Modeling the interaction of IEEE 802.3x hop-by-hop flow control and TCP end-to-end flow control

Richa Malhotra
Bell-Labs Europe
Lucent Technologies
Capitool 5
7521 PL Enschede
Netherlands
rimalhotra@lucent.com

Ronald van Haalen
Bell-Labs Europe
Lucent Technologies
Capitool 5
7521 PL Enschede
Netherlands
haalen@lucent.com

Michel Mandjes
CWI
P.O. Box 94079
1090 GB Amsterdam
Netherlands
michel@cwi.nl

Rudesindo Núñez-Queija
CWI
P.O. Box 94079
1090 GB Amsterdam
Netherlands
sindo@cwi.nl

Abstract— **Ethernet is rapidly expanding beyond its niche of local area networks. However, its success in larger metropolitan area networks will be determined by its ability to combine simplicity, low costs and quality of service. A key element in successfully transporting bursty traffic and at the same time providing QoS, is congestion control. The Ethernet standard IEEE 802.3x defines a hop-by-hop congestion control mechanism. The performance of this scheme generally depends on its interaction with higher layer application traffic, and especially, with TCP controlled traffic which has its own end-to-end congestion control mechanism. In this paper we focus on the performance modeling and analysis of this interaction. Our model takes into account the influence of various network and traffic parameters. The validity of the proposed model is assessed by comparison of the results to simulations. In our experiments we observe that an increase of the round trip time has a positive influence on the interaction of hop-by-hop and TCP congestion control, and that the use of hop-by-hop flow control is only beneficial when the load is not high.**[1]

I. Introduction

In the times of a telecommunications downturn, Ethernet's move into the metropolitan area domain is providing a rapidly growing market opportunity. Some of the reasons behind its popularity are its low costs, maturity, simplicity and a wide existing base. Over the past years, it has evolved to support greater speeds and distances as well as various levels of services, making it suitable for larger networks. Thus Ethernet seems an obvious choice to provide cost-effective solutions to transport increasing data traffic on existing circuit-switched networks. With its 'low price per bandwidth' Ethernet is expected to speed up the deployment and use of next-generation networks and services.

[1]This work has been carried out in the context of the project End-to-end Quality of Service in Next Generation Networks (EQUANET), which is supported by the Dutch ministry of Economic affairs via its agency SenterNovem.

This work was done while M. Mandjes was also affiliated with the University of Twente.

R. Núñez-Queija is also affiliated with Eindhoven University of Technology.

Ethernet is currently being deployed in existing SDH/SONET networks and in the future, might directly be deployed over optical networks. It is being sold in different flavors, namely, private line, virtual private line, private LAN and virtual private LAN. The most popular version to date is the Ethernet private line as it is simpler to map onto SDH/SONET and does not face major quality of service (QoS) issues. The other above mentioned services involve sharing network resources among various traffic streams. The main goal of using a packet-switched technology is to gain from statistical multiplexing. It is expected that the bursts of the different traffic streams will mostly not coincide and average out in time. However, there will be instances when the bursts will coincide and cause congestion and packet drops. At these instances, proper congestion control techniques provide the key to combining the gain of statistical multiplexing with QoS guarantees.

The IEEE standardization body has recently set up a congestion management study group to deal with the congestion and flow control problems in Ethernet networks. The main idea is that throughput, latency and frame discards can be improved if traffic differentiation can be combined with flow control. The goal of the congestion management study group is to examine if a standard should be defined to differentiate between the way congestion control is applied to different traffic classes. The IEEE already defines a pause mechanism or a *back-pressure* signal (see [1]) to enable congestion messages on a *hop-by-hop* basis. A congested node can send a back-pressure/pause message to its upstream neighbors to signal the stop of all transmissions towards it for a period of time. Within an Ethernet network the use of this signal results in a hop-by-hop congestion control method. An interesting and essential aspect in studying the performance of the Ethernet hop-by-hop mechanism is to understand its influence and interaction on higher layer application traffic, especially since the most widely used application transmission control protocol (TCP) has its own congestion control mechanism.

Considerable work has been done and is ongoing that aims at understanding and predicting the performance of TCP under various situations and different network parameters. Use of an additional underlying hop-by-hop congestion control scheme adds another complication to this study. On one hand, the extra buffering of packets introduced by hop-by-hop control can avoid unnecessary TCP rate fluctuations. On the other hand, if the congestion cannot be solved by hop-by-hop control it might unnecessarily delay the reaction of TCP. Another aspect which needs to be studied before the hop-by-hop mechanism can be implemented is the influence of the various network and traffic parameters on the performance. The throughput of a TCP source is known to be inversely proportional to the round trip time and the square root of the packet loss probability. This relation which is widely known as the 'root *p*' law (see [2]) might not be valid for the combination of TCP and hop-by-hop flow control.

