Probabilistic Chunk Scheduling Approach in Parallel Multiple-server DASH

Li Liu †, Chao Zhou †, Xinggong Zhang †*, Zongming Guo †, Cheng Li #

† *Institute of Computer Science & Technology, Peking University, Beijing, P.R. China*
# *Freewheel*

***Abstract*—Recently parallel Dynamic Adaptive Streaming over HTTP (DASH) has emerged as a promising way to supply higher bandwidth, connection diversity and reliability. However, it is still a big challenge to download chunks sequentially in parallel DASH due to heterogeneous and time-varying bandwidth of multiple servers. In this paper, we propose a novel probabilistic chunk scheduling approach considering time-varying bandwidth. Video chunks are scheduled to the servers which consume the least time while with the highest probability to complete downloading before the deadline. The proposed approach is formulated as a constrained optimization problem with the objective to minimize the total downloading time. Using the probabilistic model of time-varying bandwidth, we first estimate the probability of successful downloading chunks before the playback deadline. Then we estimate the download time of chunks. A near-optimal solution algorithm is designed which schedules chunks to the servers with minimal downloading time while the completion probability is under the constraint. Compared with the existing schemes, the experimental results demonstrate that our proposed scheme greatly increases the number of chunks that are received orderly.**

***Index Terms*—DASH; Multiple Servers; Probabilistic Approach; Chunk Request Scheduling**

I. INTRODUCTION

In recent years, Dynamic Adaptive Streaming over HTTP (DASH) has been widely used for video streaming over the Internet[1], [2], [3], [4]. Besides well-known single-server DASH, parallel DASH also emerges recently as a promising technology for video streaming[6], [7]. It employs multiple connections with different CDN or servers to improve streaming robustness and available bandwidth. In parallel DASH, video content is encoded into multiple versions at various bit-rates[5]. Each video version is further broken into small video chunks, which normally contains a few seconds worth of video. Using HTTP connections, DASH client is able to download chunks from multiple servers concurrently. Compared with the traditional single-server scheme, parallel DASH with multiple servers is able to supply higher bandwidth, connection diversity and reliability. However, it is still a big challenge to download chunks sequentially because of multi-servers' diversified bandwidth.

In parallel DASH, video chunks are requested from multiple servers in parallel, and the request scheduling plays a critical role in guaranteeing video chunks playback in sequence. In [7], the chunk-server scheduling scheme is same as that in single-server DASH. The video chunks are requested orderly from multiple servers (i.e., one after another). However, the download completion time may be out-of-order due to servers' heterogeneous bandwidths. On the other hand, the video chunks must be played orderly at the clients. Thus, this method may deteriorate the continuity of video playback.

In this paper, we propose a novel probabilistic chunk scheduling approach considering time-varying bandwidth of multiple servers. Video chunks are scheduled to the servers which consume the least download time while have the highest probability to complete downloading before the deadline. The scheduling approach is formulated as a constrained optimization problem with the objective to minimize the total downloading time. Using the probabilistic model of time-varying bandwidth, we first estimate the probability of successful chunks downloading before the playback deadline. Then we estimate the download time of chunks. A near-optimal solution algorithm is designed which schedules chunks to the servers with minimal downloading time while the completion probability is under the constraint.

The main contributions of this paper can be summarized in two-fold:

1) We propose a probabilistic approach to schedule chunk requests in parallel multi-server DASH. Our method considers time-varying bandwidth, and aims to ensure chunks downloading in sequence in parallel DASH under the condition that the chunks are completed before their playback deadline.
2) We formulate the approach into an constrained optimization problem. A near-optimal solution is designed that the chunks are scheduled by the estimated downloading time and completion probability.

The rest of the paper is organized as following. The formulation of the problem are presented in section 2. In section 3, the optimization problem is further studied and then our

*Corresponding author.Email:zhangxg@pku.edu.cn

This work was supported by National High-tech Technology R&D Program (863 Program) of China under Grant 2013AA013504 and National Natural Science Foundation of China under contract No. 61271020.

978-1-4799-6139-9/14/$31.00 ©2014 IEEE

scheduling algorithms are proposed. Experiments setup and emulation results are presented in section 4. We conclude the paper in section 5.

