Hop-by-hop Congestion Control over a Wireless Multi-hop Network

Yung Yi and Sanjay Shakkottai

***Abstract*— This paper focuses on congestion control over multi-hop, wireless networks. In a wireless network, an important constraint that arises is that due to the MAC (Media Access Control) layer. Many wireless MACs use a time-division strategy for channel access, where, at any point in space, the physical channel can be accessed by a single user at each instant of time.**

In this paper, we develop a fair hop-by-hop congestion control algorithm with the MAC constraint being imposed in the form of a channel access time constraint, using an optimization based framework. In the absence of delay, we show that this algorithm are globally stable using a Lyapunov function based approach. Next, in the presence of delay, we show that the hop-by-hop control algorithm has the property of spatial spreading. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. We derive bounds on the "peak load" at a node, both with hop-by-hop control, as well as with end-to-end control, show that significant gains are to be had with the hop-by-hop scheme, and validate the analytical results with simulation.

Keywords: Control theory, Mathematical programming/optimization

I. Introduction

We consider the problem of congestion control over wireless, multi-hop networks. Nodes in such networks are radio-equipped, and communicate by broadcasting over wireless links. Communication paths between nodes which are not in radio range of each other are established by intermediate nodes acting as relays to forward data toward the destination. The diverse applications of such networks range from community based roof-top networks to large-scale ad-hoc networks.

Over the past few years, the problem of congestion control has received wide-spread attention, both in the Internet context [1], [2], [3], as well as in an ad-hoc network context [4]. Most of this research has focused on modeling, analysis, algorithm development of end-to-end control schemes (such as TCP), and adaptation of such schemes to ad-hoc networks. Given routing path and bandwidth constraints, algorithms have been developed which converge and have a stable operation.

In a wireless context, however, an important additional constraint that arises is that due to the MAC (Media Access Control) layer. Many wireless MACs use a time-division strategy for channel access [5], [6], where, at any point in space, *the physical channel can be accessed by a single user at each instant of time (a time constraint)*. This paper formulates an optimization framework for congestion control algorithm in wireless multi-hop networks with the constraint imposed by the MAC. We develop a distributed, *hop-by-hop* congestion control scheme, which is shown to be stable in the absence of propagation delays.

This research was supported by NSF Grants ACI-0305644 and CNS-0325788, and a grant from the Texas Telecommunications Engineering Program (TxTEC). The authors are with the Wireless Networking and Communications Group, Department of Electrical and Computer Engineering, The University of Texas at Austin, email: {yi, shakkott}@ece.utexas.edu. A shorter version of this paper appeared in the Proceedings of IEEE Infocom, Hong Kong, March, 2004.

Hop-by-hop congestion control algorithms have been studied in the Internet context [7], [8], [9]. Such schemes provide feedback about the congestion state at a node to the hop preceding it. The preceding node then adapts its transmission rate based on this feedback. Feedback is typically provided [7], [8], [10], [11], [9] based on the queue length at the congested node. If the queue length exceeds a threshold, congestion is indicated and the preceding node is notified in order to decrease its transmission rate. It is well known that such schemes, by reacting to congestion faster than end-to-end schemes (the bottleneck node would send feedback *backward*, thus decreasing the delay in the control loop), result in better performance than a corresponding end-to-end scheme. However, Internet congestion control has been dominated by end-to-end schemes (in particular, TCP), and research in alternate mechanisms in the recent past has focused on the same [1], [3], [2], primarily due to scalability and deployability. Hop-by-hop schemes require to have per-flow state management in intermediate nodes, which generates scalability problems. However, in a wireless network, the number of flows per node is of a much smaller order than in the Internet. Further, wireless networks usually have per-flow queueing for reasons of packet scheduling [5], and the fact that different users are at different locations, thus requiring different physical layer strategies (such as the channel coding and modulation scheme of the power level). Thus, we argue that hop-by-hop schemes are feasible over a wireless network.

In this paper, we develop a hop-by-hop control scheme, which is shown to converge in the absence of delay, and allocates bandwidth to various users in a proportionally-fair manner. In the presence of delay, we show that it has the property of *spatial spreading*. In other words, focused loads at a particular spatial location in the network get "smoothed" over space. In Figure 1, we illustrate this effect. Consider a node accessed by a number of flows. While an end-to-end control scheme could result in large transient overloads (due to delayed feedback) at a single node, a hop-by-hop scheme will "push-back" and cause congestion to occur over space, resulting in smaller peak overloads. Thus, even if the bottleneck node is very close to the receiver (the "worst-case" for a hop-by-hop scheme), there are potential gains to be had due to spatial spreading. *Hence, even if the total buffer*

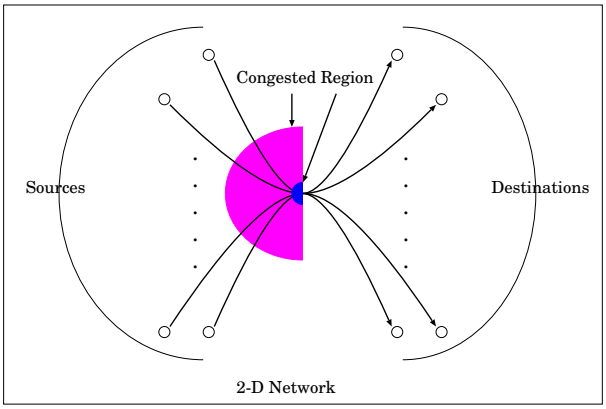

Fig. 1. Spatial Spreading with hop-by-hop controllers

requirement over the network is the same, the hop-by-hop scheme ensures that the buffers required are spatially spread.

II. Main Contributions

The main contributions in this paper are:

(i) We develop (weighted) proportionally-fair congestion control algorithms (both hop-by-hop as well as end-to-end) with the MAC constraint being imposed in the form of a channel access time constraint, using an optimization based framework. In the absence of delay, we show that these algorithms are globally stable using a Lyapunov function based approach.

(ii) We consider the evolution of these algorithms in the presence of propagation delay. We analytically show the effect of spatial spreading, by explicitly deriving the reduction in peak buffer overload under the hop-by-hop scheme for a tree network. We show that at a bottleneck node, the difference in the peak queue length between an end-to-end scheme and a hop-by-hop scheme is at least of order $L^{\alpha}N$, where L is the number of hops, N is the number of sessions, and for some $\alpha \geq 1$.

A. Related Work

The work of [12], [2] provides an optimization based framework for Internet congestion control and derives a differential equation based distributed solution. Works of [13], [1], [14], [3], [15], [16] study the stability of such end-to-end controllers in the presence of feedback delay.

In [8], [17], [7], [9], using a simulation based approach, the authors provide hop-by-hop control algorithms and show that the hop-by-hop schemes react faster than end-to-end schemes, thus reducing buffer requirements. In [10], the author proposes a framework for congestion control and routing based on push-back, where-in, queue buildup at a down-stream node causes upstream nodes to decrease rate and use alternate paths. This has been extended to the multicast case in [11].

Related work includes [18], where the authors consider max-min fair scheduling in the context of a wireless network using a similar model as that considered here for media access control (MAC). The authors develop a token based local scheduling policy at each node to ensure max-min fairness.

This paper differs in that we develop rate based (end-to-end and hop-by-hop) controllers with the objective of (weighted) proportionally-fair resource allocation among users, and with MAC constraints. We derive explicit bounds on queue lengths in the presence of propagation delay, both with an end-to-end and hop-by-hop scheme, and demonstrate spatial spreading with hop-by-hop control.

B. Organization

We begin with a description of the system model in section III, and discuss an utility function based network optimization framework.

Next, in Section VII, we illustrate spatial spreading in a hop-by-hop algorithm by means of deriving bounds on the peak queue lengths in the presence of feedback delay. We provide simulation results in Section VIII to validate the analysis.

III. System Model

Consider a network with a set L of links, a set V of vertices (nodes), and let c_l be the finite capacity of link l, for $l \in L$. Each vertex corresponds to a node in the network. Each data flow r in the network corresponds to an ordered sequence of links $l \in L$, and we denote R as the set of possible sessions[1]. Thus, we model a wireless link between any two nodes in the network to have a finite positive capacity.

In reality, wireless channels are time-varying [19], each with some average capacity which will depend on the physical layer scheme. However, in this paper, we model the link to have a fixed capacity. Such a model is accurate in two regimes: (i) where the channel changes are *masked* by the physical layer coding and modulation scheme so as to present a "constant channel" to the higher layer, or, (ii) the channel changes much slower than the congestion control scheme. In this case, using a time-scale decomposition argument, we can then formally justify a constant channel model at the time-scale of the congestion controller (thus leading to a fluid model for the MAC). Further, as in [18], we assume that at any instant of time, data flows that do not share nodes can transmit/receive simultaneously, but data flows that share a node cannot do so. In other words, simultaneous transmissions can take place over links (i.e., between a pair nodes) as long as the links do not share a common node.