The Ethernet hop-by-hop congestion control mechanism has been evaluated by simulations in the literature. However, in all the previous work, the common aspect is that the attempts concentrate on the protocol and its implementation, but not on understanding the effect of the various network and traffic parameters on the results. Reference [3] presents results from simulation of the Ethernet back-pressure-based congestion control mechanism under various scenarios. It proposes distinguishing hop-by-hop flow control based on traffic classes. Reference [4] shows simulations of TCP traffic with the IEEE 802.3x hop-by-hop flow control mechanism. There is also a significant amount of work done on ATM (Asynchronous Transfer Mode) based hop-by-hop flow control as in [5] and [6]. However, most of this work is based on negotiating transmission rates between the congested node and its upstream neighbors. The proposed mechanisms use resource management cells which do not exist in Ethernet. In this paper we develop a Markovian model that captures the interaction of Ethernet hop-by-hop congestion control with TCP end-to-end congestion control. We then assess the validity of the model by comparing it to extensive simulations. The results demonstrate the influence of various network and traffic parameters on the performance of the schemes and provide guidelines for the choice of parameters such as buffer thresholds for congestion detection.

The rest of the paper is organized as follows. In Section II, we introduce the two congestion control mechanisms. The Markov model describing the integration of the hop-by-hop and the end-to-end congestion control mechanisms is presented in Section III. The simulation model and parameter values used to obtain the results are described in Section IV. Section V, shows the performance results both with the Markov model as well as the simulations, which demonstrate the influence of various network and traffic parameters. The parameters include buffer thresholds for congestion detection, the network round trip time (RTT) and the traffic burstiness. The conclusions are presented in Section VI.

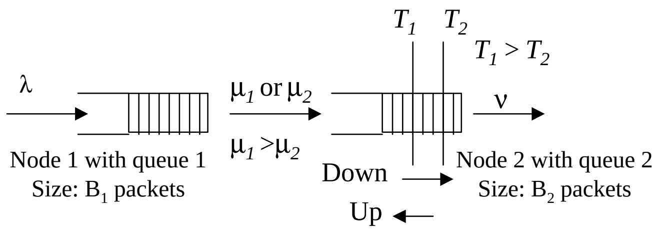

Fig. 1. Hop-by-hop flow control

II. IEEE 802.3x Hop-by-hop and TCP End-to-End flow control mechanisms

The main goal of a hop-by-hop congestion control mechanism is to contain temporary bursts and congestion by utilizing network resources (such as buffer space). The hop-by-hop nature of the congestion control scheme should prevent that congestion escalates to higher layers, for example TCP. In order to model and assess whether the interaction of hop-by-hop with end-to-end congestion control will result in improved network performance, we first need to understand the details of each of these mechanisms separately. These two different schemes as well as some initial modeling concepts are described in the subsections below.

A. IEEE 802.3x back-pressure based hop-by-hop flow control

The back-pressure scheme defined in IEEE 802.3x (see [1]), is intended to provide flow control on a hop-by-hop basis by allowing ports to 'turn off' their upstream link neighbors for a period of time. In the case of a half-duplex link this is realized by sending a jamming signal. The end-station perceives the medium as busy, stops transmitting and backs-off. For the case of a full-duplex connection, the IEEE 802.3x standard defines a MAC layer flow control mechanism. This mechanism is based on a special frame called 'pause frame' in which the pause period is specified. The end station or router receiving the pause frame looks at the pause period and does not transmit or attempt transmission for that amount of time. Alternatively a ON/OFF pause messages can be sent signalling the beginning and end of the 'transmission pause' phase.

This mechanism is illustrated by Figure 1. When the occupancy of queue 2 exceeds T_1 it can signal its upstream node 1 to stop all data transmission. When the queue occupancy of the congested node 2 drops below a low threshold T_2 the upstream neighbor can start transmission again at usual rate μ_1. The Markov model used to describe this mechanism in this paper will assume that on receiving the congestion signal, the upstream neighbor lowers its rate to a generic rate μ_2 instead of strictly stopping all transmissions. A positive value for μ_2 may mimic the delay in reception of congestion messages by the upstream nodes by allowing possible incoming packets into a congested queue even after a congestion message has been sent.

It is clear from Figure 1 and the explanation above that when queue-2 occupancy is above T_1, the service rate of queue 1 is μ_2. Similarly, when the number of packets in queue 2 is below T_2, the service rate of queue 1 is μ_1. In the region between T_1 and T_2 it can be either μ_1 or μ_2, depending on the last threshold that was crossed. For example, when the queue-2 occupancy drops from above T_1 to below T_1, queue 1 continues to transmit at μ_2. Similarly, a rise in queue-2 occupancy from below T_2 to above T_2 implies that queue 1 continues transmission at rate μ_1. We will denote the state of queue 2 corresponding to a high transmission rate (μ_1) of queue 1 as the `up` or '1' state and that corresponding to μ_2 as the `down` or '2' state.

B. TCP end-to-end flow control

The widely used transmission control protocol (TCP), works on the principle that end systems should react to congestion anywhere in an end-to-end data path by adjusting their transmission rates to avoid total collapse. The TCP congestion control mechanism is a well studied subject and is often modeled as an Additive Increase and Multiplicative Decrease (AIMD) scheme (see [7], [8] and [9]). The TCP sending rate is controlled using a window. The window is halved in the event of a packet loss and increased by one packet upon acknowledgment of reception of all packets belonging to the current window. In this paper, we will assume that the packets sent in one window follow a Poisson process with the rate corresponding to the window size.