II. Problem Formulation

In this section, we will introduce the problem of chunk request scheduling in parallel multi-server DASH, and give its optimization formulation.

In parallel multi-server DASH, a client builds connections with multiple servers, and downloads chunks concurrently. Although parallel downloading could exploit diversities of multi-server capacity, time-varying bandwidth makes it a big challenge to download chunks in sequence from multiple servers. When a chunk close to deadline is assigned to a wrong server, it would result the chunk missing the playback deadline and the playback is interrupted. To address the problem, it is a good choice to predict server capability accurately and schedule chunks by the available bandwidth. But as demonstrated in [8], the Internet bandwidth is time-varying which is hardly predicted accurately. Thus, we propose a probabilistic scheduling method to solve this problem.

In the proposed probabilistic scheduling method, chunks are scheduled by completion probability and average download time. The download completion probability is defined as the probability that a chunk is successfully received before its playback deadline. The average download time is the average downloading complete time. We would like to assign chunks to the servers which consume the smallest download time while have the highest probability to complete downloading before the deadline.

In parallel DASH, all versions of video are segmented into equal-length chunks. For the chunks to be scheduled, we define N as the number of video chunks and S as the number of servers. Let $\mathbf{M}$ be the indicator matrix with size $N \times S$, its entries $m_{ij} = 1$ if the request for chunk i is assigned to server j, otherwise $m_{ij} = 0$. For each chunk $n(1 \leq n \leq N)$, we define t_n as the average download time, and p_n as the successful download probability. The chunk scheduling problem can be formulated as:

$$\begin{aligned} \mathbf{M}^* &= \arg\min_{\mathbf{M}} \sum_{N} t_n(\mathbf{M}) \\ s.t. \quad & p_n(\mathbf{M}) \geq \eta, \forall n \leq N \end{aligned} \quad , \tag{1}$$

where η represents a given threshold value. The problem aims to find the optimal server for each chunk n so as to ensure that the chunks can be completely downloaded at the earliest time under the constraint that the chunks are successfully received before their playback deadline.

In the next section, we will analyze the probability $p_n(\mathbf{M})$ and the time $t_n(\mathbf{M})$ used in the above optimization problem, and a near-optimal solution algorithm is designed. In Table I, some symbols and their explanations used in the paper are listed.

TABLE I: Major symbols used in the paper

b_n	video bitrate of chunk n
Z	chunk size in time unit
L	the playback deadline
$\mathbf{M}$	the indicator matrix with size $N \times S$, its entries $m_{ij} = 1$ if the request for chunk i is assigned to server j, otherwise $m_{ij} = 0$
$t_n(\mathbf{M})$	the average download complete time for chunk n downloaded from server s
$p_n(\mathbf{M})$	the estimated probability that chunk n is completely downloaded from server s before its playback deadline
D_s	the sum of data amount to transfer for server s before the deadline if the request for chunk n is assigned to server s
$C_s(t, t+\tau)$	the average available bandwidth of the connection between client and server s during the time interval $(t, t+\tau)$

III. Analysis and Scheduling Algorithms

A. Analysis

To select the best server s^* for chunk $n(1 \leq n \leq N)$, we firstly need to calculate $p_n(\mathbf{M})$ and $t_n(\mathbf{M})$ for each server $s(1 \leq s \leq S)$. We begin with the analysis of $p_n(\mathbf{M})$.

All versions of video are broken into equal-length chunks, each of which contains the same playback time of Z. The video bit-rate of chunk n is denoted as b_n. Let ε_i denote the completion proportion of chunk i that has been requested before the time instant t_0, obviously, $\varepsilon_i \in [0, 1]$. Thus, if chunk i is assigned to server s, the remaining data of chunk i needs to be downloaded from server s is $(1-\varepsilon_i)Zb_i$. We define D_s as total data amount to be downloaded from server s if the request for chunk n is assigned to server s. Then we have

$$D_s = Zb_n + \Sigma_{i \in U}(1-\varepsilon_i)Zb_i, \tag{2}$$

where the second term of (2) is the data amount of chunk n, and U denotes the un-finished downloading chunks in server s.

