Adaptive Scheduling in DOCSIS-based CATV Networks

Huei-Jiun Ju and Wanjiun Liao

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: wjliao@cc.ee.ntu.edu.tw

Abstract This paper studies the effect of the DOCSIS MAC layer on the performance of two-way TCP transfers in Hybrid Fiber Coax (HFC) networks. We propose a new adaptive scheduling scheme called Long Packet Deferment (LPD) at the headend to improve TCP performance in DOCSIS-based HFC networks. LPD reduces the frequency of transmission of long packets and, if such long packets are transmitted, they are scheduled towards the end of each transmission period. Thus, it allows the system to behave as in a symmetric network earlier, reduces the round trip delay of sending data packets, and improves the aggregate downstream throughput. We have conducted simulations using network simulator *ns-2* to compare the simple first-come-first-served scheduling of DOCSIS and IEEE 802.14 like mechanism with LPD. The results show that LPD has better performance in terms of higher aggregate downstream throughput and shorter access delay.

Keywords: TCP, HFC, DOCSIS, asymmetric network

1. Introduction

A Hybrid Fiber Coax (HFC) network is a promising technology to provide broadband access to the Internet. It is a tree-and-branch, point-to-multi-point access network in the downstream direction but a multipoint-to-point, bus access network in the upstream direction. The Multimedia Cable Network System Partners (MCNS) [1] and the IEEE 802.14 Working Group [2] are developing standards to enable data communication capabilities over HFC networks.

MCNS's Data-Over-Cable Service Interface Specifications (DOCSIS) specifies the physical layer modulation and the MAC layer operation for HFC networks. In DOCSIS, the upstream channel is modeled as a stream of mini-slots, is based on a contention-based reservation control mechanism to arbitrate random access to the channel, and is controlled by the Cable Modem Termination System (CMTS) at the headend. Each Cable Modem (CM) sends requests to the CMTS for the use of the upstream channel, and waits for data grants (i.e., slots) allocated by the CMTS. The CMTS regularly transmits downstream management messages called Upstream Bandwidth Allocation (denoted as MAP), which defines transmission intervals on the upstream channel. Each transmission interval specified by a MAP contains request mini-slots and data mini-slots. Request mini-slots are used by CMs to request upstream bandwidth; data mini-slots are used by CMs to transmit data frames. All CMs learn the assignment of bandwidth from the MAP. Thus, each MAP must be received by all the CMs before the beginning of the described transmission period (i.e., its effective time).

Previous studies [1-4] have shown that TCP performance degrades when operating in HFC networks due to the bandwidth asymmetry of downstream and upstream channels. [5] investigates how DOCSIS MAC layer affects bandwidth asymmetry. This paper proposes a new protocol called "Long Packet Deferment" at the headend to solve the TCP performance problem caused by DOCSIS MAC layer scheduling and allocation mechanisms. In particular, we focus on two-way TCP transfers, i.e., both data and ACK packets are transmitted on the upstream channel.

The rest of the paper is organized as follows. Section 2 analyzes the behavior of two-way TCP transfers over DOCSIS's MAC layer. Section 3 presents the proposed mechanism called "Long Packet Deferment (LPD)" to improve TCP performance in DOCSIS-based HFC networks. Section 4 shows the simulation results using ns2 to compare the original DOCSIS MAC layer control mechanism with our mechanism. Finally, we conclude in Section 5.

2. TCP Performance over the DOCSIS MAC Layer

2.1 The Effect of Bandwidth Asymmetry on TCP Performance

Asymmetric networks, such as HFC and xDSL, are defined as networks with different channel capacities in the downstream and upstream directions. The main effect of bandwidth asymmetry on TCP performance is that TCP ACK clocking may be disrupted. [1] defines a bandwidth asymmetric ratio, k, to better understand the behavior of TCP in asymmetric networks:

$$k = \frac{\text{forward channel bandwidth}}{\text{reverse channel bandwidth}} \times \frac{\text{ACK packet length}}{\text{data packet length}} = \frac{C_d}{C_u} \times \frac{L_{ack}}{L_{data}} \quad (1)$$

TCP behaves normally when k is less than or equal to one. When bandwidth is asymmetric (i.e., $k > 1$), ACK packets arrive at the bottleneck link in the reverse direction at a rate faster than the bottleneck link can support. As a result, the sender clocks out data at a slower rate and slows down the growth of the congestion window, which in turn lowers the throughput in the downstream direction.