A single TCP source being an unrealistic scenario, we aim at modeling multiple TCP streams as input sources to a system with hop-by-hop flow control. An approach to modeling multiple TCP streams using Markov chains is to model each stream separately with its own AIMD characteristics. This approach has clear disadvantages with respect to the scalability of the Markov chain transition state space as observed in, e.g., [10]. An alternative to this approach is to model the traffic aggregate of TCP streams with a more generic AIMD mechanism. It is worth noting, though, that this option has the drawback that it is difficult to predict the exact sending rates, among which, the aggregate TCP streams will switch, when detecting packet losses. This complication arises from the fact that packet losses need not affect all TCP streams as observed in [10] and [11].

The arrival process of the aggregate input traffic stream is considered to be a Markov-modulated Poisson process(see [12]) with arrival rates varying between N different values λ_1 to λ_N, where, $\lambda_N < \lambda_{N-1} < \ldots < \lambda_1$. The state of the input traffic stream is represented by a, so that the corresponding traffic rate is λ_a where a varies from 1 to N. Every time there is a packet drop, the input traffic state increases to twice its original value, causing a significant drop in the traffic rate. The input traffic state decreases by one step at a rate equal to $1/\text{RTT}$. This corresponds to an increase in traffic rate, the increase being dependent on the values of the λ_as. This simple approach incorporates the basic AIMD nature in the input traffic.

The different λ_as represent the possible sending rates of an aggregate of multiple TCP streams. It is not straightforward to specify the value of N, i.e., the total number of possible traffic rates, nor the value of the λ_as. This is so, because multiple TCP streams, each with its own AIMD characteristics can overlap in many different ways. The variability in the synchronization arises from the fact that the number of packet drops and the proportion in which they affect the multiple TCP streams is not fixed. One extreme case could occur when the packet drops affect all streams equally, leading to the phenomenon called 'global synchronization'. In this case the AIMD nature of a single flow is preserved. The other extreme could be that only one TCP stream is affected. Our approach to study the impact of such traffic will be to vary the value of N, thus increasing the granularity of the input process and observe the trends in the results.

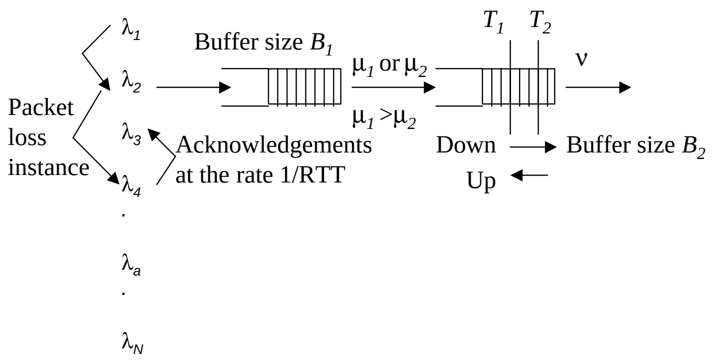

Fig. 2. Integrated hop-by-hop and end-to-end flow control model

III. Integrated model of hop-by-hop and end-to-end congestion control

In this section, our goal is to integrate the models for hop-by-hop flow control and end-to-end TCP flow control. The back-pressure based hop-by-hop congestion control mechanism, works by tuning the buffer occupancy per hop to avoid packet drops as much as possible. The TCP flow control on the other hand, reduces the rate of traffic into the network when packet drops may occur anywhere in an end-to-end data path. In Section 2, we have explained these two mechanisms in detail. The integrated model used for TCP input traffic along with back-pressure hop-by-hop flow control is shown in Figure 2. It consists of augmenting the model for hop-by-hop flow control depicted in Figure 1 with a reactive Markov-modulated arrival process that captures the AIMD behavior of multiple TCP streams. If a packet is lost at either queue 1 or queue 2, the state of the arrival process is adjusted to twice its current value. For example, if this occurs while the TCP rate is λ_3, the new rate is λ_6. On the other hand, the state of the arrival process is decreased by one level on average every 1/RTT time units.

A. Transition Equations

Below we define the states of the system shown in Figure 2. Clearly, the resulting process is also Markovian. The state of the system is represented by (i, j, k, a), which implies i

packets in queue 1, j packets in queue 2, state k – $\texttt{up}$ (1) or $\texttt{down}$ (2) – for the second queue and state a for the TCP traffic rate (varying from 1 to N). The maximum number of TCP sending rate levels is N. For clearness of presentation we enumerate the different types of transitions.