This, for instance, models a wireless system where multiple frequencies/codes are available for transmission (using FDMA/CDMA), and enables parallel communications in a neighborhood using such orthogonal FDMA/CDMA channels (see [18] for additional discussion). In addition, allowing simultaneous parallel transmissions could also model wireless systems that employ interference cancellation [19].

Thus, access constraints at the MAC/PHY layers arise due to the fact that each node has only a single transceiver, and hence cannot perform multiple transmissions or receptions simultaneously. We next describe the constraints on the data flows that follows from this wireless system model.

[1]We use the words 'session' and 'flow' interchangeably throughout this paper.

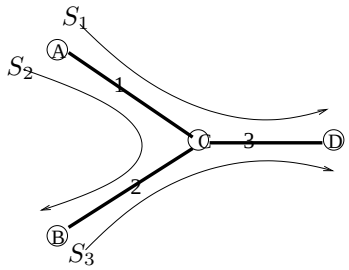

Fig. 2. Example network for time and link constraint

TABLE I
LINK AND TIME CONSTRAINTS FOR THE EXAMPLE NETWORK IN FIGURE 2

Link Constraint	Time Constraint
$\frac{x_1+x_2}{c_1} \leq 1$	$\frac{x_1+x_2}{c_1} \leq 1$
$\frac{x_2+x_3}{c_2} \leq 1$	$\frac{x_2+x_3}{c_2} \leq 1$
$\frac{x_1+x_3}{c_3} \leq 1$	$\frac{x_1+x_3}{c_3} \leq 1$
	$\frac{x_1+x_2}{c_1} + \frac{x_2+x_3}{c_2} + \frac{x_1+x_3}{c_3} \leq 1$

There are two types of constraints that are imposed, namely, (i) the *link constraint* and (ii) the *time constraint*. The link constraint corresponds to the fact that the sum of date rates of all sessions that traverses through link $l \in L$ is not greater than c_l, the capacity of link l.

The *time constraint* means that at any instant of time, there can be only one instance of communication at a given node. To illustrate a fluid model for this constraint, we consider an example shown in Figure 2.

The network consists of three sessions S_1, S_2 and S_3, with each of the sessions traversing two links as shown in Figure 2. Let $x_i, i = 1, 2, 3$ be the data rate of the sessions respectively. We observe that the time constraint is imposed on each *node* in the network. Let us consider node 'C' in the figure, and define $y_{ij}, i, j \in \{1, 2, 3\}$, by

$$y_{ij} \quad = \quad \frac{x_i}{c_j}$$

Observe that y_{11} can be interpreted as the *fraction of time* node 'C' expends to receive data of session 1 from node A over an *unit interval of time*. Similarly, y_{13} is interpreted as the fraction of time expended by node 'C' to transmit data of session 1 to node D over an *unit interval of time*. Similar interpretations hold for all y_{ij}. Thus, as total fraction of time expended at node 'C' cannot exceed '1', the *time constraint* at node 'C' is

$$\sum_{i,j} y_{ij} \quad \leq \quad 1.$$

Similar time constraints apply for all other nodes in the network. Table I presents various link and time constraints for the network in Figure 2. As we can observe from the table, the link constraints are subsumed by the time constraints. Any link constraint is trivially a time constraint, if it is the only flow and terminates at the node. In all other cases, the time constraint is strictly stronger than a link constraint. Thus, we do not need to consider link constraints, and will henceforth restrict ourselves to only time constraints. In general, the time constraints presented above are *not sufficient* to ensure that a feasible MAC protocol exists [18], [20]. However, a feasible MAC always exists if the time constraints are relaxed by replacing the RHS of the expressions (i.e., the term '1') by a parameter $\rho \leq 2/3$. This corresponds to the fact that 100% utilization of resources at each node may not be always feasible because of the network topology (see [18] for an example). However, it has been shown in [20] that if the time constraint is relaxed to $2/3$, a feasible MAC always exists.

A. An Optimization Problem

Let us denote $N(L)$, $N(V)$, and $N(R)$ as the number of links, nodes, and sessions, respectively. For any link l and session r, let $\mathcal{A}_{lr} = \frac{1}{c_l}$ if link l is in the path of flow r, and 0 other-wise. Thus, we define the matrix $\mathcal{A} \in \mathcal{R}^{N(L)\times N(R)}$ by

$$\mathcal{A} = \begin{cases} \mathcal{A}_{lr} = 1/c_l & if\ link\ l\ in\ session\ r, \\ 0 & otherwise \end{cases} \tag{1}$$

Similarly, define $\mathcal{G}_{vl} = 1$ if link l is incident on node v, and 0 other-wise. Thus, define the matrix $\mathcal{G} \in \{0,1\}^{N(v)\times N(L)}$ is defined by

$$\mathcal{G} = \begin{cases} \mathcal{G}_{vl} = 1 & if\ link\ l\ incident\ on\ node\ v, \\ 0 & otherwise \end{cases} \tag{2}$$

Using $\mathcal{G}$ and $\mathcal{A}$, time constraint for a given network can be expressed as:

$$\mathcal{G}\mathcal{A}\mathbf{x} \leq (1-\epsilon)\mathbf{1} \tag{3}$$

for some $\epsilon \in [0, 1]$, and $\mathbf{x}$ corresponds to the vector of user data rates. The parameter ϵ corresponds to the "efficiency" of the MAC protocol used, and additionally models the feasibility of a MAC protocol for the given network topology[2](see Section III, as well as later in this section for additional discussion).

For each user (session) r, let x_r be the data transmission rate. Associated with each user (session) is a utility function $U_r(\cdot)$, which is the "reward" or utility that user r gets by transmitting at the rate of x_r (see [12] for further discussion). Assume that the utility $U_r(x_r)$ is an increasing, strictly concave, and continuously differentiable function of x_r over the range $x_r \geq 0$. In this paper, we restrict ourselves to weighted proportionally fair utility functions of the form $U_r(\cdot) = w_r \log(\cdot)$. From a resource allocation point of view, the resource allocation achieved under any concave and increasing utility functions can be achieved by a weighted proportionally-fair allocation[3] [21] through appropriate choice of weights $\{w_r\}$.

The objective is to maximize total utility in the network subject to the link and time constraints. In this paper, we

[2]Since each flow path is different in the given network configuration, it would be more accurate to represent the MAC efficiency by a vector $\epsilon = [\epsilon_1, \cdots, \epsilon_r]$. However, we use a same ϵ for every flow for simplicity of analysis. Different MAC efficiency depending on the given flow path can be implemented by choosing $\tilde{t_j}$ (9) suitably.

[3]However, the transient dynamics of a decentralized controller may be different.

develop congestion control mechanisms to share the time resources in the network in a (weighted) proportionally fair manner. We consider a fluid model for the MAC, and do not focus on the actual implementation of the resource sharing mechanism at each node. For example, an ideal MAC algorithm would allow the maximum possible (subject to MAC feasibility) time-resources at each node to be used for successful data transfer. However, an ALOHA based MAC would have inefficiencies associated with it, which would allow only a fraction of the time resources at a node to be used for successful data transfer. At the fluid time-scale, the details of these different MAC protocols manifest only as an *efficiency* factor that is captured by the parameter ϵ in (3), which governs the fraction of time that the time resource at each node can be used for successful data transfer. As discussed earlier, the efficiency factor is chosen such that some MAC protocol is feasible for the given network topology. From our earlier discussion, $\epsilon \geq 1/3$ ensures that a time-division MAC is always feasible independent of the network topology [20]. Further, an inefficient MAC scheme (such as random access) would be associated with a larger value of ϵ.

Thus, for any fixed $\beta > 0$, and with session rates $\mathbf{x} = (x_r, r \in R)$, we need to solve

$$\mathbf{P}: \ \max \textstyle\sum_{r\in R} \frac{w_r}{\beta} \log(x_r)$$

subject to

$$\mathcal{G}\mathcal{A}\mathbf{x} \leq (1-\epsilon)\mathbf{1}$$

over

$$\mathbf{x} \geq 0$$

As the cost function is strictly concave and the constraint set is convex, there is a unique solution to **P**. In following sections, we develop a decentralized congestion control algorithms (both hop-by-hop and end-to-end) to address **P**.

IV. Distributed end-to-end Algorithm

A. Algorithm Description

In this section, we develop an end-to-end congestion control algorithm to solve **P**. As the optimization problem has a strictly concave cost function, and convex constraints, we solve **P** using Lagrange multipliers. The Lagrangian of the problem **P** is:

$$L(\mathbf{x}, \lambda) = \sum_{r\in R} \frac{w_r}{\beta} \log(x_r) + \lambda^T(\mathcal{G}\mathcal{A}x - (1-\epsilon)\mathbf{1}). \quad (4)$$

We denote the input and output link of a session r on v when a session r goes through v as $l_i(v,r)$ and $l_o(v,r)$, respectively (for instance, in Figure 2, are $l_i(C,1) = 1$ and $l_o(C,1) = 3$). For completeness, for the source and destination nodes, we define $c_{l_i(s(r),r)} = \infty$ and $c_{l_o(d(r),r)} = \infty$ respectively, where the source and the destination of session r are denoted by $s(r)$ and $d(r)$. Let us denote $A_j(r)$ as the set of all downstream nodes from j in the path of session r. Thus, $A_{s(r)}(r)$ is the collection of all nodes in the path of session r.