2.2 The Effect of DOCSIS MAC Layer on Bandwidth Asymmetry

[5] showed that eq (1) alone cannot adequately explain TCP's behavior in DOCSIS-based HFC networks. Considering the MAC layer operation of DOCSIS v1.1, *k* is modified as

$$k = \alpha \times \frac{C_d \times T_{usv}}{d \times L_{data} \times N_{dCM}} \qquad (2)$$

$$\text{where } \alpha = \begin{cases} 1 & \text{, for one-way transfers} \\ \dfrac{N_{dCM} L_{data}}{N_{dCM} L_{data} + \dfrac{N_{uCM} L_{ack}}{d}} & \text{, for two-way transfers} \end{cases}$$

where *d* is a parameter of the delayed ACK policy (i.e., sending one ACK packet to acknowledge the receipt of *d* data packets), T_{usv} is defined as the average time between sending two consecutive packets in the CM buffer, N_{dCM} is the number of simultaneous TCP downloading, and N_{uCM} is the number of simultaneous TCP uploading.

In DOCSIS, the upstream channel is modeled as a stream of mini-slots. A transmission starts only at the beginning of any mini-slot. Let N_{u_ack} be the number of mini-slots used to transmit one ACK packet on the upstream. Given L_{ack}, C_u, and t_{ms}, we can derive the number of mini-slots used to transmit an ACK packet as

$$N_{u\_ack} = \left\lceil \frac{L_{ack}}{C_u} \times \frac{1}{t_{ms}} \right\rceil \qquad (3)$$

where L_{ack} is the size of an ACK packet, t_{ms} is time period defined as one mini-slot on the upstream channel, and C_u is the upstream channel capacity. Similarly, we can derive the number of mini-slots used to transmit a data packet as

$N_{u\_data} = \left\lceil \frac{L_{data}}{C_u} \times \frac{1}{t_{ms}} \right\rceil$, where L_{data} is the size of a data packet.

A MAP describes the bandwidth allocation in a transmission period. It should be received by all the participating CMs before its effective time. Each MAP may be transmitted before some requests, especially piggybacked ones, have arrived and been processed at the CMTS. The late requests are deemed pending in the current transmission period and become backlogged requests in the next period. These pending requests plus new requests which arrive at the CMTS during the next transmission period will be waiting to be granted in the next MAP. Let D_{MAP} be the time difference between when a MAP is transmitted and when it goes into effect. Since the size of the ACK packet is fixed, the maximum number of pending requests, N_{p_REQ}, (i.e., those arrive at the CMTS during a D_{MAP}.) can be expressed as

$$N_{p\_REQ} = \left\lfloor \frac{D_{MAP}}{t_{ms}} \times \frac{1}{N_{u\_ack}} \right\rfloor \qquad (4)$$

Let N_c be the number of mini-slots allocated to the request contention period. Thus, for an ACK packet,

(1) when $N_{dCM} \leqq 2N_{p\_REQ}$, T_{usv} is bounded by

$$(N_c + N_{uCM} N_{u\_data} + N_{dCM} N_{u\_ack})\, t_{ms} \le T_{usv} \le (2\, N_c + 2N_{uCM} N_{u\_data} + N_{dCM} N_{u\_ack}) \times t_{ms} \qquad (5)$$

(2) when $N_{dCM} > 2N_{p\_REQ}$, T_{usv} is bounded by

$$(N_c + N_{uCM} N_{u\_data} + N_{dCM} N_{u\_ack})\, t_{ms} \le T_{usv} \le \frac{[N_c + N_{uCM} N_{u\_data} + (N_{dCM} - N_{p\_REQ}) N_{u\_ack}]\, t_{ms} N_{dCM}}{(N_{dCM} - N_{p\_REQ})} \qquad (6)$$

Substituting eqs. (5) and (6) into (2), we can derive the two bounds of *k* for the two cases accordingly.