1. $(i, j, k, a) \xrightarrow{\lambda_a} (i+1, j, k, a)$
2. $(B_1, j, k, a) \xrightarrow{\lambda_a} (B_1, j, k, 2a)$, if $2a \le N$
3. $(B_1, j, k, a) \xrightarrow{\lambda_a} (B_1, j, k, N)$, if $2a > N$
4. $(i, j, k, a) \xrightarrow{\nu} (i, j-1, k, a)$, if $j \ge 1$, $j \ne T_2$
5. $(i, T_2, k, a) \xrightarrow{\nu} (i, T_2 - 1, 1, a)$, if $1 \le T_2 \le B_2$
6. $(i, j, 1, a) \xrightarrow{\mu_1} (i-1, j+1, 1, a)$, if $j \ne B_2$, $i \ge 1$
7. $(i, j, 2, a) \xrightarrow{\mu_2} (i-1, j+1, 2, a)$, if $j \ne B_2$, $i \ge 1$
8. $(i, T_1, 1, a) \xrightarrow{\mu_1} (i-1, T_1+1, 2, a)$, if $T_1 < B_2$, $i \ge 1$
9. $(i, B_2, 1, a) \xrightarrow{\mu_1} (i-1, B_2, 1, 2a)$, if $2a \le N$, $i \ge 1$
10. $(i, B_2, 1, a) \xrightarrow{\mu_1} (i-1, B_2, 1, N)$, if $2a > N$, $i \ge 1$
11. $(i, B_2, 2, a) \xrightarrow{\mu_2} (i-1, B_2, 2, 2a)$, if $2a \le N$, $i \ge 1$
12. $(i, B_2, 2, a) \xrightarrow{\mu_2} (i-1, B_2, 2, N)$, if $2a > N$, $i \ge 1$
13. $(i, j, 1, a) \xrightarrow{1/\text{RTT}} (i, j, 1, a-1)$, $a > 1$

Transition equation 1 implies that arrivals into the first queue occur at a rate λ_a. Equations 2 and 3 correspond to losses when queue 1 is full, while simultaneously increasing the input process state by a factor of two. Equation 3 in addition implies that if the calculated increase in the state of the input traffic is more than N then the state will drop to state N. Equation 4 implies departures from the second queue occur at rate ν as long as the second queue is non empty. Equation 5 ensures that when a departure from the second queue coincides with the queue occupancy dropping below the low threshold T_2, then the state of queue 2 changes into '1' ($\texttt{up}$). Equations 6 to 8 are about departures from queue 1 and simultaneous arrivals into queue 2. This happens at rate μ_1 if the state of queue 2 is '1' or at rate μ_2 if the state of queue 2 is '2'. However, when a departure from queue 1 and simultaneous arrival into queue 2 coincides with an increase of the queue occupancy of queue 2 from T_1 to T_1+1 then the state of queue 2 simultaneously changes to '2' ($\texttt{down}$). It is important to note that if T_1 is set equal to B_2 then effectively hop-by-hop flow control is not activated. Transition equations from 9 up to 12 are similar to 6 through 8, but with the additional condition that queue 2 is full when departures from queue 1 arrive into queue 2. This implies that packets will be lost and this will simultaneously affect the state of the input process and trigger a lowering of the traffic rate (increase by a factor of 2 of the state descriptor a). Equation 13 deals with an increase of the TCP traffic rate. The RTT determines the rate at which acknowledgements arrive and corresponds to the increase in TCP window size, therefore increasing the rate of the input process.

B. Performance Measures

By $\pi(i, j, k, a)$ we denote the stationary probabilities obtained after solving the system $\pi \cdot Q = 0$ and $\pi \cdot e = 1$, where Q is the transition matrix of the above described Markov model and e' is a row vector with all entries equal to 1. Then, the expected number of packets in queue 1 is

$$E[\text{Queue 1}] = \sum_i i \sum_{j,k,a} \pi(i, j, k, a)$$

and the expected number of packets in queue 2 is

$$E[\text{Queue 2}] = \sum_j j \sum_{i,k,a} \pi(i, j, k, a).$$

From Little's law, the expected waiting time (that is, the time spent in the queue, including transmission) of packets in queue 1 is

$$EW_1 = \frac{E[\text{Queue 1}]}{\lambda_{\text{effec},1}}$$

where, $\lambda_{\text{effec},1}$ is the effective arrival rate at queue 1 and is given by

$$\lambda_{\text{effec},1} = \sum_{a=1}^{N} \lambda_a \sum_{\substack{i,j,k \\ i<B_1}} \pi(i, j, k, a).$$

In addition, let us denote the effective loss rate at queue 1 by $\lambda_{\text{loss},1}$. This can be computed by

$$\lambda_{\text{loss},1} = \sum_{a=1}^{N} \lambda_a \sum_{j,k} \pi(B_1, j, k, a).$$

The loss probability of queue 1 is

$$P_1 = \frac{\lambda_{\text{loss},1}}{\lambda_{\text{effec},1} + \lambda_{\text{loss},1}} .$$

The expected waiting time of customers in queue 2 is

$$E[W_2] = \frac{E[\text{Queue 2}]}{\lambda_{\text{effec},2}}$$

where $\lambda_{\text{effec},2}$ is the effective arrival rate at queue 2 and is given by

$$\lambda_{\text{effec},2} = \mu_1 \sum_{\substack{i,j,a:i>0 \\ j<B_2}} \pi(i, j, 1, a) + \mu_2 \sum_{\substack{i,j,a:i>0 \\ j<B_2}} \pi(i, j, 2, a).$$

Let us denote the loss rate at queue 2 by $\lambda_{loss,2}$:

$$\lambda_{\text{loss},2} = \mu_1 \sum_{\substack{i,a \\ i>0}} \pi(i, B_2, 1, a) + \mu_2 \sum_{\substack{i,a \\ i>0}} \pi(i, B_2, 2, a).$$

