Token-Tray/ Weighted Queuing-Time (TT/WQT): An Adaptive Batching Policy for Near Video-on-Demand System*

Wushao Wen[1+], S.-H. Gary Chan[2], Biswanath Mukherjee[1]
[1]Networks Research Lab, University of California, Davis, CA 95616
[2]Dept. of Computer Science, Hong Kong Univ. of Sci. and Tech., Kowloon, Hong Kong
wswen@ece.ucdavis.edu, gchan@cs.ust.hk, mukherje@cs.ucdavis.edu

Abstract—**In near video-on-demand (near-VoD), requests for a video title are grouped together (i.e. batched) and are served with a single multicast stream, thereby increasing the number of concurrent users which can be supported by the system. Since users may not be able to tolerate the delay incurred by batching and hence cancel their requests, a batching policy should be designed so as to achieve low user loss and high revenue (given by the total pay-per-view collected over a long period of time across all movies). We propose an adaptive batching policy which offers users low delay at low arrival rate, and gates the allocation of the channels at high rate. Such adaptivity is achieved by the use of a simple "token-tray" (TT) scheme which governs when a stream may be allocated to a movie. In assigning a movie to a stream, we propose a weight function which depends on the user queuing-time and its pay-per-view(hence the term weighted queuing-time" (WQT). By comparing our batching policy (TT/WQT) with a number of traditional ones (FCFS, Forced-Wait, Batch-Size-Based scheme, rtc.), our scheme is shown to achieve the highest revenue and lowest loss rate even when the arrival rate changes, with the user loss rate across the movies being fairly uniform, and the user delay being fairly low even at high arrival rate.**

Keywords— **Near video-on-demand, request batching, multicasting, service revenue, user reneging, user loss.**

I. Introduction

Advances in networking and server technologies have made the provisioning of on-demand video services to the home a reality [1], [2], [3], [4]. When a video is popular, the use of a single multicast channel (i.e., multicast stream) serving many users simultaneously may offer a more cost-effective solution. In this scheme, requests for a given video arriving over a period of time are grouped (i.e., batched) together and served with a single multicast stream. This is referred to as near video-on-demand (n_VoD).

We show in Fig. 1 a n_VoD system, in which a video server delivers its movies(or video) to the users through a multicast-capable network.

In providing n_VoD service, the limited server channels have to be assigned carefully to the movies via an efficient batching policy. The design of a batching policy generally involves two components: i) *when* to *allocate* a free channel to a movie (channel allocation); and ii) *which* movie should be *assigned* to an allocated channel (movie assignment). The performance of a n_VoD system depends very much on the batching policy used.

This paper is organized as follow. In the following section, we discuss previous work. Then, we describe the allocation and assignment schemes (TT/WQT) in Section III. In Section IV, we present some illustrative numerical results based on our detailed simulation model of TT/WQT, followed by its comparison with some other traditional schemes from the literature. We finally conclude in Section V.

* Gary Chan's work was supported, in part, by the Hong Kong Telecom Institute of Information Technology, and by the Sino Software Research Institute in the HKUST.
+Contact author.

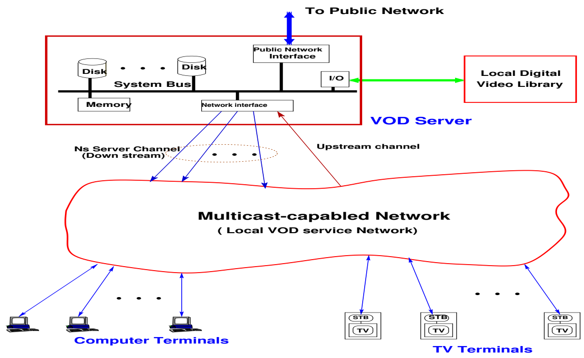

Fig. 1. A near video-on-demand (n_VoD) system.