By differentiating (4), we have

$$\frac{dL}{dx_r} = \frac{w_r}{\beta x_r} - \sum_{j\in A_{s(r)}(r)} \big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j\big) = 0 \quad (5)$$

Therefore, the unique solution to the problem **P** is given by the following condition:

$$x_r = \frac{w_r}{\beta \sum_{j\in A_{s(r)}(r)} \big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j\big)} \quad (6)$$

We now present rate adaptation mechanisms for session sources. At each time t, we denote the transmission rate of session r by $x_r(t)$. Suppose that $x_r(t)$ adapts according to

$$\dot{x_r}(t) = \kappa\Big(w_r - \beta x_r(t) \sum_{j\in A_{s(r)}(r)} \big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j(t)\big)\Big), \quad (7)$$

where

$$\lambda_j(t) = p_j\Big(\sum_{s\in D(j)} x_s(t)\big(\frac{1}{c_{l_i(j,s)}} + \frac{1}{c_{l_o(j,s)}}\big)\Big), \quad (8)$$

$p_j(y)$ is a *marking function* at node j, and determines the fraction of flow to be marked. Here, $D(j)$ corresponds to the set of sessions incident on node j.

This function is an indicator of (time) congestion at a node, and sources adapt based on the congestion indication [12], [2]. As in the Internet context, this function is assumed to be a continuous, increasing function with range $[0, 1]$.

Observe that (7) is analogous to the differential equation developed in [12]. However, (7) differs from the algorithm of [12] in that (7) handles relative transmission or reception times instead of actual rates.

To understand the intuition for (7), observe that λ_j is interpreted as the price for using node j per unit time. In addition, $x_r(t)(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})$ is the fraction of time the MAC at node j expends in receiving and re-transmitting (to the next hop) the data from session r. As *time* is the resource in our formulation, the total cost of using node j equals $x_r(t)(\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j$. Thus, the source congestion control mechanism tries to equalize the aggregate cost $x_r(t)\sum_{j\in A_{s(r)}(r)}\big((\frac{1}{c_{l_i(j,r)}} + \frac{1}{c_{l_o(j,r)}})\lambda_j(t)\big)$ with w_r.

B. Marking function

As discussed earlier, corresponding to each node j in the network is a marking function $p_j(\cdot)$. In this paper, we consider a marking function of the form

$$p_j(y) = \Big(\frac{y - \widetilde{t}_j}{y}\Big)^+ \quad (9)$$

As seen in (8), the parameter y of $p_j(y)$ is the sum of MAC time utilizations by *all flows*, both incoming and outgoing, at node j.

Thus, $p_j(y)$ marks the fraction of flow which exceeds a *time threshold* $\widetilde{t}_j$. Observe that the total time utilization at the MAC cannot exceed 1. Thus, $\widetilde{t}_j < 1$ is a parameter that *controls the desired time utilization* at the link. For instance, for an inefficient MAC (say, random access), one would set $\widetilde{t}_j \ll 1$. We will discuss the choice of this parameter in Section IV-C.

C. Stability Analysis

In this section, we show that the system of controllers defined in (7) is globally stable. The proof is analogous to that in [12]. Let the function $U(\mathbf{x})$ be defined by

$$U(\mathbf{x}) = -\sum_{j\in V}\int_0^{\sum_{s\in D(j)} x_s(t)\left(\frac{1}{c_{l_i(j,s)}}+\frac{1}{c_{l_o(j,s)}}\right)} p_j(y)\,dy + \sum_{r\in R}\frac{w_r}{\beta}\log x_r \tag{10}$$

We can show that $U(\cdot)$ is a strictly concave function, with a unique equilibrium $\mathbf{x}^*$. Analogous to controlling utilization by using a virtual capacity [22], [15], the equilibrium rate $\mathbf{x}^*$ can be suitably chosen by choosing appropriate values for the time thresholds $\{\widetilde{t}_j\}$, and the constant β. In particular, this choice could be such that the equilibrium $\mathbf{x}^*$ solves the optimization problem **P** discussed in Section III-A. Adaptively choosing these parameters in a manner similar to that in [23], [22] is a topic for future research. We now show that the congestion controllers described by (7) and (8) converge to this equilibrium point.

Proposition 4.1: $U(\mathbf{x})$ is a strictly concave, Lyapunov function for the system of differential equations (7). The unique value of $\mathbf{x}$ maximizing $U(\mathbf{x})$, denoted by $\mathbf{x}^*$ is a stable point of the system, to which all trajectories converge.

Proof: The proof is analogous to that in [12], where we use the utility function defined in (10). We skip the details. ■

V. Distributed hop-by-hop Algorithm

In this section, we develop a distributed hop-by-hop algorithm for congestion control. First, we observe that the congestion controller at the source of each session reacts based on the sum of the congestion prices at each node. Instead of passing this feedback downstream as in the end-to-end algorithm, one could envisage a scheme where each node passes the (partial sum) price upstream. In other words, each node adds its current congestion cost to that it received from a downstream node, and passes this information toward the upstream node. The source will ultimately receive the sum of all price information from the corresponding downstream nodes and use the information for controlling rates. We refer to Figure 3 for the illustration of the hop-by-hop algorithm.

The basic idea of a hop-by-hop algorithm is that every node in the path of the session operates a congestion control algorithm. In Figure 3, the congestion price at node C is passed to the upstream node B. Node B computes its local congestion price and adds it to the congestion price from node C. Node B adapts its transmission rate to node C based on this sum of congestion prices. In addition, node B passes this sum of two prices to the upstream node A. Using this "price passing" method, the source of session 1 receives aggregate congestion price from its downstream nodes and controls its transmission rate based on it.

Let us denote $a_r^i(t)$ as the *actual transmission rate* at the i-th hop of session r in the hop-by-hop control algorithm. Corresponding to each node i along the path of session r, is

feedback A: $\frac{1}{c_1}\lambda_A(t) + \left(\frac{1}{c_1}+\frac{1}{c_2}\right)\lambda_B(t) + \frac{1}{c_2}\lambda_C(t)$

feedback B: $\left(\frac{1}{c_1}+\frac{1}{c_2}\right)\lambda_B(t) + \frac{1}{c_2}\lambda_C(t)$

feedback C: $\frac{1}{c_2}\lambda_C(t)$

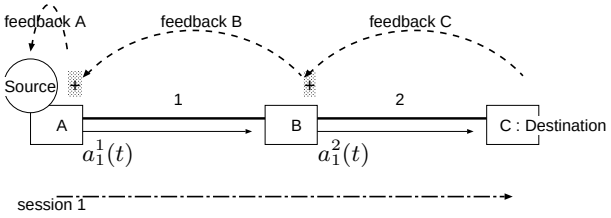

Fig. 3. hop-by-hop Congestion Control Algorithm

a *virtual transmission rate* $c_r^i(t)$, which is described by

$$\dot{c_r^i}(t) = \kappa\Big(w_r - \beta a_r^i(t)\sum_{j\in A_k(r)}\left(\frac{1}{c_{l_i(j,r)}}+\frac{1}{c_{l_o(j,r)}}\right)\lambda_j(t)\Big), \tag{11}$$

$$a_r^i(t) = \min[c_r^i(t), a_r^{i-1}(t)], \tag{12}$$

where k is the node corresponding to the i-th hop of session r. $\{\lambda_j(t)\}$ are defined similar to that in (8), but with the actual transmission rates instead of the source transmission rates. Along the path of each flow r, and for each hop i, the initial conditions for the virtual transmission rates are assumed to satisfy $c_r^i(0) \geq c_r^{i-1}(0)$ (in particular, all of them could be equal).

Thus, in the above algorithm, we sum over all prices downstream along session r. Thus, each node operates a (per-flow) controller based on the perceived congestion due to *downstream nodes*, and determines the maximum rate it can transmit at (the virtual transmission rate). The actual rate it chooses transmits at the rate of the minimum of the *incoming* data rate[4] from $i-1$-th hop node in the session's path (the previous hop node), i.e, $a_r^{i-1}(t)$, and the maximum possible rate $c_r^i(t)$.

We comment that at each intermediate node, the controller has knowledge of the local link rates, as well as the "ramp-up" constant w_r for each of the sessions that is incident on the node. It can be shown that the stability analysis and later analysis are valid even if the node uses an upper bound on the ramp-up constant. Thus, from an implementation perspective, one could assume that $\{w_r\}$ are globally bounded by some value w, and use this value at each intermediate node. Heuristically, the convergence proofs are valid even when a bound is used because the data transmission rate into the network is ultimately governed by the source, which will use the correct value of w_r. However, to keep the exposition simple, we will use the exact value of w_r at each node in this paper.