The loss probability of queue 2 is

$$P_2 = \frac{\lambda_{\text{loss},2}}{\lambda_{\text{effec},2} + \lambda_{\text{loss},2}}.$$

Finally, the net throughput of the entire system can be expressed in different ways:

$$\begin{aligned} \text{Throughput} &= \lambda_{\text{effec},2} \\ &= \lambda_{\text{effec},1} \times (1 - P_2) \\ &= \nu \times \sum_{\substack{j>0 \\ i,j,k,a}} \pi(i, j, k, a). \end{aligned}$$

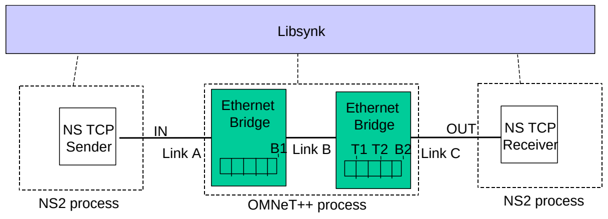

Fig. 3. Simulation Model

IV. Simulation Model and mapping parameters

In the previous sections, we have presented the Markov model of the interaction of the Ethernet hop-by-hop flow control and the TCP end-to-end flow control mechanisms. In many ways this model is a simplification of reality. For TCP, only the main characteristics were included in order to reduce the model complexity. To assess how close the model comes to reality, a simulator containing a network model with 2 nodes was used. Figure 3 shows the simulation model. The OMNeT++ simulator (see [13]) was used to model the Ethernet Bridge, which includes the standard Ethernet mechanisms such as MAC address learning and forwarding, and the hop-by-hop back-pressure signal. in the simulations, TCP traffic was generated using the NS-2 TCP stack ([14]). The two different simulation environments were combined with LibSynk ([15]) as shown in Figure 3.

It is worthwhile to mention the differences between the simulation model and the Markov model.

- The Markov model assumes exponential service times whereas in the simulation these are fixed and deterministic. The time it takes to transmit a packet is computed using the packet size and the link capacity. We have, however, used link speeds, which would lead to the same number of packets being processed, as the average service times in the Markov model.
- Another aspect which is not explicitly taken into account by the Markov model is the dependency of the RTT on the maximum (possible) burst size. The RTT governs the maximum window size and thus the maximum rate. However, this is expected to become noticeable only when there are few TCP flows active, which is not the scenario we aim for. When multiple TCP flows are active, this phenomenon is less relevant.
- The Markov model does not model the slow start behavior of TCP flows.
- The input process in the Markov model is a Markov-modulated Poisson process (see [12]). TCP packets sent within a window do not resemble a Poisson process. Still the Markov model captures high and low sending rates and their adaptation to network congestion.

In order to execute and compare the tests with the Markov model and the simulation model we first need to map the parameters from one onto the other. For both models we have used $B_1 = 10$ and $B_2 = 20$ packets. For the relation between the input traffic sending rates i.e., the λ_as and the simulated TCP traffic scenario we fixed the link speed of Link A to λ_1. Note that for the Markov model arrivals occur according to a stochastic point process, so that the actual inflow of packets may exceed the specified rates for some periods of time. We have executed simulations with different scenarios for file downloads and compared the results against the Markov model.

V. Results

In this section we present the results from the Markov model and the simulation model. The goal is to obtain a better understanding of the effect of various network and traffic related parameters on the interaction of the hop-by-hop and end-to-end congestion control schemes. This comparison also helps in assessing the validity of the Markov models. We have investigated the influence of three main aspects. The first aspect being the different steps in the input process, which determine the granularity of bursts of the input traffic stream. The second aspect is the RTT, which influences the rate at which the input process recovers from loss; the third aspect is the choice of thresholds. The performance measure used, is TCP-level throughput, relating to successfully transmitted packets, often also referred to as goodput. The graphs plot the expected increase in TCP throughput with the use of hop-by-hop flow control as compared to TCP without any additional form of flow control. For all these different aspects we compare the trends observed in the Markov model with simulations for different traffic download scenarios. Having incorporated most of the relevant aspects into the Markov model, the results aim at showing consistent trends, both with the Markov model and the simulations.

A. Scenarios

Since the main goal of hop-by-hop flow control is to contain temporary congestion, we have looked at scenarios with bursty traffic input. It is important to note that in our model and the simulations we do not assume infinite buffer capacity. For the Markov model we have used either $N = 2$ or $N = 3$. In all cases $\lambda_N = 1$ and different values for λ_1 and, if $N = 3$ for λ_2, are considered. The other parameters are as follows $\mu_1 = 100$, $\mu_2 = 2$ and $\nu = 15$ packets per second. The corresponding simulation scenario considered is with (see Figure 3) link speeds of links B and C equal to 100 and 15 packets per second respectively, which translates into 1200 kbps and 180 kbps. The capacity of link A is varied along with λ_1. The TCP Reno version is used for the simulations with packet length as well as MTU equal to 1526 bytes.