We define $C_s(t)$ as the available bandwidth of the connection between client and server s around the time t. Obviously, chunk n can be successfully received from server s before the deadline L only if the total downloading traffic amount $\int_{t_0}^{t_0+L} C_s(t)$ is no less than D_s, which can be defined as

$$p_n(\mathbf{M}) = P(\int_{t_0}^{t_0+L} C_s(t) \geq D_s), \tag{3}$$

where the function $P(\cdot)$ return the probability value.

As demonstrated in the references [8], [9], the available bandwidth in a time slot τ can be viewed as a stationary and identically distributed stochastic process, and at any time instant t the process is described by the same random variable $C_s(t, t+\tau)$. Furthermore, $C_s(t, t+\tau)$ can be viewed as a Gaussian random variable with mean $\mu_{s,\tau}$ and variance ${\sigma_{s,\tau}}^2$. The mean does not depend on τ. The variance $\sigma^2_{s,\tau} = \mathrm{Var}[C_s(t, t+\tau)]$, however, depends strongly on τ. Under the assumption that $C_s(t)$ is independently and identically distributed, the variance decreases inversely proportional with the length of the measuring time duration, i.e. $\mathrm{Var}[C_s(t, t+k\tau)] = \mathrm{Var}[C_s(t, t+\tau)]/k$.

Thus, we can know that $C_s(t, t+L)$ is also a Gaussian random variable. The mean of $C_s(t, t+L)$ is equal to $\mu_{s,\tau}$, and its variance is determined as $\text{Var}[C_s(t, t+L)] = \sigma^2_{s,\tau}/(L/\tau)$. Finally, we can get $C_s(t, t+L) \sim (\mu_s, (\sigma^2_{s,\tau}\tau/L))$.

Let $F_s(\cdot)$ be the cumulative distribution function of $C_s(t, t+L)$, and we have

$$F_s(x) = P(C_s(t, t+L) \leq x), \tag{4}$$

$$= \Phi\left(\frac{x-\mu_{s,\tau}}{\sigma_{s,\tau}} * \sqrt{\frac{L}{\tau}}\right), \tag{5}$$

$$= \frac{1}{2}\left(1 + erf\left(\frac{x-\mu_{s,\tau}}{\sqrt{2}\sigma_{s,\tau}} * \sqrt{\frac{L}{\tau}}\right)\right), \tag{6}$$

where Φ denotes the cumulative distribution function of the standard normal distribution and $erf(\cdot)$ denotes the Gauss error function. Now, according to formulas (3), (4) and (6), we can determine $p_n(\mathbf{M})$ as

$$p_n(\mathbf{M}) = 1 - F_s(D_s/L), \tag{7}$$

$$= \frac{1}{2}\left(1 - erf\left((\frac{D_s}{L} - \mu_{s,\tau})\frac{1}{\sigma_{s,\tau}} * \sqrt{\frac{L}{2\tau}}\right)\right), \tag{8}$$

and

$$t_n(\mathbf{M}) = t + \frac{D_s}{\mu_s}, \tag{9}$$

Finally, we can get $p_n(\mathbf{M})$, $t_n(\mathbf{M})$ from (8), (9) respectively.

B. Scheduling Algorithms

For each candidate solution matrix $\mathbf{M}$, we firstly calculate the completion probability and average download time for each chunk n. And then, the chunks are assigned to the servers which consume the smallest average download time under the constraint that all chunks' probability is no less than the given threshold value. However, it is not impossible that no potential matrix $\mathbf{M}$ meets the constraint. In that case, the matrix with the largest average probability is chosen. Finally, chunks are assigned to servers as summarized in Algorithm 1.

In Algorithm 1, the Boolean variable $exist$ reflects that if there are matrixs that meet the constraint. Variable $exist$ is initialized to be false. For each potential matrix $\mathbf{M}$, we firstly calculate $p_n(\mathbf{M})$ and $t_n(\mathbf{M})$ for each chunk n. And if there is a matrix that meet the constraint, set $exist$ to be true. Then, the server $\mathbf{M}^*$ is selected according to $exist$.