[4]For the source node for each fbw, (12) is not considered, as there is no upstream node. Instead we let the actual and virtual transmission rates to be the same.

Proposition 5.1: The transmission rates for the hop-by-hop controller described in (11) and (12) converge to the equilibrium value $\mathbf{x}^* = (x_1^*, \ldots, x_R^*)^T$ given in Proposition 4.1. In particular, for each route r, and for each hop i, $a_r^i(t) \rightarrow x_r^*$ as $t \rightarrow \infty$.

Proof: At any fixed time t, note that $a_r^i(t)$ *decreases* along a flow path. This follows from (12), where the $\min$ implies that $a_r^i(t) \leq a_r^{i-1}(t)$. Further, from (11), it follows that the the sum of the Lagrange multipliers *decreases* along a flow path (as smaller number of non-negative terms added as we get closer to the destination). These two facts imply that

$$\frac{d}{dt}\left(c_r^i(t) - c_r^{i-1}(t)\right) \quad \geq \quad 0,$$

from (11). This observation, along with the ordering of the initial conditions of $\{c_r^i(0)\}$ implies that $c_r^i(t)$ *increases* along a flow path (i.e., for increasing values of i). In other words, for each i, r, t, we have

$$c_r^i(t) \quad \geq \quad c_r^{i-1}(t) \tag{13}$$

Next, for each i, r, from (12), we have that

$$a_r^i(t) \quad \leq \quad c_r^i(t) \tag{14}$$

Thus, from (13) and (14), we have $c_r^i(t) \geq c_r^{i-1}(t) \geq a_r^{i-1}(t)$. Hence, it follows that

$$\begin{aligned} a_r^i(t) &= \min\{c_r^i(t), a_r^{i-1}(t)\} \\ &= a_r^{i-1}(t). \end{aligned}$$

This implies that, for each flow r, the actual transmission rates $\{a_r^i(t), i = 1, 2, \ldots\}$ are the same over all hops. This in-turn implies that the source dynamics for flow r is the same as source dynamics of the end-to-end controller. Hence, from Proposition 4.1, we are done. ∎

VI. Congestion Control with Delay

In the previous section where we proved stability, we assumed that the time resource was large enough so that queueing did not occur (or equivalently, the time threshold $\widetilde{t}$ are suitably chosen). In this section, we do not make such an assumption. We will study the dynamics with queueing in the presence of feedback delay.

For the end-to-end algorithm, we denote the output transmission rate of session r at k-th hop traversing the link l by $x_{r,l}^k(t)$. The superscript k corresponds to the fact that link l is k-th hop of the path of the session r. Thus, $x_{r,l_i(j,r)}^{k-1}(t)$ and $x_{r,l_o(j,r)}^{k}(t)$ are the incoming input and outgoing transmission rate in the end-to-end algorithm respectively.

Similarly, for the hop-by-hop algorithm, we denote the actual and maximum (virtual) sending rate of session r at k-th hop traversing the link l. by $a_{r,l}^k(t)$ and $c_{r,l}^k(t)$, respectively. Thus, $a_{r,l_i(j,r)}^k(t)$ and $a_{r,l_o(j,r)}^k(t)$ are the actual incoming input and actual outgoing transmission rates of session r at node j respectively.

Finally, each node has a per-flow buffer to temporarily store data before forwarding. We denote the queue length of session r at node j by $q_{rj}(t)$.

A. The End-to-End Controller with Delay

Unlike in the delay-free case considered in Section IV, queueing can occur at intermediate nodes due to feedback delay. In this section, we describe the detailed dynamics of rates for a session at each node.

For each node j, let us define $E_I^j(t)$ by

$$E_I^j(t) \quad = \quad \sum_{r \in D(j)} \frac{x_{r,l_i(j,r)}^{k-1}(t)}{c_{l_i(j,r)}}$$

Thus, $E_I^j(t)$ is the fraction of the time resource at the MAC of node j consumed by incoming flows, and $D(j)$ corresponds to the set of sessions incident on node j. We will assume that the MAC protocol at the nodes ensure that $E_I^j(t) < 1$. Thus, if a timing overload occurs at a node, data loss will occur, causing unsuccessful transmissions to be queued at the preceding hop (where the data was transmitted from). We assume a suitable error and collision detection mechanism exists such that data is queued in case of timing overload. Thus, from a fluid model perspective, we can assume that the successful data transmission into a node j satisfies $E_I^j(t) < 1$. In addition, a poor MAC protocol may not be able to support a time utilization of '1' (for instance an ALOHA based MAC would have a maximum time utilization less than 0.36). However, in the following discussions, we will assume that the MAC can support a time utilization of '1' for notational ease. The results that are presented can be easily generalized to non-ideal MACs by suitable scaling. Let us now define

$$E_O^j(t) \quad = \quad \sum_{r \in D(j)} \frac{x_{r,l_i(j,r)}^{k-1}(t)}{c_{l_o(j,r)}}$$

The interpretation of $E_O^j(t)$ is the following: If there is no congestion at the node j, the output transmission rates would simply be equal to the incoming rate. $E_O^j(t)$ is the time utilization at the MAC in such a case.

We now consider the following two cases.

(i) $E_I^j(t) + E_O^j(t) > 1$
As the time utilization at the node will exceed '1' if the output flow rates equal the input flow rates, we decrease the transmitted output rates such that the time constraint is met. In other words, we choose $\alpha(t) \in (0, 1]$ such that $E_I^j(t) + \alpha(t) E_O^j(t) = 1$, and set the output transmission rate by $x_{r,l_o(j,r)}^k(t) = \alpha(t) x_{r,l_i(j,r)}^{k-1}(t)$. The remaining flow (of fraction $1 - \alpha(t)$) is queued at node j.

(ii) $E_I(t) + E_O(t) \leq 1$
In this case, the output flow rate for each session can be set to *at least* the input rate of the corresponding session. If some of the sessions have strictly positive queue lengths, i.e., users with backlogged queues (corresponding to congestion in the past), these users are allocated output rates that are greater than their input rates. The rates will be allocated in some fair manner (for example, a proportional rate increase to all backlogged users), subject to the timing constrain being met. Let us denote $Q_j^+(t)$ be the set of backlogged sessions at node j at time t. We choose $\alpha(t) > 1$ such that the time

utilization at the node is less than or equal to one, and for all sessions $r \in Q_j^+(t)$, $x^k_{r,l_o(j,r)}(t) = \alpha(t) x^{k-1}_{r,l_i(j,r)}(t)$.

B. The Hop-by-Hop Controller with Delay

We now develop the dynamics of the hop-by-hop controller with delay. As in the Section VI-A, we define the total time utilization due to incoming flows at node j, by

$$H_I^j(t) \quad = \quad \sum_{r \in D(j)} \frac{a^{k-1}_{r,l_i(j,r)}(t)}{c_{l_i(j,r)}}$$

Let us denote $Q_j^+(t)$ be the set of backlogged sessions at node j at time t, and define

$$H_O^j(t) = \sum_{\substack{r \in D(j) \\ r \in Q_j^+(t)}} \frac{c^k_{r,l_o(j,r)}(t)}{c_{l_o(j,r)}} + \sum_{\substack{r \in D(j) \\ r \notin Q_j^+(t)}} \frac{\min[c^k_{r,l_o(j,r)}(t), a^{k-1}_{r,l_i(j,r)}(t)]}{c_{l_o(j,r)}}$$

where $c^k_{r,l_o(j,r)}(t)$ is the maximum possible rate for flow r at node k, and is described by (11). We now consider two cases:

(i) $H_I^j(t) + H_O^j(t) \leq 1$
In this case, there is no scarce time resource at this node. If the user queues are zero, the output rate is simply equal to the input rate. In general, the output rate for session r is given by

$$a^k_{r,l_o(j,r)}(t) = \begin{cases} \min[c^k_{r,l_o(j,r)}(t), a^{k-1}_{r,l_i(j,r)}(t)] & if\ q_{rj}(t) = 0, \\ c^k_{r,l_o(j,r)}(t) & if\ q_{rj}(t) > 0 \end{cases}$$

(ii) $H_I^j(t) + H_O^j(t) > 1$
In this case, the time resource at node j is potentially not sufficient to handle the output rate. Similar to Case (i) for the end-to-end controller in Section VI-A, we choose $\alpha(t) \in [0, 1)$ such that $H_I^j(t) + \alpha(t) H_O^j(t) = 1$, and set the output transmission rate correspondingly.

VII. Spatial Spreading

In this section, we derive the peak occupied buffer size with the end-to-end controller as well as with the hop-by-hop controller described in Section VI. We consider the evolution of these algorithms in the presence of propagation delay. We analytically show the effect of spatial spreading by explicitly deriving the reduction in peak buffer overload under the hop-by-hop scheme. Consider the tree network in Figure 4, with N sessions and L links between each of the sources and the common destination. Such a network could model a community roof-top wireless network, with the common node being connected to a wired infrastructure. The source node for each session resides on a (different) node as shown in Figure 4. We assume that each link has a round-trip delay of d, and the corresponding end-to-end delay for the session being $D = Ld$.