3. Long Packet Deferment

In DOCSIS, the CMTS allocates at most one data grant to each client in a MAP, irrespective of the number of mini-slots requested. The design philosophy behind "one-CM-one-Data-IE" in a MAP is to fairly share the channel bandwidth, i.e. no one will monopolize all of the bandwidth in a transmission period. Such fairness, however, holds only for one-way TCP transfers. With two-way TCP transfers, both data and ACK packets are transmitted upstream. Typically, TCP packets are ten times larger than ACK packets. Thus, both types of packets require different sizes of data grants to be allocated by the CMTS. [5] showed that with two way transfers, the upstream TCP traffic (i.e., long data packets) might throttle the downstream traffic (i.e., short ACK packets). The reasons are as follows.

(1) Long data packets going upstream cause high asymmetric ratio, and thus long round trip time, to short ACK packets. Long round trip delay in turn reduces the growth rate of the downloading CM's congestion window because the congestion window of TCP grows at a rate inversely proportional to average round trip delay.
(2) The "one-CM-one-Data-IE" allocation of DOCSIS leads to both upstream and downstream transfers experiencing the same round trip delay. Thus, the congestion windows of these two types of transfers grow at approximately the same rate. This results in low bandwidth utilization on the downstream channel because the downstream channel typically has far higher capacity than the upstream one. Should these two types of TCP transfers grow their congestion windows at the same rates, it would underutilize the bandwidth on the downstream channel.

To summarize, DOCSIS treats TCP data packets the same way as ACK packets on the upstream channel. It allows each CM to send at most one data grant in every MAP, irrespective of its frame size. This causes the same round trip delays for both downstream and upstream transfers, and results in poor bandwidth utilization on the downstream channel. To solve this problem, we propose a mechanism called "Long Packet Deferment" (LPD), which treats long packets (data packets) differently from short packets (ACK packets) on the upstream channel. The design goal of the LPD protocol is to reduce the sending rates of long packets and increase those of short packets, in an attempt to achieve true fairness in resource sharing. Thus, we can shorten the round trip delay of downstream TCP transfers without seriously degrading the performance of upstream TCP transfers. Note that the LPD mechanism is designed to operate at the CMTS only. No modification is required on CMs.

3.1 LPD Fundamentals

We assume all data packets are of fixed size, and distinguish between two types of requests only (i.e., for long and short data grants). Later we will extend to packets with variable size in Sec. 3.3. Let δ be the threshold to determine the type of a request, where $\delta = \frac{N_{u\_data} - N_{u\_ack}}{2}$. Intuitively, if the data grant size requested exceeds the threshold δ, the request should be deferred a few more steps before the allocation is granted. Let N_{def} be the number of steps a long packet should be deferred. Each downloading CM (i.e., those transferring short ACK packets) can get a data grant in every MAP, but each uploading CM (those transferring long data packets) can only get one data grant in every N_{def} MAPs.

Suppose that the CMTS have two types of queues to store requests: a long job queue and a short job queue. Upon receipt of a new request, the CMTS processes the request as follows. If the data grant size requested is larger than δ, the CMTS initializes the request's deferred step to N_{def} and puts it into the long job queue; otherwise, the number of deferred steps is set to one and the request is put into the short job queue. Later when it comes to the transmission time of the next MAP, the CMTS will start processing requests from the short job queue, followed by the long job queue, all on a first-come-first-served basis. The CMTS allocates data grants only to those requests with the number of deferred steps less than or equal to one, and removes the allocated requests from the respective queues. For those with larger-than-one deferred steps, the CMTS decrements the values of their deferred steps, issues them data pending IEs in the MAP, and puts them back into the queues. This process continues until the limitation of a MAP is reached (2048 mini-slots and 240 IEs), when the CMTS will stop both allocating data grants and decrementing the deferred steps. It will then start issuing the remaining eligible requests (i.e., those with deferred steps of less than or equal to one) data pending IEs on the MAP because the CMs should be notified that their requests are pending, not lost.

