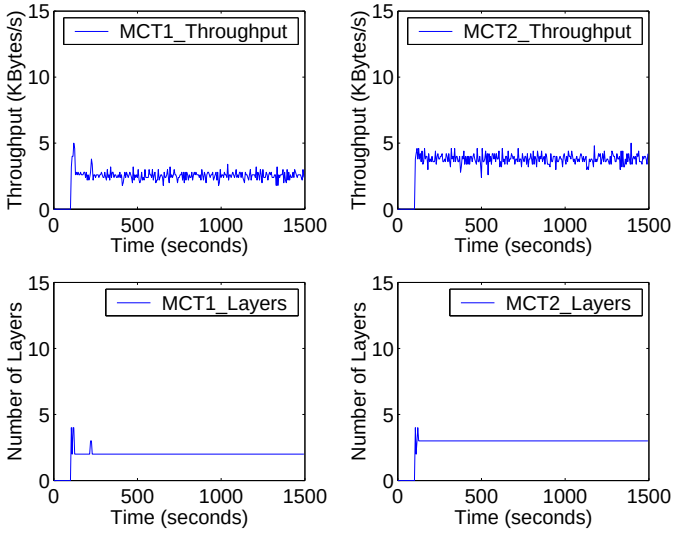

Fig. 15. Individual Multicast Throughput and Number of Layers (VBR multicast traffic with average burst time of 500ms)

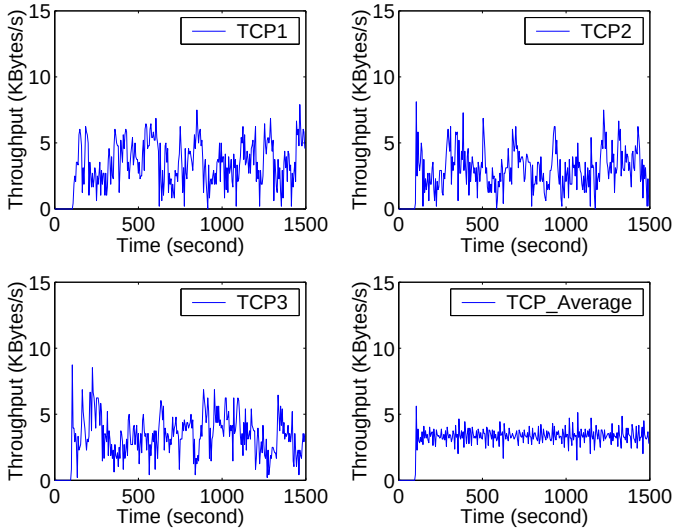

Fig. 16. Individual TCP Throughput and Average Per-flow TCP Throughput (VBR multicast traffic with average burst time of 500ms)

in [23] two different dropping policies for layered video, namely uniform dropping and priority dropping and show that priority dropping performs better in general than uniform dropping if implementation complexity is not an issue. To reduce the complexity of priority dropping, the layered multicast scheme proposed in [24] considers only two levels of priority among layers. Receivers map one layer as low priority and all other layers as high priority. Routers then confine the losses of the flow on the bottleneck to the low priority layer. Therefore, the low priority layer serves as something like a "buffer" to absorb losses.

None of these network-assisted schemes is fully localized and require additional interactions and cooperations either between routers or between routers and receivers, or both. For example, with the scheme in [22], a router requires constant feedback from all immediate downstream routers, while the priority dropping employed in [23] and [24] requires priority information feedback to related routers. In contrast, the scheme proposed in this paper is fully localized and does not require explicit interactions between nodes specifically for congestion control.

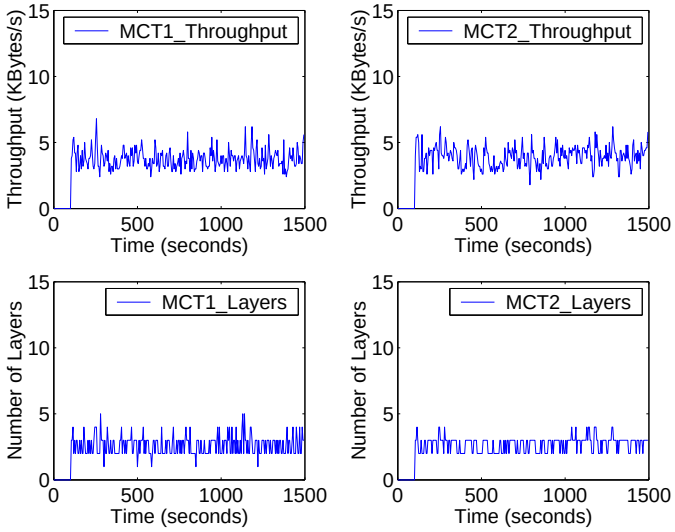

Fig. 17. Individual Multicast Throughput and Number of Layers (VBR multicast traffic with average burst time of 200ms)