Algorithm 1 Probabilistic Scheduling Algorithms

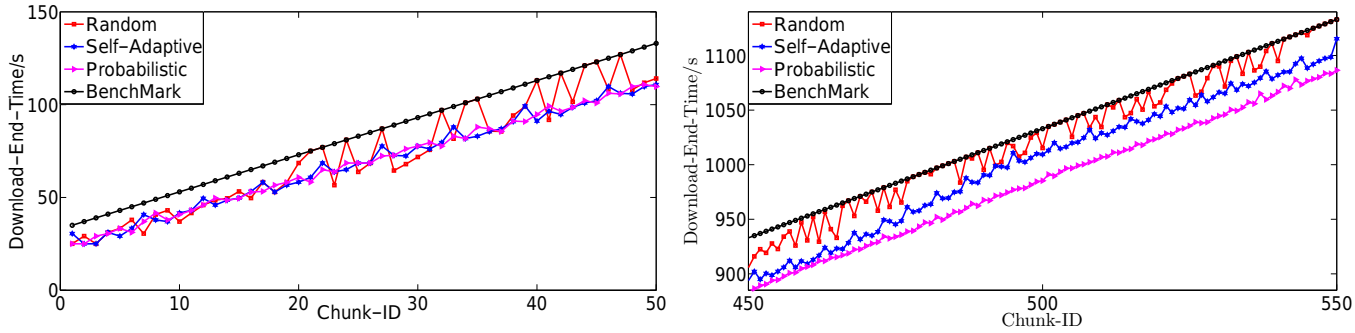
```
initialize exist ⇐ false
for each M do
    for n ⇐ 1 to N do
        initialize meet ⇐ true
        calculate p_n(M) with formula (8)
        calculate t_n(M) with formula (9)
        if p_n(M) < η then
            meet ⇐ false
        end if
    end for
    if meet then
        exist ⇐ true
    end if
end for
if exist then
        M* = arg min_M Σ_N t_n(M)
    s.t.  p_n(M) ≥ η, ∀n ≤ N
else
    M* = arg max_M Σ_N p_n(M)
end if
return
```

IV. Performance Evaluation

In this section, we evaluate our scheduling approach via real network traces. The bandwidth traces are extracted from three different servers. All traces are extracted during the same time with length of about 3000s. In our experiments, all three servers provide five different versions of video bitrates {3.0Mbps, 4.0Mbps, 5.0Mbps, 6.0Mbps, 7.0Mbps}. Each video is broken into a sequence of small equal-length video chunks, each chunk containing 2s worth of video content. The emulation starts from 0s and ends at 2000s. The client starts to send chunk requests at 20s and to play video at 35s. Liu's self-adaptive chunk assignment scheme in [7] and the random scheduling method are implemented for comparison.

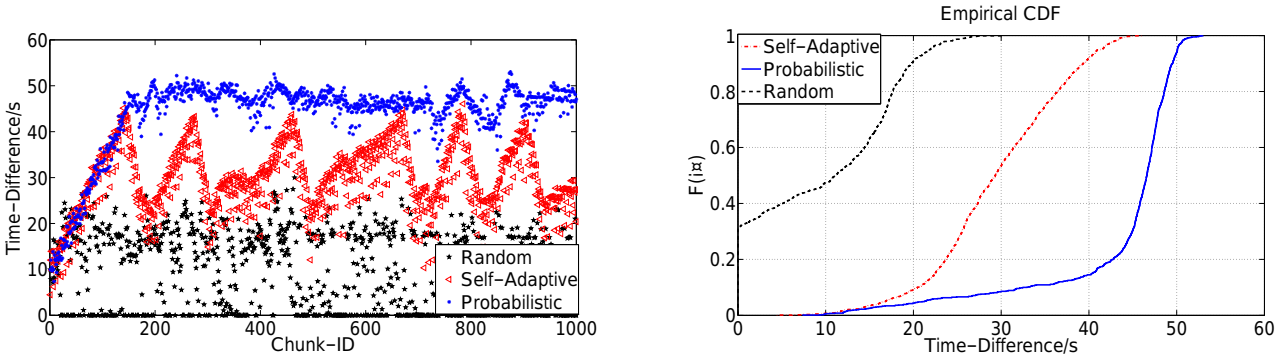
Fig.1 shows the complete download time of each chunk from all three methods, compared with the BenchMark. And the points on the BenchMark are the playback deadline of each chunk. The figures clearly show that our curve is much smoother than the others, and thus our probabilistic scheduling scheme can ensure continuous playback. The random method performs poorly and its curve shakes violently. In Liu's self-adaptive strategy, the chunks are orderly requested. However it takes different time to download them. Thus, Liu's curve is always oscillating, which may deteriorate the continuous video playback. In our method, the chunk download requests are assigned according to their complete download time. So our curve is always smooth, which greatly improve the quality of video playback.