We assume that the intermediate links (each accessed by only one flow) are well provisioned so that congestion occurs

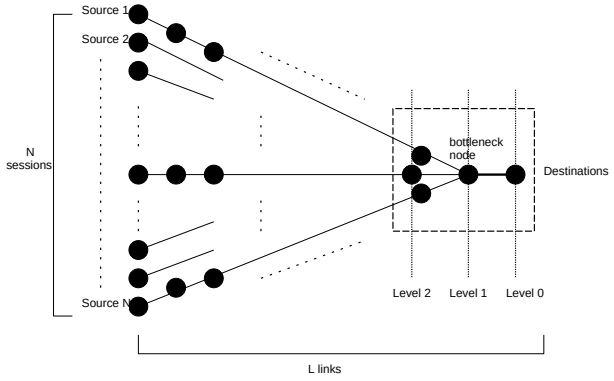

Fig. 4. A tree network with delay

only at the common access point for all the flows (the bottleneck node in Figure 4). Since we consider a system with N flows, we scale the capacities of the bottleneck node with the input and output capacities of the bottleneck node being Nc_I and Nc_O respectively. This scaling ensures that the *steady-state rate allocated to each user is invariant with the number of sessions.* Physically, this would correspond to a bandwidth scaling at the bottleneck.

We first consider the end-to-end scheme and compute maximum queue length at the bottleneck node. As we scale the number of flows N, we need to scale the congestion price appropriately such that the equilibrium rate for each user is invariant with N (this is analogous to scaling the marking function in [15], [25]). To do so, we let the fraction of the flow $x_j(t)$ that is marked be invariant to N. This in-turn implies that the controller marks based on the *normalized* time utilized at the node. Hence, the dynamics of each flow $x_j(t)$ is given by

$$\dot{x_j}(t) = \kappa \Big[w_j - \beta x_j(t-D) \big(\frac{1}{c_I} + \frac{1}{c_O}\big) \; p\Big(\big(\frac{1}{Nc_I} + \frac{1}{Nc_O}\big) \sum_{k=1}^{N} x_k(t-D) \Big) \Big] \qquad (15)$$

Thus, the steady-state rate of flow j, denoted by x_j^* is given by

$$x_j^* \quad = \quad \frac{w_j x^*}{\beta(x^*(1/c_I + 1/c_O) - \widetilde{t})},$$

where x^* is the average steady-state rate over all flows, and is invariant with N.

We finally comment that we have assumed that the feedback (marks) do not experience congestion, and that the delay in the feedback is solely due to propagation delays. As we have per-flow queueing, a packet implementation to approximate this could be the following. When congestion occurs at a node, instead of marking the incoming packet (implemented via setting the ECN (Explicit Congestion Notification) bit [26]), one could mark the head-of-line (outgoing) packet in the queue of the corresponding user. This would ensure that the queueing delays are minimized for the feedback, and that the source gets the appropriate feedback. Such a scheme is feasible in

a wireless context, as per-flow queueing is expected to be implemented for scheduling as well as physical layer reasons.

Let $x(t) = \frac{1}{N}\sum_{j=1}^{N} x_j(t)$ and $w = \frac{1}{N}\sum_{j=1}^{N} w_j$. By summing (15) across all sessions, we obtain

$$\begin{aligned}\dot{x}(t) &= \kappa\Big[w - \beta\Big(x(t-D)\big(\frac{1}{c_I}+\frac{1}{c_O}\big) - \widetilde{t}\Big)^+\Big] \\ &= \big(\frac{1}{c_I}+\frac{1}{c_O}\big)\kappa\beta\Big[\frac{w}{\beta\big(\frac{1}{c_I}+\frac{1}{c_O}\big)} \\ &\quad - \Big(x(t-D) - \frac{\widetilde{t}}{\big(\frac{1}{c_I}+\frac{1}{c_O}\big)}\Big)^+\Big] \qquad (16)\end{aligned}$$

Now, denoting $\widetilde{c} = \frac{\widetilde{t}}{\big(\frac{1}{c_I}+\frac{1}{c_O}\big)}$, $\widetilde{\kappa} = \kappa(\frac{1}{c_I}+\frac{1}{c_O})\beta$, and $\widetilde{w} = w/(\beta(\frac{1}{c_I}+\frac{1}{c_O}))$, we have

$$\dot{x}(t) = \widetilde{\kappa}[\widetilde{w} - (x(t-D) - \widetilde{c})^+] \qquad (17)$$

Next, for each time t, under the end-to-end control scheme, let us denote the average queue length (across sessions) at the bottleneck node by $q^e(t)$, and the average input and output rates by $x_I(t)$ and $x_O(t)$ respectively. Observe that congestion occurs at the node if $\frac{x_I(t)}{c_I} + \frac{x_O(t)}{c_O} > 1$. Further, observe that $x_I(t) \le c_I$. We now describe the dynamics of the queue length process. We consider several cases:

(i) $\frac{c_I c_O}{c_I+c_O} < x_I(t) \le c_I$

$$\begin{aligned}\dot{q}^e(t) &= x_I(t) - x_O(t) \\ &= c_I\frac{x_I(t)}{c_I} - c_O\big(1 - \frac{x_I(t)}{c_I}\big) \\ &= \big(\frac{c_I+c_O}{c_I}\big)\big[x_I(t) - \frac{c_I c_O}{c_I+c_O}\big] \qquad (18)\end{aligned}$$

(ii) $x_I(t) \le \frac{c_I c_O}{c_I+c_O}$ and $q^e(t) > 0$
The dynamics of $\dot{q}^e(t)$ are identical to that in Case (i).

(iii) $x_I(t) \le \frac{c_I c_O}{c_I+c_O}$ and $q^e(t) = 0$
In this case, as the buffer at the bottleneck node is empty, and there is no congestion, we have $\dot{q}^e(t) = 0$.

Thus, with

$$\dot{\widetilde{q}}^e(t) = \frac{\dot{q}^e(t)}{(c_I+c_O)/c_I}, \qquad (19)$$

we have

$$\dot{\widetilde{q}}^e(t) = \begin{cases} x_I(t) - c & if\ \widetilde{q}^e(t) > 0, \\ (x_I(t)-c)^+ & if\ \widetilde{q}^e(t) = 0 \end{cases} \qquad (20)$$

where $c = \frac{c_I c_O}{c_I+c_O}$.

We next derive the "worst-case" peak queue lengths at the bottleneck node under the end-to-end controller as well as a hop-by-hop controller, due to initial transients. Let ${Q_{max}}^e(T)$ be the (unscaled) maximum queue length at the bottleneck node with end-to-end control with the round-trip delay for each session being T. Thus, this would correspond to the tree network in Figure 4 having L links per session, with *each link* having a round-trip delay of T/L. Also let ${q_{max}}^e(T) = {Q_{max}}^e(T)/N$ be the peak queue length for the scaled system defined by (17) and (20). With this definition, we have

Lemma 7.1: Fix any $\delta > 0$. Then, $\exists(L_o, \alpha)$ with $\alpha \ge 1$ and $L_o \ge 1$, such that $\forall L \ge L_o$, $L^\alpha {Q_{max}}^e(\delta) \le {Q_{max}}^e(L\delta)$.

The proof is presented in Appendix A. Using this result, we prove the main result of this section. Let ${Q_{max}}^h(T)$ be the (unscaled) maximum queue length (due to initial transients) from with the *hop-by-hop control*, and ${q_{max}}^h(T)$ be the corresponding scaled queue-length, where T is the round-trip delay for each session. We then have

Proposition 7.1: Fix any $\delta > 0$. Then, $\exists(L_o, \alpha)$ with $\alpha \ge 1$ and $L_o \ge 1$, such that $\forall L \ge L_o$, $L^\alpha {q_{max}}^h(L\delta) \le {q_{max}}^e(L\delta)$.

Proof: Observe that an upper bound on the queue length at the bottleneck node with the hop-by-hop control can be derived by assuming an infinite backlog of data at all the nodes preceding the bottleneck node (the Level 2 nodes in Figure 4). As the control loop for this hop has round trip delay δ, with no intermediate nodes, it follows that ${q_{max}}^h(L\delta) \le {q_{max}}^e(\delta)$. Thus, from Lemma 7.1, the desired result follows. ■

Remark 7.1: As we are computing the peak load due to initial transients, let us interpret N as the number of flows which start up at approximately the same time. Then, from Proposition 7.1, we have for some $\alpha \ge 1$,

$${Q_{max}}^e(Ld) - {Q_{max}}^h(Ld) \quad \sim \quad O(L^\alpha N)$$

Thus, even if number of flows are relatively small, potentially significant gains are to be had due to the multiplicative effect of the delay in the control loop. In Section VIII, we will see that we achieve significant gains even with only five flows.

