A Call Admission Control Algorithm for Long-lived and Short-lived Streaming Media [*]

Yi-jung Lo
Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan
r91942028@ntu.edu.tw

Zsehong Tsai
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
ztsai@cc.ee.ntu.edu.tw

Abstract

Call admission control (CAC) is one important QoS mechanism for maintaining performance of streaming media services. In this paper, we propose a novel CAC algorithm, which handles long-lived and short-lived streams differently to satisfy their different QoS requirements. Under such mechanism, we show that not only one can guarantee the quality of each stream, but also improve the system utilization.

1. Introduction

The most common CAC algorithm design approaches used for streaming media servers have included utilization-based[1], measurement-based[2], and delay bound-based[3]. Nonetheless, many CAC algorithms adopted in Internet are hybrid[4]. Besides, almost all existing CAC algorithms regard each stream as long-lived; that is, the lifetime of a stream is not considered. If the lifetime of a newcomer is too short to suffer the predicted delay bounds, actually one can admit it while keeping QoS for all admitted streams.

In this paper, we focus on such problem and then divide Internet traffic streams into two categories: long-lived and short-lived streams, which are corresponding to different web and Internet streaming media. In this algorithm, long-lived and short-lived streams are handled differently to satisfy their different QoS requirements and to fit their traffic characteristics.

* This work is sponsored by National Science Council under grant NSC093-2213-E-002-056.

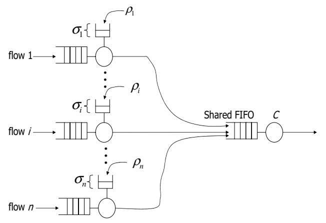

Figure 1. Queueing Model

2. System Models

In [5], we can know that during the buffering period, a streaming media server sends packets at a faster rate until the client's buffer is full and then the server will slow down the transmission to a sustainable rate. We call such scenario as Fast Start. Additionally, packets may not arrive periodically because delay jitters might happen in the inside of servers. Thus, the throughput of media streams may be disrupted for short moments and the decreases of throughput would cause a burst arrival later. Hence, we need other parameters to describe its traffic characteristic.

Besides the above discussion, the file size of a stream should be also concerned. Therefore, we employ five parameters, (R_i, r_i, t_i, X_i, $LIFE_i$), to describe the traffic pattern of stream i, where R_i is the rate during Fast Start period, r_i is the sustainable average rate, t_i is the period length of Fast Start (i.e., the buffer size of the client i is $R_i t_i$), X_i is the maximum burst size, and $LIFE_i$ is the lifetime, equivalently the length of playback.

As for our proposed queueing model, we employ leaky buckets[6] as the per-stream shaper and a shared FIFO as the scheduler. The parameters are: ρ_i, the token generation rate of shaper i, σ_i, the token bucket depth of shaper i, and C, the service rate of the shared FIFO. Last but not least, the buffer of each shaper or FIFO is assumed infinite.

Furthermore, To reduce computation and storage complexity, we arrange the key traffic shape parameters proportional to its average rate and hypothesize that both the Fast Start period length and the maximum burst have their upper bounds respectively. That is,

$$R = \frac{R_1}{r_1} = \cdots = \frac{R_i}{r_i} = \cdots = \frac{R_n}{r_n} \quad (1)$$

$$\rho = \frac{\rho_1}{r_1} = \cdots = \frac{\rho_i}{r_i} = \cdots = \frac{\rho_n}{r_n} \quad (2)$$

$$\sigma = \frac{\sigma_1}{r_1} = \cdots = \frac{\sigma_i}{r_i} = \cdots = \frac{\sigma_n}{r_n} \quad (3)$$

$$t \geq t_1, \ldots, t_i, \ldots, t_n \quad (4)$$

$$X \geq \frac{X_1}{r_1}, \ldots, \frac{X_i}{r_i}, \ldots, \frac{X_n}{r_n} \quad (5)$$

where R is the common Fast Start data rate ratio, ρ is the common normalized token generation rate, σ is the common normalized leaky bucket depth, t is the maximum length of Fast Start period and X is the maximum burst ratio. By such arrangement, besides lower complexity, large number of parameters is avoided as the number of streams increases.