II. Previous Work

Movie-assignment scheme based on per-movie queuing has been studied in [5], [6]:

- First-Come-First-Served (FCFS): Whenever a server channel is released, the movie corresponding to the earliest request arrival is served (along with all of the current waiting requests for the same movie).
- Maximum-Queue-Length (MQL): Incoming requests join separate queues according to the movies they request. Whenever a channel is released, the longest queue is served first.

Delaying channel allocation has also been considered previously [7], [8], [9], [10], [11], [12], [13], [14]. We briefly describe some of these schemes here:

- Forced-Wait [8], [10], [12] — In forced-wait, the first user in the request queue of a movie is forced to wait for at least a certain amount of time before the movie can be served. When there are more than one movie satisfying this condition at the instant of channel allocation, the movie with the earliest arrival will be served.
- Batch-Size-Based scheme [7], [9], [11] — This scheme keeps on batching requests until at least a certain number M of requests are collected for a movie. If there are more than one movie with at least M users at channel allocation, then the channel will be assigned to the queue with the most requests.
- Fixed Scheduling [9], [13], [14] — In fixed scheduling, a certain fixed number of channels is given to a movie and the movie is scheduled at regular intervals according to this number.

However, all of these schemes are non-adaptive and channel allocation becomes too cautious and conservative when the arrival rate drops (so that users suffer unnecessary delay).

III. TT/WQT: Our Proposed Batching Policy

In this section, we first describe the token-tray channel-allocation scheme, and then the weighted queuing-time movie-assignment scheme. By integrating the two schemes, we have a batching policy termed "token-tray/ weighted queuing-time" (TT/WQT).

A. The Token-Tray (TT) Channel-Allocation Scheme

The token-tray scheme determines when to allocate a channel. The scheme is able to offer low start-up delay to users when the request rate is low, and it gates the allocation of the channels (and hence offers low loss rate) when the request rate is high.

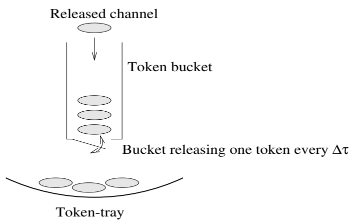

Fig. 2. The token-tray (TT) channel-allocation scheme.

There are N_s tokens in the system, each of which specifically corresponds to a channel. An idle channel corresponds to a free token which may be allocated to a movie. The time at which a free channel may be allocated is governed by a leaky token bucket and a token tray as shown in Fig. 2. At the release of a channel (i.e., after a movie finishes), the channel's corresponding token is dropped into the leaky bucket. Independent of this operation, the token bucket opens once every $\Delta\tau$ minutes, releasing one token, if any, from the bucket onto the token tray at each opening. Clearly, in order to space out the allocation of the channels at regular intervals at high arrival rate, $\Delta\tau$ should be set to $\overline{T}_h/N_s$, where $\overline{T}_h$ minutes is the average holding time of the streams (corresponding to the movie length played) in the system. In reality, $\overline{T}_h$ is estimated by keeping a running average of the channel holding time.[1]

Tokens in the token tray correspond to the channels which may be allocated immediately. Upon the assignment of a movie to a channel, the token corresponding to that channel is taken out of the tray. Let the number of tokens in the token bucket be T_b, the number of tokens in the token tray be T_t, and the number of channels currently in use be T_u. We obviously have $T_b + T_t + T_u = N_s$.

It should be clear that, when the arrival rate is low, there would be plenty of tokens on the tray and hence the system offers low delay to the users. On the other hand, when the arrival rate is high, the release of tokens onto the tray is gated by $\Delta\tau$, offering a low loss rate to the users.

[1] Such a running average is calculated as follows. If we have played $(N-1)$ movies so far, and the current estimate of $\overline{T}_h$ is $\hat{T}_h(N-1)$, then upon the finish of the Nth movies with holding time $\overline{T}_h(N)$, the new estimate of $\overline{T}_h$ (denoted by $\hat{T}_h(N)$) is given by $\hat{T}_h(N) = (\hat{T}_h(N-1) \times (N-1) + T_h(N))/N$, from which we obviously have $\lim_{N\to\infty} \hat{T}_h(N) = \overline{T}_h$. The interval $\Delta\tau$ is updated by the estimate of $\overline{T}_h$ divided by N_s at each opening of the token bucket.