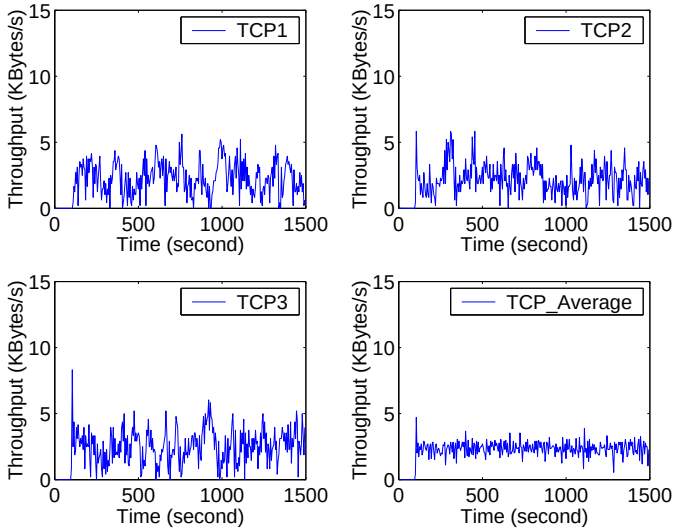

Fig. 18. Individual TCP Throughput and Average Per-flow TCP Throughput (VBR multicast traffic with average burst time of 200ms)

VI. Conclusion

This paper presents a fully localized scheme to support multicasting in wireless ad hoc networks such as mobile ad hoc networks and ensures unicast flows their shares of bandwidth on a link. Existing multicast congestion control schemes are usually designed for wireline networks. Meanwhile, they can not ensure fairness with TCP. Wireless ad hoc networks pose more serious challenges for those schemes because wireless ad hoc networks have limited bandwidth, significant channel access delays, and high link error rates. Instead of relying on end-to-end schemes for supporting multicasting in wireless ad hoc networks, this paper, based on a cross-layer approach, proposes a fully localized scheme that protects unicast flows from being throttled by multicast flows in wireless and mobile ad hoc networks. The proposed scheme combines layered multicast with routing-based congestion control to achieve its goals in a fully localized way. Our analysis and extensive simulations show that the proposed scheme is effective in facilitating multicasting in wireless ad hoc networks while preventing unicast flows from being throttled.

References

[1] E. M. Royer and C. E. Perkins, "Multicast operation of the ad hoc on-demand distance vector routing protocol," in *Proceedings of MobiCom*, Seattle, WA, Aug. 1999, pp. 207–218.

[2] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," in *Proceedings of IEEE WCNC*, New Orleans, LA, Sep. 1999, pp. 1298–1304.

[3] I. Rhee, N. Balaguru, and G. Rouskas, "MTCP: scalable TCP-like congestion control for reliable multicast," in *Proc of IEEE INFOCOM.*, March 1999, pp. 1265–1273.

[4] L. Rizzo, "PGMCC: A TCP-friendly single-rate multicast congestion control scheme," in *Proc of ACM SIGCOMM.*, Aug 2000.

[5] J. Widmer and M. Handley, "Extending equation-based congestion control to multicast applications," in *Proc of ACM SIGCOMM.*, San Diego, CA, Aug. 2001.

[6] S. Shi and M. Waldvogel, "A rate-based end-to-end multicast congestion control protocol," in *Proceedings of ISCC*, July 2000.

[7] S. Bhattacharyya, D. Towsley, and J. Kurose, "The loss path multiplicity problem in multicast congestion control," in *Proc of IEEE INFOCOM.*, 1999, pp. 856–863.

[8] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc of ACM SIGCOMM.*, Aug 1996, pp. 117–130.