3. Performance Analysis

3.1. Service Curves

Usually, we set $r_i < \rho_i < R_i$ and $\sigma_i \leq X_i$ for stable operations. Therefore, the service curve will be "shaped". We let T_i be the time length that shaper i needs to finish transporting $R_i t_i$ bits in the Fast Start period and T be the upper bound of T_i. It is convinced that T_i reaches T when there is a maximum burst arrival during the Fast Start period. Thus, after some simple calculatoin, we can get

$$T = \frac{X}{R} + \frac{Rt - \sigma}{\rho} \geq T_i. \quad (6)$$

Next, regardless of Fast Start, let T_{si} be the maximum delay in the shaper of a packet belonging to stream i and T_s be the upper bound of T_{si}. One can derive

$$T_{si} = \frac{X_i - \sigma_i}{\rho_i} \leq \frac{X - \sigma}{\rho} = T_s. \quad (7)$$

Additionally, let T_{fi} be the time that shaper i needs to clear its queue while the maximum burst arrives and T_f be the upper bound of T_{fi}. Hence,

$$T_{fi} = \frac{X_i - \sigma_i}{\rho_i - r_i} \leq \frac{X - \sigma}{\rho - 1} = T_f. \quad (8)$$

By observing Eq. (6)$\sim$(8), we can know that all the three bounds are independent of the index i.

3.2. Time-to-Play

Time-to-Play is an important performance metric. To introduce the following, we need some definitions first. Let TP_i be the period length that the proposed queueing model needs to finish transporting $R_i t_i$ bits for stream i during the Fast Start period. And let $delayFS_i$ be the period length that the proposed queueing model needs to finish transporting $R_i t_i$ bits for stream i during the Fast Start period. And let TP be the maximum Time-to-Play that a customer can tolerate, which depends only on the client's endurance and is independent of system states. Thus, by the definition of T_i, TP_i, and $delayFS_i$, we can conclude that

$$TP_i = T_i + delayFS_i. \quad (9)$$

Next, we denote the newcomer as stream n and only consider the overbooked traffic condition, i.e., $\frac{C}{\rho} \leq \sum_{i=1}^{n-1} r_i < C$. Also, we are interested in F_T, the long-term reserved bandwidth for long-lived streams. By standard utilization-based CAC approach and reserving its maximum bandwidth required, F_T is derived to be

$$F_T = \frac{C(TP + T_f - T) - r_n Rt}{X + T_f}. \quad (10)$$

Theorem 1. *If $\sum_{i=1}^{n-1} r_i \leq F_T$, then $TP_n \leq TP$.*
Proof:

In [6], the authors have concluded that the maximum queue size of FIFO happens when every stream is "greedy" at the same time. Thus, the maximum queue size of FIFO becomes $(X + T_f)\sum_{i=1}^{n-1} r_i - CT_f + r_n Rt$ and by Eq. (10)

$$\begin{aligned} C \cdot delayFS_n \leq & (X + T_f)\sum_{i=1}^{n-1} r_i - CT_f + r_n Rt \\ \leq & C(TP - T). \end{aligned} \quad (11)$$

By Eq. (6), (9), and (11), TP_n and TP are related by

$$TP_n = T_n + delayFS_n \leq TP.$$

Q.E.D.

Subsequently, we will introduce another theorem for other cases. Let Q be the notation of the queue size of FIFO and $DATA_T$ be the maximum total data amount that all streams except stream n can send in T seconds.

Theorem 2. *When $F_T < \sum_{i=1}^{n-1} r_i < C$ and $(Q+DATA_T+r_nRt) \leq (C \cdot TP)$, $TP_n \leq TP$.*
Proof:

By deriving the delay bound of $delayFS_n$, $C \cdot delayFS_n \leq (Q + r_n Rt + DATA_T) - CT$. Hence, under the given condition, the right-handed side will lead to $delayFS_n \leq TP - T$. Hence, we can conclude that $TP_n = T_n + delayFS_n \leq TP$.

Q.E.D.

By Theorem 1 and 2, we can know that under the stable condition, TP_n is bounded.