B. Round trip time

The round trip time has an important influence on the performance of TCP. Since the RTT dictates the rate at which acknowledgements are received, it influences the way in which the window and thus the TCP sending rate is increased. Thus, the RTT provides the key to recovery of TCP streams from losses. In this section our goal is to study its influence in conjunction with hop-by-hop flow control. This subsection

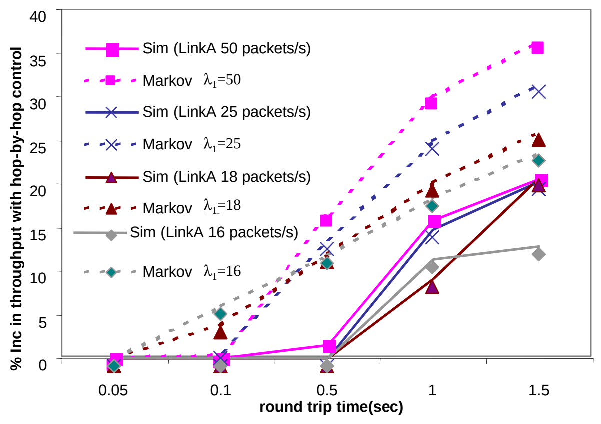

Fig. 4. Throughput benefits of hop-by-hop flow control and its dependence on round trip time

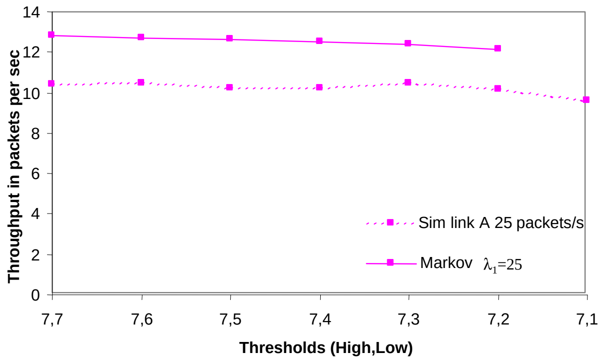

Fig. 5. Results with varying low threshold values while the high threshold is kept constant

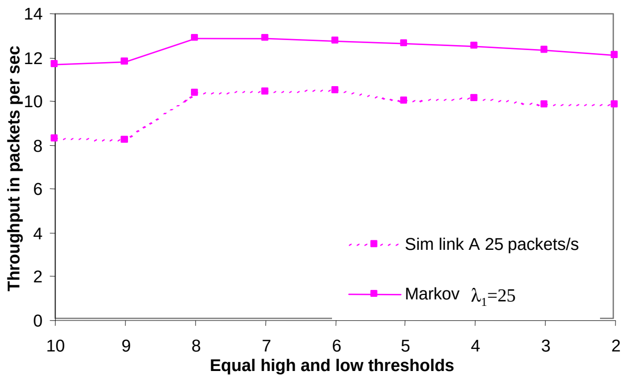

Fig. 6. Results showing varying choice of thresholds with high threshold kept equal to the low threshold

shows the results for varying RTT. For the Markov model we use $N = 2$, $\lambda_2 = 1$ and varying values for λ_1. Similarly, for the simulation model we vary the capacity of link A in Figure 3. The traffic scenario used for the simulations is 8 flows, each flow sending an 80 KB file with 1 second intervals. The RTT is increased by introducing additional link delay. Figure 4 plots the percentage increase in throughput on using hop-by-hop flow control in combination with TCP as compared to TCP alone.

It can be observed from Figure 4 that as the RTT increases, the throughput benefit with the use of hop-by-hop flow control also increases. This general trend was also observed with various other simulation scenarios with smaller numbers of TCP flows and varying file sizes. In order to explain this, it is important to note that with hop-by-hop flow control we use more buffer space, so that there will in general be fewer packets lost. When the RTT is short, TCP recovers quickly from loss and switches back to a faster rate much sooner than when the RTT is long. Thus with a short RTT the congestion level is constantly rather high, ruling out any further optimization. When the RTT is long, the impact of packet loss is greater, since it takes longer to recover from losses. At loss instances, the fact that hop-by-hop flow control causes less packet losses, provides a visible and significant benefit. As there is less loss with the use of hop-by-hop flow control, the TCP sending rate is better adjusted to network resources. With hop-by-hop flow control TCP lowers its sending rate only when the resources (buffer space) are completely exhausted, thus avoiding unnecessary fluctuations in its sending rate.

The Markov model captures the general trend of increase of throughput with increase of RTT. However it seems that the Markov model is more accurate with the trend for higher load λ_1/ν. This can be explained as follows. Since the number of TCP flows and the traffic scenario is kept constant in the simulations, low values of link speeds (λ_1) increase the chance that the link capacity is always utilized to its maximum due to excessive load. For greater link speed values, the same traffic scenario might not provide enough load to constantly send at a rate equal to the link speed irrespective of the amount of packet drops. Variations in the traffic sent on link A with simulations brings it closer to the Poisson assumption in the Markov model, while constant and complete utilization of link A, makes it more deterministic.