[9] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," in *Proc of IEEE INFOCOM.*, San Franciso, March 1998, pp. 996–1003 Vol. 3.

[10] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc of ACM SIGCOMM.*, Vancouver, Canada, Sept 1998, pp. 56–67.

[11] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver, "FLID-DL: Congestion control for layered multicast," in *Proceedings of NGC 2000*, November 2000, pp. 71–81.

[12] W. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 373–386, February 2001.

[13] M. Luby and V. Goyal, "Wave and equation based rate control using multicast round trip time," in *Proc of ACM SIGCOMM.*, Pittsburgh, Pennsylvania, USA., Aug. 2002, pp. 191–204.

[14] J. Byers, M. Luby, and M. Mitzenmacher, "Fine-Grained layered multicast," in *Proc of IEEE INFOCOM.*, April 2001.

[15] G. Kwon and J. Byers, "Smooth multirate multicast congestion control," in *Proc of IEEE INFOCOM.*, March 2003.

[16] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. Sreenan, "Stability and fairness issues in layered multicast," in *Proceedings of the NOSSDAV*, June 1999, pp. 31–44.

[17] A. Legout and E. W. Biersack, "Pathological behaviors for RLM and RLC," in *Proceedings of the NOSSDAV*, June 2000.

[18] O. Papaemmanouil and U. Cetintemel, "Semcast: Semantic multicast for content-based data dissemination," in *Proc. WebDB*, June 2004.

[19] R. Jain, D. Chiu, and W. Hawe, "A quantitativemeasure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report TR-301*, September 1984.

[20] N. Sastry and S. Lam, "CYRF: A theory of window based unicast congestion control," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 330–342, April 2005.

[21] The network simulator - ns-2. [Online]. Available: www.isi.edu/nsnam/ns/

[22] S. Sarkar and L. Tassiulas, "Back pressure based multicast scheduling for fair bandwidth allocation," in *Proc of IEEE INFOCOM.*, Alaska, 2001.

[23] S. Bajaj, L. Breslau, and S. Shenker, "Uniform versus priority dropping for layered video," in *Proc of ACM SIGCOMM.*, Sep. 1998.
[24] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, C. Sreenan, and S. Wen, "A simple loss differentiation approach to layered multicast," in *Proc of IEEE INFOCOM.*, Israel, March 2000.

PLACE PHOTO HERE

Jun Peng received his Ph.D. degree in electrical engineering from the Electrical, Computer, and Systems Engineering Department of Rensselaer Polytechnic Institute, Troy, NY, USA in 2004. He is currently an assistant professor at the Electrical Engineering Department of University of Texas - Pan American, Edinburg, TX, USA. His general research interests are in computer communication networks and embedded wireless systems. He has published referred papers in conferences and journals in the areas of medium access control, congestion control, error control, and protocol design and analysis. He has recently been working on or exploring topics in medium access control, broadcasting control, routing, and security in the general area of wireless networks. He is a member of IEEE.

PLACE PHOTO HERE

Biplab Sikdar (S'98, M'02) received the B. Tech degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, the M. Tech degree in electrical engineering from Indian Institute of Technology, Kanpur and Ph.D in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA in 1996, 1998 and 2001, respectively. He is currently an Associate Professor in the Department of Electrical, Computer and Systems Engineering of Rensselaer Polytechnic Institute, Troy, NY, USA. His research interests include wireless MAC protocols, network routing and multicast protocols, network security and queuing theory.

Dr. Sikdar is a member of IEEE, Eta Kappa Nu and Tau Beta Pi.

PLACE PHOTO HERE

Liang Cheng received the BS degree from Huazhong University of Science and Technology, Wuhan, China, and the MS degree from Tsinghua University, Beijing, China. He received the PhD degree in electrical and computer engineering from Rutgers, The State University of New Jersey in May 2002. Since August 2002, he has been with the Department of Computer Science and Engineering at Lehigh University as an assistant professor and directed the Laboratory of Networking Group (LONGLAB). His research interests are wireless networks and mobile computing. He has been the principal investigator (PI) and a co-PI of projects funded by the US National Science Foundation (NSF), the Defense Advanced Research Projects Agency (DARPA), and other sponsors. He has published more than 50 technical papers and holds 1 patent. He is a receipt of Christian R. & Mary F. Lindback Foundation Minority Junior Faculty Award in 2004.