B. The Weighted Queuing-Time (WQT) Movie-Assignment Scheme

The system maintains a separate request queue for each movie, with the queue indexed by $i = 1, 2, ..., N_v$, where N_v is the number of movies. When a channel is to be allocated (indicated by the presence of a token in the token tray), there may be more than one non-empty request queue. The particular movie to be assigned to the channel is given by the movie-assignment scheme. An assignment scheme has a significant effect on the system revenue, the user loss rate, and the delay experienced by the users.

We consider a movie-assignment scheme termed "weighted queuing-time" (WQT), which takes into consideration the users' pay-per-view (PPV) and their waiting time, so as to achieve high profit and low user loss. Associated with each request queue i is a weight W_i, which is a function of the waiting time and PPV of the (un-reneged) requests in the queue. The movie with the highest weight will be assigned to an idle channel. Obviously, W_i should be high if the revenue collected from serving the queue is high and if the users in the queue have been sufficiently delayed.

We consider the following weighting function for the request queue i:

$$W_i = \sum_{j=1}^{q_i} PPV_j \times (WT_j)^{\alpha},$$

where q_i is the length of queue i, PPV_j is the pay-per-view of the jth request in queue i, WT_j is the waiting time for the jth request (in queue i), and α is a parameter indicating how important the user's queuing time is (e.g., a higher α means that the scheme favors a queue with the oldest request). Clearly, as α increases to infinity, this scheme becomes the First-Come-First-Served (FCFS) scheme. When α decreases to zero, this scheme becomes the Maximum-Revenue-First policy (MRF), in which we select the queue with the highest revenue (a greedy algorithm). If the PPV is the same for all requests, then MRF is equivalent to serving first the queue with the maximum number of outstanding requests. For comparison, we have also studied the *Round-Robin* (RR) scheme, in which the channel is assigned to a queue in a round-robin manner.

IV. Illustrative Numerical Examples

We have studied the performance of our new batching policy using simulation.[2]

A. Simulation Model and Performance Measures

We consider the following in our simulation:

- Video popularity: A random arrival requests video i with probability ν_i. We consider the following Zipf distribution in our study which has also been used by other investigators [5], [6]:

$$\nu_i = \frac{i^{\zeta}}{\sum_{j=1}^{N_v} j^{\zeta}},$$

According to some actual rental data, $\zeta \approx 0.7 - 0.8$ [5]. In our study, we take $N_v = 100$ (with $\zeta = 0.746$)/

[2] The analysis of the policy is complicated by the the weight function for the queues and the non-exponential channel holding time and non-exponential user reneging time.

- The movies have different channel holding time T_h, following a uniform distribution with minimum play-time a minutes and maximum play-time b minutes, i.e., $T_h \sim U[a, b]$. Default value are $a = 60, b = 150$.
- The aggregate request arrival process is Poisson with rate λ requests/minute. Therefore, the request rate for the ith video is given by $\nu_i \lambda$ requests/minute.
- Each user has a certain maximum time which he/she is willing to wait. If the requested video is not displayed within this time, the user leaves the system (i.e., reneges) and hence is lost. Different users have different tolerable delay. We consider that the tolerable waiting time for a user is independent identically distributed according to a normal distribution truncated on the left, with a mean of μ minutes and a standard deviation of σ minutes before the truncation [6]. The minimum time which a user is willing to wait (i.e., the truncation point) is u_{min} minutes. Default value are $\mu = 7, \sigma = 10, u_{min} = 2$.
- Each incoming user has a certain PPV (in dollars). As an example, we consider the PPV is independent identically distributed according to a uniform distribution $PPV \sim U[m, n]$.Default value are $m = 2, n = 12$.
- There are N_s channels and N_v videos in the system. Default value are $N_s = 300, N_v = 100$.