VIII. Simulation Results

In this section, we present simulation results that compare the hop-by-hop algorithm with the end-to-end algorithm. Through both fluid and packet simulation, we show that there is a significant decrease in the peak load with the hop-by-hop algorithm.

A. Fluid Simulation

The topology used in the simulation is a tree network shown in Figure 4, with $N = 5$ and $L = 5$. In other words, we consider a network with five hops, and five sessions. Each input and output link of the bottleneck node is set to have a capacity of 40. Thus, the equilibrium sum rate over sessions at the bottleneck node is 20 under timing constraint. The round-trip delay per hop is assumed to be 4 units, leading to an end-to-end round trip propagation delay of $D = 20$ units. For example, if time is measured in milli-seconds, and capacity in bytes per unit-time, this system would correspond to a 2 msec one-way delay per hop, with the capacity of the link being 40 kbytes/sec. However, due to the time constraint at the MAC, this capacity will be shared by the incoming and outgoing components of each flow.

Figure 5 shows the aggregate rate at the bottleneck node with the end-to-end controller as well as the hop-by-hop controller. We see that the convergence times to steady-state are approximately the same, as the end-to-end delay is the same, and the bottleneck node is very close to the destination (the "worst-case" for the hop-by-hop controller). However, if we consider the corresponding peak queue lengths at the bottleneck, we see that there is a significant difference, as predicted by the analytical results in Section VII. This

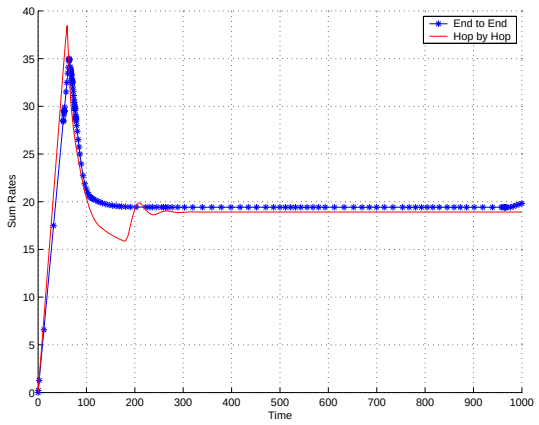

Fig. 5. Sum Rates of both controllers with $\kappa = 1$, $w = 0.13$, and $\tilde{t} = 0.415$.

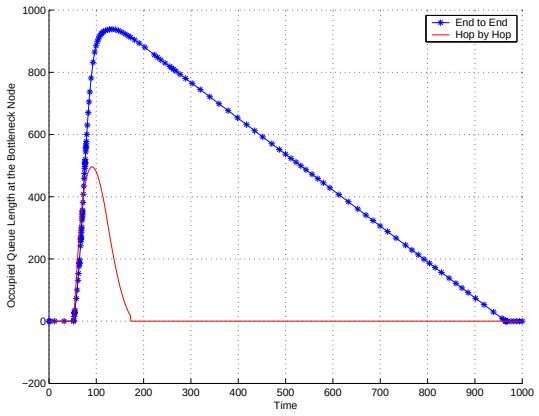

Fig. 6. Occupied queue length at the bottleneck node ($D = 20$)

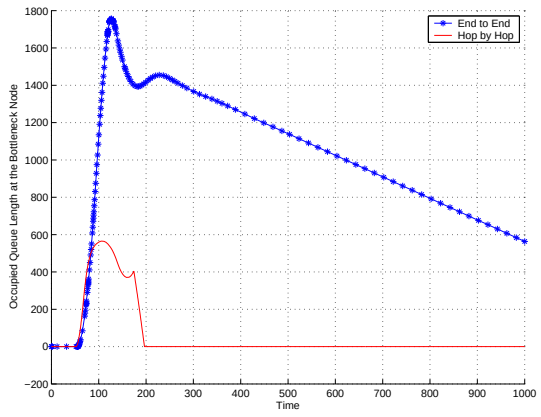

Fig. 7. Occupied queue length at the bottleneck node ($D = 40$)

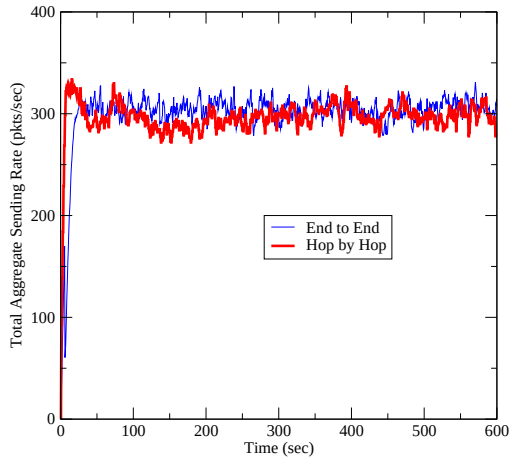

Fig. 8. Rates of a session of both controllers, One hop delay=10 msec

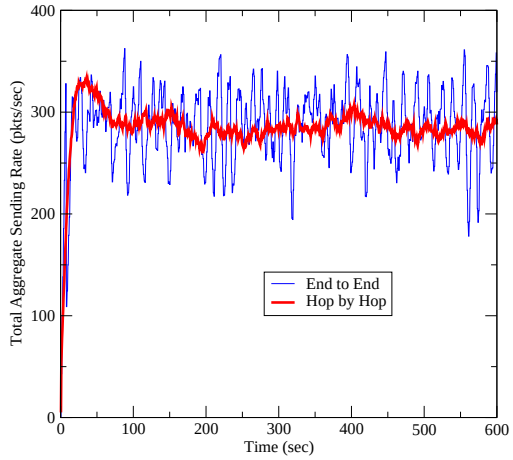

Fig. 9. Rates of a session of both controllers, One hop delay=200 msec

illustrates the effect of spatial spreading. Even though the convergence properties are about the same, the peak queue length at the bottleneck node under the hop-by-hop scheme is smaller.

In Figure 7, we increase the round trip delay to $D = 40$ (corresponding to a one-way per hop delay of 4 msec), this effect is exacerbated. Thus, the results in this paper argue for considering hop-by-hop controllers for a wireless multi-hop network.

B. Packet Simulations

In this section, we consider packet level simulations using the ns-2 [27] simulator. We have made suitable modifications in ns-2 to support hop-by-hop as well as end-to-end controllers with a MAC. In our implementation, we approximate the fluid model for a MAC with a time-division MAC.

The network topology used in the packet simulation is the same as that in the fluid simulation presented in Section VIII. Each of the input and output links at the bottleneck node has a bandwidth of 5 Mbps. Since we set the size of packet to be 1000 bytes, this corresponds to 625 pkts/sec. Thus, the equilibrium rate over one session at the bottleneck is 62.5 pkts/sec. We suitably set w and $\tilde{t}$ such that the link utilization ratio is about 95%.

Figures 8 and 9 plot the instantaneous sum rate of five sessions for 10 msec and 200 msec of one-hop delay, respectively. The 10 msec case corresponds to an efficient MAC, where the hop-delay primarily occurs due to the propagation delay and physical layer air interface. On the other-hand, the case where the hop delay is 200 msec corresponds to a

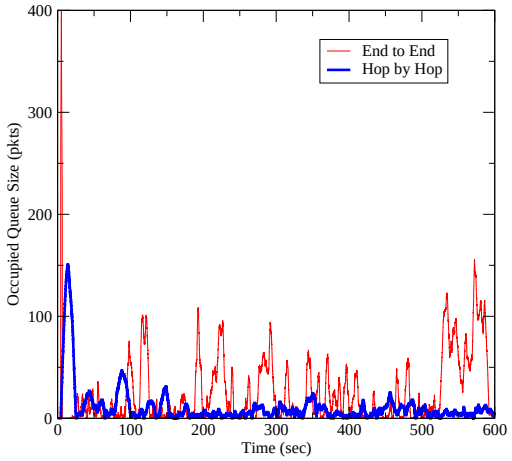

Fig. 10. Occupied queue length at the bottleneck node. One hop delay=10 msec

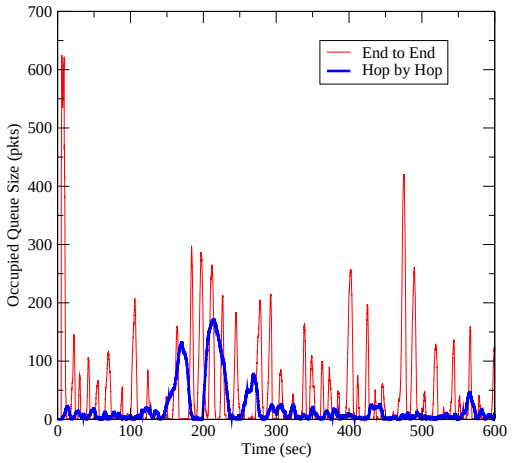

Fig. 11. Occupied queue length at the bottleneck node. One hop delay=200 msec

network with large per-hop delays (possibly due to the MAC implementation). In both of these cases (Figures 8 and 9), we see that the transmission rates (with the end-to-end and hop-by-hop controllers oscillate about the corresponding fixed point. As expected, we observe that for the large delay case, the end-to-end algorithm leads to larger oscillations than the hop-by-hop algorithm.