C. *Thresholds*

In this section we study the influence of the thresholds T_1 and T_2. The scenario considered is the same as in the previous section but now with $\lambda_1 = 25$. In Figure 5 the high threshold T_1 is kept constant and the lower threshold T_2 is varied. In Figure 6 the placement of the threshold is varied while keeping $T_1 = T_2$.

From Figure 5 we observe the general trend that the throughput tends to decrease as the low threshold T_2 is placed further away from T_1. At the same time it is important to note that this difference is very small. If one were to consider the overhead of the hop-by-hop flow control messages sent everytime the thresholds were crossed, it would be preferred to have the thresholds placed far apart. Figure 6 varies the threshold values from 2 to B_2 keeping $T_1 = T_2$. The results indicate that the thresholds should not be placed too close to B_2 as around this value, the performance degrades significantly. The trends shown in Figures 5 and 6 were also observed with other values

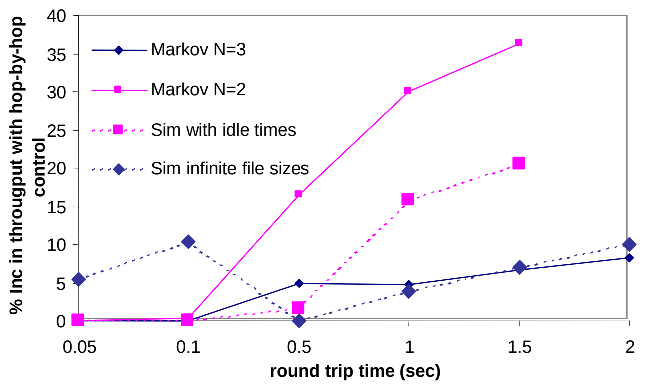

Fig. 7. Effect of input (TCP) traffic sending rate granularity

of λ_1. The optimum in our experiments was always about 70-80% of buffer size. However, it should be kept in mind that all the additional tests were done with same values of μ_2 and the same delay on link A.

D. Granularity of bursts

It was explained in the previous sections that for a given TCP traffic scenario it is difficult to estimate the value of N and the various λ_as for the Markov model. This complication arises from the fact that the aggregate traffic behavior of multiple TCP streams is highly complex, since there might be no consistency in the fluctuation of traffic rates of the various TCP streams. Our goal in this section is to study the influence of the granularity of the variations in TCP sending rate on the performance of hop-by-hop flow control. For the Markov model this implies that we increase N and introduce corresponding intermediate values for λ_a. We have compared the results of the Markov model with different TCP simulation traffic scenarios. We plot the results with $N = 2$ and $N = 3$. It is important to mention that the values and trends observed with higher values of N did not show extreme differences from $N = 3$. We observed that only $N = 2$ provides significantly higher throughput values as compared to other values of N. For the results shown in Figure 7, we have always used $\lambda_1 = 50$ and $\lambda_N = 1$. For $N = 3$, we have used $\lambda_2 = 25$. We have observed in additional tests that λ_a being linear or non-linear in a does not have a major impact on the results. The comparison presented with different simulation scenarios also helps understand the reason behind the difference in performance of the traffic scenarios. The simulation scenarios include the previous results with 8 TCP flows sending file sizes of 80 KB every second and an alternative scenario where 8 flows are used to send infinite file sizes, i.e., there is always data to send.

It can be observed from Figure 7 that the simulations with idle times show far greater benefit with the use of hop-by-hop flow control as compared to the simulations where there is always infinite data to be sent. The reason behind this difference is not only the level of congestion caused by the two scenarios but also the different burstiness of their traffic characteristics. The scenario with idle times, provides hop-by-hop flow control the opportunity to buffer packets from the high rate preceding the idle time, instead of dropping them. During the idle time, the buffered packets can be sent directly rather than waiting for retransmissions. So hop-by-hop flow control manages to smoothen the TCP sending rate. However, with files of infinite sizes there is always data to send, also the level of congestion is far too high to be bridged by extra buffering. There will still be drops and it will result in fluctuations in TCP sending rate. Apparently due to the multiple streams the drop in the aggregate sending rate is not that great. This phenomenon can also be better understood by looking at the results from the Markov models. The Markov model with 2 input steps, follows the trends of the simulation scenario with idle times whereas the more granular input traffic steps in the Markov model follows trends of the simulations without any idle times. From these results it seems evident that when the TCP streams react in granular steps to loss instead of extreme rates, the extra buffering has limited impact. In other words, when TCP itself smoothens out traffic due to congestion, hop-by-hop has nothing more to add. The simulations and the Markov model do however show some very different results with large file sizes for low RTTs. We have observed higher packet loss in the simulations for RTT=0.1 sec. The reason for this could be that many packets are lost affecting all streams at the same time, providing bursts in the TCP input traffic which the hop-by-hop flow control manages to compensate for with extra buffering.

VI. Conclusions

In this paper we have modeled the interaction of the IEEE 802.3x hop-by-hop congestion control mechanism with TCP end-to-end congestion control. We have introduced a Markov model and compared it with simulations of a real TCP stack. The Markov model aims at capturing the interaction of hop-by-hop with TCP congestion control for multiple TCP streams and their aggregate traffic behavior. It does not aim at modeling details of a single TCP flow. The results indicate that the model manages to capture the qualitative performance trends. Only in some specific cases, certain TCP effects and packet drop pattern cause an unexpected degradation in performance without the use of hop-by-hop flow control. With simulation, these cases show an unexpected increase in the benefit of using hop-by-hop flow control.