3.2 Analysis of the LPD Mechanism

The LPD mechanism makes downloading CMs experience different round trip delays from uploading CMs, with the ratio of the upstream packet sending rates of these two types of CMs being N_{def}:1. In addition, LPD forces the CMTS to place long data grants at the end of each MAP because it starts allocation from the short job queue. Thus, the LPD mechanism keeps the asymmetric ratio and TCP round trip delay at the lower bound if there is any long data grant allocated in a MAP.

(a) Asymmetric ratio

The LPD mechanism causes different round trip delays to downstream and upstream TCP transfers. We first derive the upstream data service times for both downloading and uploading CMs.

(1) For downloading CMs (i.e., ACK packet),

$$T_{usv} = (N_c + \frac{N_{uCM}}{N_{def}} \times N_{u\_data} + N_{dCM} N_{u\_ack})\, t_{ms} \qquad (7)$$

Since the number of mini-slots for a long data grant (for TCP packet) is much larger than the number of mini-slots for a short data grant (for ACK packet), $N_{u\_data} >> N_{u\_ack}$ and $N_{def} >> 1$. Comparing eqs. (7) with (5) and (6), we see that the upstream data service time of the downloading CM is significantly reduced.

(2) For uploading CMs (i.e., data packet),

$$T_{usv} = (N_{def} N_c + N_{uCM} N_{u\_data} + N_{def} N_{dCM} N_{u\_ack})\, t_{ms} \qquad (8)$$

Comparing eqs. (8) with (5) and (6), we see that the LPD mechanism may slightly increase the upstream data service time of the uploading CM.

The asymmetric ratio k of downstream TCP transfers for LPD can be derived as

$$k = \frac{N_{def} N_{dCM} L_{data}}{N_{def} N_{dCM} L_{data} + \frac{N_{uCM} L_{data}}{d}} \times \frac{C_d}{d \times L_{data}} \times$$
$$\frac{[N_c + \frac{N_{uCM}}{N_{def}} \times N_{u\_data} + N_{dCM} N_{u\_ack}]\, t_{ms}}{N_{dCM}} \qquad (9)$$

Compared with the original scheduling (i.e., simple FCFS), k in eq. (9) is smaller due to smaller upstream data service time. A smaller k will speed up the dropping of the value of k to one, when the system will behave normally, i.e., as in the symmetric network. This in turn shortens the round trip delay, which results in larger downstream throughout.

(b) Round trip delay

With two-way transfers, the system is mostly operated as in asymmetric networks. Eq. (10) shows the average round trip delay of sending a packet as k>1.

$$RTT = 2T + T_{trans} + B_{CM} \times T_{usv} \tag{10}$$

where B_{CM} is the buffer size of the CM, and

$$T_{trans} = \begin{cases} \frac{L_{data}}{C_d} + \frac{L_{ack}}{C_u}, \text{for downstream traffic} \\ \frac{L_{data}}{C_u} + \frac{L_{ack}}{C_d}, \text{for upstream traffic} \end{cases}$$

Substituting eqs. (7) and (8) into (10), we can derive the average round trip delays of downstream TCP transfers and upstream TCP transfers accordingly.

The last term of eq. (10) dominates RTT. Thus, LPD has far shorter round trip delay for the downstream TCP transfer compared to the original scheduling (i.e., simple FCFS) but slightly longer round trip delay for the upstream TCP transfer.

3.3 The Number of Deferred Steps

In this section, we determine the number of steps a packet should be deferred and generalize LPD to handle data packets with variable size. Let L_x be the data grant size of a new request R_x. There are two considerations to determining D_x, the number of steps request R_x should be deferred before being granted.

(1) Balance the data rate.
If we let $D_x = \left\lfloor \frac{L_x}{N_{u\_ack}} \right\rfloor$, the downloading and uploading TCP transfers will have the same transmission rate on the upstream channel. $\left\lfloor \frac{L_x}{N_{u\_ack}} \right\rfloor$ should be the upper bound of D_x, because if D_x is set to a larger value, the mechanism will penalize long packets. Actually, if we let $D_x = \left\lfloor \frac{L_x}{N_{u\_ack}} \right\rfloor$, the system behaves like an IEEE 802.14 mechanism, which has fixed PDUs. Thus, D_x should be set to a value smaller than this bound because DOCSIS is benefited from supporting variable length PDUs [6].