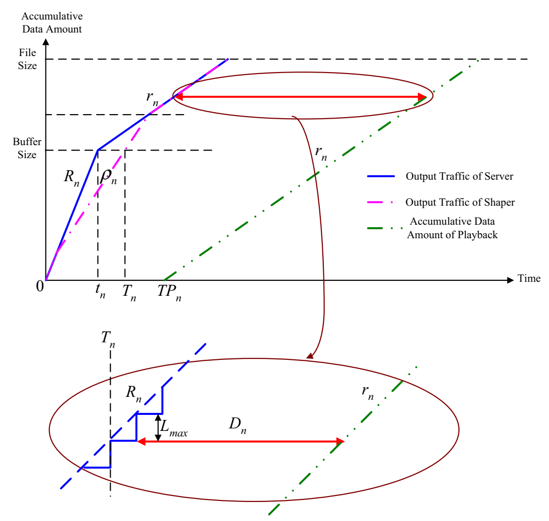

Figure 2. Maximum Tolerable Delay

3.3. End-to-end Delay

To simplify our problem, we need to first define end-to-end delay precisely and the delay bound. Let the end-to-end delay of an arbitrary packet belonging to stream i, d_i, defined as the time that the packet needs to go through the proposed queueing model. And let the delay bound of an arbitrary packet belonging to stream i, D_i, defined as the maximum end-to-end delay that the customer can tolerate.

Figure 2 illustrates a simple example. The client plays the media at time TP_n at rate r_n. As we said before, $r_n Rt$ is the client's buffer size and it is believed that Rt is delay jitter tolerance. Therefore, D_n seems equal to Rt. However, $D_n = Rt$ holds only in fluid models.

The lower part of Figure 2 is shown by a larger scale so that we can observe packet transmission clearly. It can be observed that after the Fast Start period, the maximum horizontal distance between the two service curves minus the packet transmission time is the delay bound, D_n. After some simple calculation, we can know that D_n satisfies

$$D_n = Rt - \frac{L}{r_n}, \tag{12}$$

where L is the packet size. Then, let the target stream be stream j. In the former sectoin, we know that the maximum delay in shaper is T_s and the maximum queue size of FIFO is $(X + T_f)\sum_{i=1}^{n} r_i - r_j Rt - CT_f$. We let F_j replace $\sum_{i=1}^{n} r_i$ and let d_j equal to D_j. Thus, we will get

$$F_j = \frac{C\left(Rt - \frac{L_{max}}{r_j} + T_f - T_s\right) + r_j T_f}{X + T_f} \tag{13}$$

We can define F_j as the minimum system utilization that leads d_j to reach D_j. Next, we show that if the lifetime of a short-lived stream is short enough, the end-to-end delay of its any packets will never exceed its delay bound.

Theorem 3. *If $LIFE_n < LIFE_{ss}$, d_n will never exceed D_n, where $LIFE_{ss}$ is given by*

$$\begin{aligned} LIFE_{ss} = &\frac{1}{(\rho - 1)(C - r_n)}\{C[D_n - TP - (R-1)t] \\ &+ \rho[T - T_s + (R-1)t](C - r_n) + r_n Rt\}. \end{aligned}$$

Proof:

The queue length of FIFO at time $(LIFE_n - (R-1)t + T_s)$, the moment that the last packet of stream n enters FIFO at, is $C(TP - T) + \left(\rho \sum_{i=1}^{n-1} r_i - C\right)[LIFE_n - (R-1)t + T_s - T] + r_n(LIFE_n - Rt)$ and it must be less than or equal to $C(D_n - T_s)$. Hence,

$$\begin{aligned} LIFE_n \le &\frac{1}{\rho \sum_{i=1}^{n-1} r_i - C + r_n}\{C[D_n - TP - (R-1)t] \\ &+ \rho[T - T_s + (R-1)t]\sum_{i=1}^{n-1} r_i + r_n Rt\}. \end{aligned} \tag{14}$$

It is obvious that the larger the value of $\sum_{i=1}^{n-1} r_i$, the larger the value of right handed-side of Eq. (14). Under stable condition, $\sum_{i=1}^{n-1} r_i$ can not exceed $C - r_n$. Thus, by using the above properties to derive the upper bound of right-handed side of Eq. (14), one can derive

$$LIFE_n < LIFE_{ss}. \tag{15}$$

Therefore, if Eq. (15) holds, d_n will never exceed D_n.

Q.E.D.

3.4. The LS Call Admission Control Algorithm

We call the proposed algorithm "the Long-lived and Short-lived (LS) CAC Algorithm." The detailed decision process is as following. First, one should distinguish the incoming request is for a long-lived or a short-lived stream. Second, for long-lived streams Theorem 1 is used and for short-lived streams one should use Theorem 2 and 3. Last, one should update system states if necessarily.