The model also provides useful insight in the effect of various network and congestion control mechanism parameters on the results. Studying the influence of thresholds for the hop-by-hop congestion control scheme, we have observed that for the scenarios considered, setting the high and low thresholds close to each other seems most optimal. The choice of the threshold should be at 70-80% of the buffer size. This percentage should be adjusted if there is significant transmission delay on the link to which the congestion messages are sent. Setting the thresholds far apart from each other did not show significant degradation in performance. Considering the fact that we did not model the overhead in sending the congestion message on the downstream traffic, it is probably advisable to set the thresholds further from each other (with the additional constraint that the low threshold does not coincide with empty buffer).

The influence of the RTT on the performance was also studied. It can be concluded from the results that increasing the RTT provides greater benefit with the use of hop-by-hop congestion control along with TCP. Since the RTT determines the increase in TCP sending rate after a packet loss, longer RTTs will delay TCP's recovery from loss. In these cases hop-by-hop flow control buffers avoid unnecessary packet loss thus reducing unnecessary long reductions in TCP sending rate. We also observed that the greater the load or congestion level, the less the influence of the RTT on the results as well as that of hop-by-hop congestion control. Short RTTs allow TCP to recover very quickly from packet loss and almost always keep sending at a high traffic rate thus causing extreme congestion. It is difficult to solve extreme congestion if the input traffic rate is not adapted. It is also important to note that since the RTT controls the maximum window size, a very large value of RTT will force TCP flows to send at such a low rate that there will be no congestion at all. In these cases again hop-by-hop flow control will not provide any real benefits.

Burstiness and the level of congestion definitely play a role in the performance improvement of any congestion mechanism. We have studied this aspect by looking at results from the Markov model with extreme differences in input traffic rate and with more granular functions and comparing them with 2 distinct simulation scenarios. TCP traffic scenarios with idle times between files show the largest benefit with the use of hop-by-hop congestion control. The Markov model with extreme bursty changes in traffic rate follows closely these trends in the simulations. When the traffic has idle times, the hop-by-hop congestion control helps in smoothing out traffic and avoids drops and unnecessary retransmissions. However, when there are no idle times and there is always traffic to send, TCP end-to-end congestion mostly seems to adapt well to the bottleneck, thus use of hop-by-hop control does not provide significant benefit. There are some exceptions when certain combinations of the RTT and TCP parameters cause significant drops that can possibly be avoided by using hop-by-hop control. These are subjects for future research.

REFERENCES

[1] Annex 31 B, "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification", *IEEE standard 802.3*, 1998 Edition.

[2] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation", Proc. ACM SIGCOMM, September 1998, pp. 303-314.

[3] W. Noureddine, F. Tobagi, "Selective back-pressure in switched Ethernet LANs", Proc. Globecom, Rio de Janeiro, Brazil, December 1999, 1256-1263.

[4] J. Wechta, A. Eberlein, F. Halsall, "The interaction of the TCP flow control procedure in the end nodes on the proposed flow control mechanism for use in IEEE 802.3x switches", Proc. of the Eigth IFIP conference on high performance networking (HPN'98), Vienna, Austria, September 1998, 515-534.

[5] C. M. Pazos, J. C. Sanchez Agrelo, M. Gerla, "Using back-pressure to improve TCP performance with many flows", Proc. IEEE Infocom, New York, USA, March 1999, 431-438.

[6] P. Mishra, H. Kanakia, "A hop by hop based congestion control scheme", Proc. Communications architectures & protocols, Baltimore, Maryland, USA, August 1992, 112-123.

[7] Y. R. Yang, S. S. Lam, "General AIMD congestion control", Proc. International Conference on Network Protocols, Osaka, Japan, November 2000, 187-198.

[8] V. Dumas, F. Guillemin, P. Robert. "A Markovian analysis of Additive-Increase Multiplicative-Decrease (AIMD) algorithms", *Advances in Applied Probability* 34(1) (2002), 85–111.

[9] F. Baccelli, D. Hong. "The AIMD Model for TCP Sessions Sharing a Common Router", Proc. 39th Annual Allerton Conference on Communication, Control and Computing, Allerton Park, Illinois, USA, October 2001.

[10] N. van Foreest, M. Mandjes, W. Scheinhardt, "A versatile model for asymmetric TCP sources", Proc. of the 18th International Teletraffic Congress, Berlin, Germany, August 2003, 631-640.

[11] J. Crowcroft, P. Oechslin. "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP". *Computer Communication Review* 28 (1998), 53-69.

[12] W. Fischer, K. Meier-Hellstern. "The Markov-modulated Poisson process (MMPP) cookbook", *Performance Evaluation* 18 (1993), 149-171.

[13] OMNET++, http://www.omnetpp.org/

[14] The Network simulator ns-2, http://www.isi.edu/nsnam/ns/

[15] K. Perumalla, LibSynk, http://www.cc.gatech.edu/fac/kalyan/libsynk.htm