For each chunk n, we define time difference as the difference value between its complete download time $t_{e,n}$ and its playback deadline $t_{d,n}$, i.e., $t_{d,n} - t_{e,n}$. A negative time difference results in playback interruption since the client must play video chunks orderly. On the other hand, if the time differences is always small and fluctuate severely, the playback interruptions may incur when network bandwidths deteriorate, which is not uncommon in the real network. Fig.2(a) shows the time difference of each chunk. It is clearly that our scatterplot is much smoother than others and always the highest, while Liu's fluctuates drastically. Fig.2(b) depicts the CDF of chunks' time differences from different schemes. The random method performs really poor and its CDF simply starts from roughly 0.3, which indicates that many chunks are not



(a) Chunk ID $\in (0, 50)$ (b) Chunk ID $\in (450, 550)$

Fig. 1: Complete download time with given chunk ID



(a) Time Difference with given chunk ID (b) CDF of the Time Differences

Fig. 2: Time Difference between completely download time and playback deadline

received in time. Liu's time differences are mostly distributed from 20s to 40s. At the same time, our time differences are mainly distributed from 40s to 50s. It is clearly that our scheme greatly improves the quality of video playback.

Table II depicts the proportion of chunks that are received orderly from three methods. The proportions of Random, Self-Adaptive and Probabilistic are 51.4%, 64.7% and 79.7% respectively. Our method greatly improve the proportion of chunks that are received orderly.

TABLE II: The proportion of chunks that are orderly received

scheme	Random	Self-Adaptive	Probabilistic
proportion(%)	51.4	64.7	79.7

V. Conclusion

In this paper, we proposed a novel probabilistic chunk scheduling approach for parallel multi-server DASH. Video chunks are scheduled to the servers which consume the smallest download time while have the highest probability to complete downloading before the deadline. The scheduling approach is formulated as a constrained optimization problem with the objective to minimize the total downloading time. A near-optimal solution algorithm is designed which schedules chunks to the servers with minimal downloading time while the completion probability is under the constraint. Compared with the existing schemes, the experimental results demonstrated that our proposed scheme greatly increased the number of chunks that are received orderly.

References

[1] T.Stockhammer. Dynamic Adaptive Streaming over HTTP:standards and design principles. In Proc. of ACM MMSys11,2011.
[2] S. Akhshabi etc. An Experimental Evaluation of RateAdaptation Algorithms in Adaptive Streaming over HTTP. In Proc. of ACM MMSys11, 2011.
[3] M. Watson. Http Adaptive Streaming in Practice. Netflix, Tech. Rep., 2011.
[4] A. Zambelli. IIS smooth streaming technical overview. Microsoft Corporation, 2009.
[5] James F.Kurose and Keith W.Ross. Computer Networking: A Top-Down Approach, 6th Edition. Addison-Wesley, 2012.
[6] V.K.Adhikari, Y.Guo, F.Hao, M.Varvello, V.Hilt, M.Steiner, and Z.-L.Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In Proc. of IEEE INFOCOM, 2012.
[7] Guibin Tian and Yong Liu. Towards Agile and Smooth Video Adaption in Dynamic HTTP Streaming. In Proc. of ACM CoNEXT, 2012.
[8] J.Kilpi and I.Norros. Testing the Gaussian approximation of aggregate traffic. In Proc. of ACM SIGCOMM Workshop on Internet measurment, 2002.
[9] M.Jain and C.Dovrolis. End-to-end estimation of the available bandwidth variation range. In Proc. of ACM SIGMETRICS Performance Evaluation Review, 2005.