(2) Balance the round trip delay.
We try to let the downstream and upstream TCP transfers fill up the downstream and upstream channels at the same rate. Since the TCP congestion window grows at a rate inversely proportional to its round trip delay,

$$\frac{\frac{1}{RTT_d} N_{dCM}}{\frac{1}{RTT_u} N_{uCM}} = \frac{C_d}{C_u} \tag{13}$$

In addition, the last term in eq. (10) dominates RTT. Thus, we have

$$D_x = N_{def} = \frac{N_{uCM}}{N_{dCM}} \frac{C_d}{C_u} \tag{14}$$

Typically, N_{uCM} is smaller than N_{dCM}. Both values are usually very hard to determine dynamically.

From the discussion above, D_x should be upper bounded by $\left\lfloor r\frac{L_x}{N_{u\_ack}} \right\rfloor$ and $\left\lfloor r\frac{C_d}{C_u} \right\rfloor$, where $0 < r < 1$. In the LPD protocol, we set the number of deferment groups $n = \left\lfloor r\frac{C_d}{C_u} \right\rfloor$, where $0 < r < 1$[1]. Each group corresponds to a certain range of data grant size. D_x of request R_x is determined as follows.

(1) If $0 < L_x < \frac{2}{r} N_{u\_ack}$, $D_x = 1$ and R_x is put to the queue of group one.
(2) If $\frac{k}{r} N_{u\_ack} \le L_x < \frac{k+1}{r} N_{u\_ack}$, $k = 2,3\cdots, \left\lfloor r\frac{C_d}{C_u} \right\rfloor - 1$, $D_x = k$ and R_x is put to the queue of group *k*.
(3) If $L_x \ge \left\lfloor \frac{C_d}{C_u} \right\rfloor N_{u\_ack}$, $D_x = \left\lfloor r\frac{C_d}{C_u} \right\rfloor$ and R_x is put to the queue of group *n*.

Usually, $\left\lfloor r\frac{C_d}{C_u} \right\rfloor$ is not very large. Thus, this mechanism will be very efficient.

The LPD protocol is generalized as follows. The CMTS maintains *n* deferment queues to delay requests asking for different data grant sizes. Each time a request arrives at the CMTS, it is processed as follows. According to the data grant size requested, the CMTS assigns a number of deferred steps to the request and put it to the corresponding queue. When it comes to the next MAP, the CMTS allocates the data grants if the event has been deferred enough steps, in the order from the shortest job queue to the longest job queue until the MAP limitation is reached. This gives shorter packets higher priorities to send, and adjusts the data rates of downloading and uploading CMs accordingly.

4. Simulation

This section presents the simulation results using ns2 for the LPD mechanism, the 802.14-like mechanism (denoted as L2S in the figures), and the original DOCSIS control mechanism.

4.1 Simulation Environment

The simulation parameter setting is shown in Table 1. We

[1] From the simulation, r=0.5 can achieve the best balance in throughputs of both upstream and downstream transfers.

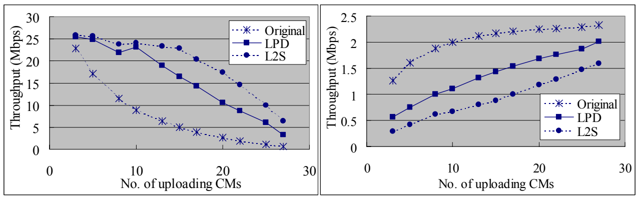
use two-way TCP bulk transfers as traffic sources and each CM can have only one TCP transfer at a time.