In Figures 10 and 11, we plot the evolution of the bottleneck queue length (measured every 20 msec). The spatial spreading effect is clearly illustrated by these plots, where we see that the peak buffer size with the hop-by-hop controller is much smaller than that of the end-to-end controller. These fluid and packet simulations indicate that significant gains are to be had with the hop-by-hop scheme, and validate our analytical results.

IX. Acknowledgments

The authors would like to thank Prof. Gustavo de Veciana for his comments and valuable discussions.

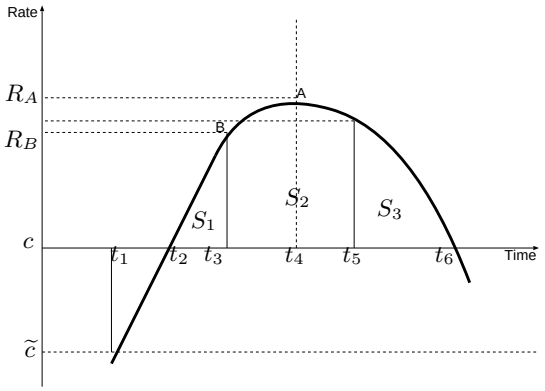

Fig. 12. Trajectory of $x(t)$ when $1 \leq \delta\widetilde{\kappa} \leq 2+\sqrt{2}$

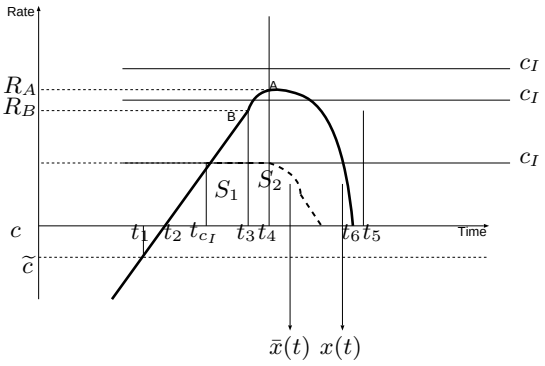

Fig. 13. Trajectory of $x(t)$ when $\delta\widetilde{\kappa} > 2+\sqrt{2}$

Appendix A: Proof of Lemma 7.1

Lemma 7.1: Fix any $\delta > 0$. Then, $\exists(L_o, \alpha)$ with $\alpha \geq 1$ and $L_o \geq 1$, such that $\forall L \geq L_o$, $L^{\alpha} {Q_{max}}^{e}(\delta) \leq {Q_{max}}^{e}(L\delta)$.

Proof: Let us define the following time epochs. Let t_1 be the time such that $x(t)$ crosses $\widetilde{c}$, t_2 be the first time after t_1 such that $x(t)$ crosses c, $t_3 = t_1 + \delta$, $t_4 = t_2 + \delta$, $t_5 = t_3 + \delta$, and t_6 be the first time after t_4 such that $x(t)$ crosses c. These times are illustrated in Figure 12 and Figure 13. Formally, we can define these time epochs by

$$
\begin{aligned}
t_1 &= \inf\{t > 0 : x(t) > \widetilde{c}\} \\
t_2 &= \inf\{t > t_1 : x(t) > c\} \\
t_3 &= t_1 + \delta \\
t_4 &= t_2 + \delta \\
t_5 &= t_3 + \delta \\
t_6 &= \inf\{t > t_4 : x(t) < c\}
\end{aligned}
$$

We note that depending on $\widetilde{\kappa}\delta$, t_6 can be greater (Figure 12) or less (Figure 13) than t_5. Also denote the corresponding values of the trajectories by $R_A = x(t_4)$ and $R_B = x(t_3)$.

Now observe that the peak queue length at the bottleneck node is given by

$$
{q_{max}}^{e}(\delta) = \int_{t_2}^{t_6} x(t)\, dt \tag{21}
$$

We assume that the initial condition satisfies $x(s) \leq \widetilde{c}, \forall s \leq 0$. First, we note that for fixed $\widetilde{\kappa}$ such that $\delta\widetilde{\kappa} < 1$, we have ${q_{max}}^e(\delta) < {q_{max}}^e(1/\widetilde{\kappa})$, which follows from a monotonicity property of the peak queue length with respect to delay (we skip the proof due to space constraints). Thus, we will henceforth consider the case where $\delta\widetilde{\kappa} \geq 1$.

By the assumption about the initial condition, over $[t_1, t_3]$, we have

$$\dot{x}(t) = \widetilde{\kappa}\widetilde{w} \tag{22}$$

Using the fact that $c - \widetilde{c} = \widetilde{\kappa}\widetilde{w}(t_2 - t_1)$, we have

$$t_2 - t_1 = \frac{c - \widetilde{c}}{\widetilde{\kappa}\widetilde{w}} = \frac{1}{\widetilde{\kappa}} \tag{23}$$

In addition,

$$t_4 - t_3 = t_2 - t_1 = \frac{1}{\widetilde{\kappa}} \tag{24}$$

We can see that $x(t)$ achieves the maximum at t_4 since $\dot{x}(t_4) = 0$. This follows from the fact that

$$\begin{aligned} \dot{x}(t_4) &= \widetilde{\kappa}(\widetilde{w} - x(t_2)p(x(t_2))) \\ &= \widetilde{\kappa}(\widetilde{w} - cp(c)) \\ &= 0. \end{aligned}$$

Now, let $\bar{x}(t)$ be the input arrival rate at time t to the bottleneck node. We have $\bar{x}(t) \leq c_I$, from the input link bandwidth constraint at the bottle neck node[5] Recall that $R_B = x(t_3)$. As $R_B - \widetilde{c} = (t_3 - t_1)\widetilde{\kappa}\widetilde{w}$, we have

$$R_B = \delta\widetilde{\kappa}\widetilde{w} + \widetilde{c} \tag{25}$$

Depending the relative values of t_5 and t_6, the trajectory of $\bar{x}(t)$ is either of the form shown in Figure 12 or that in Figure 13. We now derive a sufficient condition on $\delta\widetilde{\kappa}$ such that $t_5 \geq t_6$. It can be shown that the upper bound on t_6, denoted by $\hat{t}_6$, occurs when the input link bandwidth constraint does not limit the arrival rate at the bottleneck node (i.e., $\bar{x}(t) = x(t)$), which corresponds to the case $R_A > c_I$ in Figure 13. We define $y(t) = x(t + t_3)$, and we have

$$\begin{aligned} y(t - \delta) &= x(t + t_3 - \delta) \\ &= \widetilde{\kappa}\widetilde{w}t + \widetilde{c} \end{aligned}$$

Thus, for $t \in [0, \delta]$, we have

$$\begin{aligned} \dot{y}(t) &= \widetilde{\kappa}(\widetilde{w} - (\widetilde{\kappa}\widetilde{w}t + \widetilde{c})p(\widetilde{\kappa}\widetilde{w}t + \widetilde{c})) \\ &= \widetilde{\kappa}(\widetilde{w} - \widetilde{\kappa}\widetilde{w}t) \end{aligned} \tag{26}$$

Using the fact that $y(0) = R_B$, from (25) and (26), we have

$$y(t) = \widetilde{\kappa}(\widetilde{w}t - \frac{\widetilde{\kappa}\widetilde{w}t^2}{2}) + d\widetilde{\kappa}\widetilde{w} + \widetilde{c} \tag{27}$$

By definition, we have $y(\delta) = x(t_5)$. We now derive the condition on $\delta\widetilde{\kappa}$ such that $y(\delta) = c$. This will correspond to $\bar{t}_6 = t_5$.

$$\begin{aligned} y(\delta) = c &\Leftrightarrow \widetilde{\kappa}(\widetilde{w}t - \frac{\widetilde{\kappa}wt^2}{2}) + d\widetilde{\kappa}\widetilde{w} + \widetilde{c} = c \\ &\Leftrightarrow \widetilde{\kappa}(\widetilde{w}\delta - \frac{\widetilde{\kappa}\widetilde{w}\delta^2}{2}) + \delta\widetilde{\kappa}\widetilde{w} - \widetilde{w} = 0 \\ &\Leftrightarrow (\widetilde{\kappa}\delta)^2 - 4\widetilde{\kappa}\delta + 2 = 0 \end{aligned} \tag{28}$$

[5]Thus, it is possible that $\bar{x}(t) < x(t)$, in which the MAC could cause data to be temporarily buffered at nodes preceding the bottleneck node, see Case (i) in Section VI-A.

Solving, we get $\widetilde{\kappa}\delta = 2 + \sqrt{2}$. Thus, for all $\delta > 0$ such that $\widetilde{\kappa}\delta \geq 2 + \sqrt{2}$, this condition ensures that the trajectory of $x(t)$, and thus $\bar{x}(t)$, is of the form shown in Figure 13.