4. Simulation Results

4.1. Parameters of Streams and Shapers

In our simulation, non-preemptive higher priority is granted to packets of long-lived streams than the priority of the packets of short-lived streams. Meanwhile, we have six kinds of average rate: 6Mbps, 3Mbps, 1.5Mbps, 700kbps, 500kbps, and 300kbps. Moreover, the lifetime of a long-lived stream is infinite and that of a

	The Number of Accepted Long-lived Flows	The Number of Accepted Short-lived Flows	Prob($TP_i > TP$)	The Average of TP_i
No-control Case	41	733	0.3023	850 sec
Utilization-based	41	0	0.00	5.7 sec
LS (Long-lived and Short-lived)	41	195	0.00	5.8 sec

Table 1. Comparison of Time-to-Play

	# of Packets	Prob($d_i > D_i$)	Average End-to-end Delay
No-control Case	29,954,679	0.1559	185sec
Utilization-based	25,000,554	0.00	0.32sec
LS (Long-lived and Short-lived)	27,996,830	0.00	0.33sec

Table 2. Comparison of End-to-end Delay

short-lived stream can be 24, 30, 120, 150, 240, or 300 seconds. As for other designed parameters, C is 100Mbps, L is 1500 bytes, R is 1.5, t is 3.6, X is 4.2, ρ is 1.2, σ is 0.12, and *TP* is 8 seconds.

Besides, the traffic pattern of each stream is an ON-OFF model. While it is ON, the server sends packets periodically; otherwise, it will not send any packet. All packets accumulated during OFF are sent at the start of the next ON period. Moreover, the state sojourn times of ON and OFF are both exponential-distributed. The mean sojourn time of ON is 40 seconds and that of OFF is 4 seconds.

Finally, in order to observe congestion, long-lived streams occupy all bandwidth after the start. Therefore, the number of on-going long-lived streams(700kbps) is 6 and the number of other streams is 7. Also, the short-lived stream request arrives according to a Poisson process, with a mean inter-arrival time equal to 5 seconds.

We will compare the LS CAC algorithm with the utilization-based CAC algorithm and the no-control case according to Time-to-Play, end-to-end delay, and system utilization. We calculate the system utilization every 0.15 seconds and the system utilization is defined as *System Utilization*=$\frac{\text{Actual Data Rate}}{C}$. And, this simulation lasts for 3600 seconds.

4.2. Numerical Results

The results are illustrated in Table 1, 2, and 3. By observing the comparison of the three aspects, it is obvious that the system utilization of the LS CAC algorithm is lower than that of the no-control case a little but higher than that of the utilization-based algorithm a lot. And the performance of the LS CAC algorithm is almost the same as that of the utilization-based algorithm and much better than that of the no-control case. Hence, in summary, we can know that the LS CAC algorithm provides the best balanced performance.

	Average Utilization	Max Utilization
No-control Case	0.9985	1.00
Utilization-based	0.8334	1.00
LS (Long-lived and Short-lived)	0.9332	1.00

Table 3. Comparison of Utilization

5. Conclusions

We have proposed a novel method, the LS CAC algorithm, to increase system utilization by taking into account the lifetime of streams, which is very different from existing utilization-based, measurement-based, or delay bound-based CAC algorithms. The proposed architecture and algorithm are easily to implement since its computation and storage complexity are also in proportional to the number of streams.

References

[1] A. S. Tanenbaum, *Computer Networks*, Englewood Cliffs, NJ: Prentice Hall, 3 edition, 1996.

[2] M. Grossglauser and D.N.C Tse, "A time-scale decomposition approach to measurement-based admission control," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 550–563, Aug. 2003.

[3] M. Li and Z. Tsai, "A traffic shaper for supporting cbr and vbr services in atm networks," *Performance Evaluation*, pp. 243–264, Jan. 2000.

[4] W.-J. Hsu and Z. Tsai, "A call admission control algorithm based on stochastic performance bound for wireless networks," in *the 3rd IEEE Pacific Rim Conference on Multimedia 2002*, Dec. 2002.

[5] Microsoft Windows Digital Media Division, "An introduction to windows media services 9 series," Microsoft Corporation, Sept. 2002.

[6] A. K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.