Parameter	Value
C_d	26.97 Mbps
C_u	2.56 Mbps
T	0.5 ms
D_{MAP}	2 ms
t_{ms}	50 us
N_c	50 (mini-slots)
MAP limit	2048 mini-slots and 240 IEs
L_{data}	1024 bytes
L_{ack}	64 bytes
B_{CM}	20 (packets)

Table 1 Parameter Setting

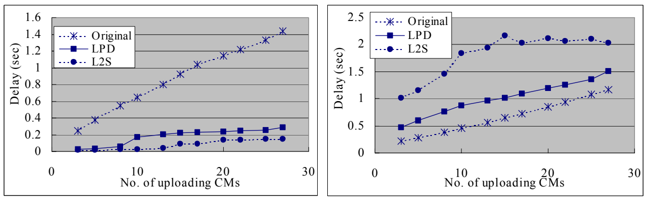
4.2 Simulation Results

Fig. 1 shows the aggregate throughputs of downstream (Fig. 1 (a)) and upstream transfers (Fig. 1 (b)), varying uploading CMs from 1 to 30. The dotted lines are for the original DOCSIS; the solid lines with squares and circles are for LPD and L2S, respectively. Fig. 1 (a) shows that both LPD and L2S have about 80% improvement of aggregate downstream bandwidth over the original DOCSIS when less than one-third of CMs are doing upstream transferring. They have slight performance degradation on the upstream direction as shown in Fig. 1 (b) (note that Figs. 1 (a) and (b) have different scales in throughput). LPD has better improvement than L2S. Thus, the number of deferment groups does not need to be very large. Overall, LPD has the best balance between downstream and upstream bandwidth utilization when the proportion of uploading CMs is between 1/3 and 5/6. Since typically CMs perform downloads in HFC networks, it is rare that more than 5/6 of CMs are uploading. The region of interest is when the proportion of uploading CMs is less than 5/6, when performance improvement is observed.



(a) Downstream (b) Upstream
Figure 1. Aggregate throughput

Fig. 2 shows the access delays of downloading CMs (Fig. 2 (a)) and uploading CMs (Fig. 2 (b)), for the three mechanisms as a function of simultaneous uploading CMs. It shows that of the three mechanisms, simple FCFS (i.e., the original DOCSIS) has the largest access delay for downloading CMs, and the 802.14-like mechanism (i.e., L2S) has the largest access delay for uploading CMs. The LPD mechanism renders the access delays of downloading CMs much shorter, without significantly increasing the access delays of uploading CMs.



(a) Downstream (b) Upstream
Figure 2. Access delay

5. Conclusion

This paper proposes a new mechanism called "Long Packet Deferment" to significantly improve the performance of TCP over the DOCSIS MAC control mechanism. The LPD mechanism is aimed at reducing the negative impact of the DOCSIS MAC layer operation on bandwidth asymmetry, which improves the performance of TCP. With LPD, only the modification to the CMTS is required. We have also conducted simulations using network simulator *ns-2* to compare the original control mechanism of DOCSIS with our mechanism. The results show that the proposed LPD mechanism has better performance in terms of both aggregate downstream throughput and access delay.

Acknowledgement

This work was supported by the Communications Software Technology project of the Institute for Information Industry (III) and sponsored by Ministry of Economic Affairs (MOEA), R. O. C.

References

[1]. H. Balakrishnan, V. N. Padmanabhan and Randy H. Katz, "The effects of asymmetry on TCP performance," Mobile Networks and Applications 4, p219-p241, 1999.

[2]. Yi-Der Li and Wanjiun Liao, "Improving TCP Performance over Asymmetric Networks," Proc. IEEE ICC 2001, Helsinki, Finland, June 2001.

[3]. S. Varma, "Performance and Buffering Requirements of TCP Applications in Asymmetric Networks, " Proc. IEEE INFOCOM '99, 1999.

[4]. T. V. Lakshman, U. Madhow and Bernhard Suter, "Window-based Error Recovery and Flow Control with A Alow Acknowledgement Channel: A Study of TCP/IP Performance," Proc. IEEE INFOCOM '97, 1997.

[5]. Hue-Jiun Ju and Wanjiun Liao, "Fast Request Transmission in DOCSIS-Based CATV Networks," to appear in Proc. IEEE International Conference in Multimedia and Expo (ICME), Aug, 2002.

[6]. M. T. Ali, R. Grover, G. Stamatelos, and David D. Falconer, "Performance Evaluation of Candidate MAC Protocols for LMCS/LMDS Networks," IEEE Journal on Selected Areas in Communications, Vol. 18, No. 7, July 2000