Further, from the monotonicity property of the queue length with respect to delay, for fixed $\widetilde{\kappa}$, and any δ such that $\widetilde{\kappa}\delta \leq 2 + \sqrt{2}$, we have ${q_{max}}^e(\delta) \leq {q_{max}}^e\left(\frac{2+\sqrt{2}}{\widetilde{\kappa}}\right)$. Henceforth, we only consider the case where $\widetilde{\kappa}\delta \geq 2 + \sqrt{2}$ (corresponding to Figure 13).

The peak queue-length computation differs depending on the relative position of c_I with R_A and R_B. We first consider the case where $c_I > R_A$ (see Figure 13).

Let us denote the area of the region S_1 (over the time interval $[t_2, t_3]$) in Figure 13 by $A(S_1)$. Then,

$$\begin{aligned} A(S_1) &= \frac{1}{2}(t_3 - t_2)(R_B - c) \\ &= \frac{\widetilde{w}}{2\widetilde{\kappa}}(\delta\widetilde{\kappa} - 1)^2 \end{aligned} \tag{29}$$

By definition, we have

$$A(S_2) = \int_0^s y(t)\ dt, \tag{30}$$

where s is chosen such that $y(s) = c$. From (27), we have $s = \frac{1+\sqrt{2\delta\widetilde{\kappa}-1}}{\widetilde{\kappa}}$. Thus, we have

$$\begin{aligned} A(S_2) &= \int_0^s (y(t) - c)\, dt \\ &= \frac{-\left(\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)^3\widetilde{\kappa}\widetilde{w}\right)}{6\,\widetilde{\kappa}^2} \\ &+ \frac{\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)\ (\delta\widetilde{\kappa}\widetilde{w} - \widetilde{w})}{\widetilde{\kappa}} \\ &+ \frac{\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)^2\widetilde{w}}{2\,\widetilde{\kappa}} \end{aligned} \tag{31}$$

Thus, from the equations (29) and (31), when we have $c_I > R_B$, the peak queue length is given by

$$\begin{aligned} A(S_1) + A(S_2) = &\frac{\widetilde{w}}{2\widetilde{\kappa}}(\delta\widetilde{\kappa} - 1)^2 + \\ &+ \frac{-\left(\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)^3\widetilde{\kappa}\widetilde{w}\right)}{6\,\widetilde{\kappa}^2} \\ &+ \frac{\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)\ (\delta\widetilde{\kappa}\widetilde{w} - \widetilde{w})}{\widetilde{\kappa}} \\ &+ \frac{\left(1 + \sqrt{-1 + 2\delta\widetilde{\kappa}}\right)^2\widetilde{w}}{2\,\widetilde{\kappa}} \end{aligned} \tag{32}$$

Next, we consider the case where $c_I < R_B$. Let t_{c_I} be the first time after t_2 such that $\bar{x}(t)$ hits c_I. Let us define $\Delta_t = t_{c_I} - t_2$, and we have

$$\Delta_t = \frac{c_I - c}{\widetilde{\kappa}\widetilde{w}}$$

Next, define for $t \in [0, \Delta_t]$,

$$\bar{y}_1(t) = \bar{x}(t + t_4)$$

and for $t \in [0, \delta - \Delta_t]$, define

$$\bar{y}_2(t) = \bar{x}(t + t_4 + \Delta_t)$$

Thus, we have

$$\begin{aligned}\bar{y}_1(t-\delta) &= \widetilde{\kappa}\widetilde{w}t + c \\ \bar{y}_2(t-\delta) &= c_I \end{aligned} \tag{33}$$

Thus, for $t \in [0, \Delta_t]$, we have

$$\begin{aligned}\dot{\bar{y}}_1(t) &= \widetilde{\kappa}(\widetilde{w} - (\widetilde{\kappa}\widetilde{w}t + c)p(\widetilde{\kappa}\widetilde{w}t + c)) \\ &= -\widetilde{\kappa}^2\widetilde{w}t \end{aligned} \tag{34}$$

Similarly, for $t \in [0, \delta - \Delta_t]$, we have

$$\begin{aligned}\dot{\bar{y}}_2(t) &= \widetilde{\kappa}(\widetilde{w} - c_I p(c_I)) \\ &= -\widetilde{\kappa}(c_I - c) \end{aligned} \tag{35}$$

Also, by definition, we have

$$\begin{aligned}\bar{y}_1(0) &= c_I \\ \bar{y}_2(0) &= \bar{y}_1(\Delta_t) \end{aligned} \tag{36}$$

Thus, integrating, we have

$$\begin{aligned}\bar{y}_1(t) &= -\widetilde{\kappa}^2\widetilde{w}\frac{t^2}{2} + c_I \\ \bar{y}_2(t) &= -\widetilde{\kappa}(c_I - c)t - \frac{1}{2\widetilde{w}}(c_I - c)^2 + c_I \end{aligned} \tag{37}$$

In addition, defining $\bar{t}$ to be the first time such that $\bar{y}_2(\bar{t}) = c$,

$$\bar{t} = \frac{1}{\widetilde{\kappa}}(1 - \frac{c_I - c}{2\widetilde{w}}) \tag{38}$$

Thus, the peak queue length at the bottleneck node when $c_I < R_B$ is given by

$$\begin{aligned}&\frac{1}{2}(2(t_4 - t_2) - \Delta_t)(c_I - c) + M \\ &= \frac{1}{2}(2\delta - \Delta_t)(c_I - c) + M, \end{aligned} \tag{39}$$

where $M = \int_0^{\Delta_t} \bar{y}_1(t)\ dt + \int_0^{\bar{t}} \bar{y}_2(t)\ dt$ is *independent of* δ.

Finally, we need to perform a similar computation when $R_B < c_I < R_A$. We skip the details due to space reasons. In any case, it can be seen that (39) provides a lower bound, and (32) provides an upper bound for this case.

To complete our proof, choose L large enough such that $L\widetilde{\kappa}\delta > 2 + \sqrt{2}$. From (39) and (32), the result follows. ■

REFERENCES

[1] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.

[2] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," in *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 2000, vol. 3, pp. 1323–1332.

[3] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *Proceedings of the IEEE Conference on Decision and Control*, December 2001, vol. 1, pp. 185–190.

[4] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proceedings of IEEE/ACM Mobicom*, August 1999, pp. 219–230.

[5] P. Bender, P. Black, M. Grob, R. Padovani, N Sindhushayana, and A. Viterbi, "CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users," *IEEE Communications Magazine*, pp. 70–77, July 2000.

[6] "IEEE 802.11b standards," Available for download at http://grouper.ieee.org/groups/802/11/main.html.

[7] H. T. Kung and A. Chapman, "The FCVC (flow controlled virtual channel) proposal for atm networks," in *Proceedings of ICNP*, October 1993.

[8] P. P. Mishra and H. Kanakia, "A hop by hop rate based congestion control scheme," in *Proceedings of ACM Sigcomm*, August 1992.

[9] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for atm networks: Credit update protocol, adaptive credit allocation and statistical multiplexing," in *Proceeding of ACM Sigcomm*, 1994, pp. 101–114.

[10] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 236–250, February 1995.

[11] S. Sarkar and L. Tassiulas, "Back pressure based multicast scheduling for fair bandwidth allocation," in *Proceedings of IEEE Infocom*, 2001, pp. 1123–1132.

[12] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[13] G. Vinnicombe, "On the stability of end-to-end congestion control for the Internet," 2001, University of Cambridge Technical Report.

[14] L. Massoulie, "Stability of distributed congestion control with heterogenous feedback delays," *Technical Report, Microsoft Research, Cambridge, UK*, 2000.

[15] S. Shakkottai, R. Srikant, and S. Meyn, "Bounds on the throughput of congestion controllers in the presence of feedback delay," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 972–981, December 2003.

[16] R. Johari and D. Tan, "End-to-end congestion control for the Internet: Delays and stability," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, December 2001.

[17] C. M. Ozveren, R. J. Simcoe, and G. Varghese, "Reliable and efficient hop-by-hop flow control," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 642–650, 1995.

[18] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks," in *Proceedings of IEEE Infocom*, New York, NY, June 2002, pp. 763–772.

[19] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Upper Saddle River, NJ, 2002.

[20] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, 1988.

[21] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[22] S. Kunniyur and R. Srikant, "A time-scale decomposition approach to adaptive ECN marking," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 882–894, June 2002.

[23] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proceedings of ACM Sigcomm*, San Diego, CA, August 2001, pp. 123–134.

[24] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network," Technical Report, Wireless Networking and Communications Group, Department of Electrical and Computer Engineering, The University of Texas at Austin, 2003.

[25] S. Shakkottai and R. Srikant, "How good are deterministic fluid models of Internet congestion control?," in *Proceedings of IEEE Infocom*, New York, NY, June 2002, vol. 2, pp. 497–505.

[26] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[27] UCB/LBNL/ISI, Network Simulator - ns (version 2). Available at `http://www.isi.edu/nsnam/ns`