We are interested in the following performance measures:

- *System revenue*: System revenue is given by the total PPV collected over a long period of time (infinite horizon), which translates to revenue rate R ($/min.), the amount of dollars collected per minute at the steady state of the system's operation.
- *User loss rate*: User loss rate is defined as the fraction of users who renege from the system. We consider the overall loss rate, p_l, given by the total number of users lost divided by the total number of arrivals over our simulation run, and the loss rate across movies, given by the total users lost for a movie divided by the total number of requests for that movie over our simulation run (which is 350 hours of simulated system time). Obviously, if the loss rate across the movies is uniform, there is no discrimination against requests for any movies. Note that, even though the loss rate across the movies should be as uniform as possible (for fairness), achieving such uniformity may mean sacrificing revenue, because we may then have to serve more frequently some less popular movies with less income. Therefore, in reality, there is a trade-off between the degree of the uniformity (i.e., fairness) and the system revenue.
- *User delay*: User delay is the delay between the time when a user requests a movie until the time when the movie starts playing. We consider the average overall user delay in the system, $\overline{D}$ (minutes), the average delay across the movies, and user delay distribution (if a request reneges, then the waiting time of this user will not be counted).

B. Simulation Results and Comparisons

In this section, we first present the performance of the TT/WQT batching policy, and then we compare this policy with other traditional ones.

B.1 Performance of the TT/WQT Batching Policy and Effect of its Various Parameters

We first examine the effect of α. Recall that, as α increases from zero, the weight put on the user waiting time in a queue increases and the movie-assignment scheme transits from Most-Revenue-First (for $\alpha = 0$) to First-Come-First-Served (for $\alpha = \infty$). By increasing α, we can achieve a trade-off between the loss rate and the revenue rate.

Figure 3 shows the overall loss rate p_l and revenue rate R versus α using TT/WQT for a user request rate $\lambda = 10$ req/min. As α increases, p_l first decreases sharply and then slowly rises (to a limiting loss rate corresponding to TT/FCFS). This is due to the fact that, when α is low, the system discriminates against unpopular movies, starving the corresponding users and causing them to renege. On the other hand, if α is too high, the system serves the request queue with the earliest arrival first, hence serving the unpopular movies quite often. However, since fewer channels (than necessary) are now assigned to the more popular movies, more of their users renege and hence the loss rate increases. Regarding the system revenue R, it first increases quite sharply and then decreases slowly to a limiting value corresponding to TT/FCFS as α increases. It is surprising to note that MRF (i.e., at the point $\alpha = 0$) does not lead to high revenue. This is because the scheme does not take into account the user's waiting time, leading to a high user loss rate, and hence high uncollected revenue, of the less popular movies. This example shows that a greedy approach may not lead to high revenue. As α increases, R decreases due to the increase in p_l. A good choice of α can hence achieve high R and low p_l. We note from Fig. 3 that $\alpha = 2.0$ is a good value to use.

We show in Fig. 4 the loss rate with respect to movie popularity (loss rate across movies), given α. (Note that all the curves end on movie popularity around 0.115, the probability of accessing the most popular movie in our system.) We clearly see that, when α is low (e.g., $\alpha \approx 0$), the policy favors popular movies; when α is high, the loss across the movies is more uniform.

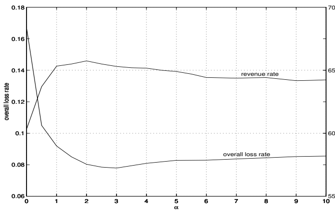

Fig. 3. Effect of α on R and p_l for TT/WQT ($\lambda = 10$ req/min).

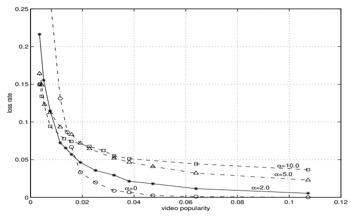

Fig. 4. Loss rate vs. movie popularity for different values of α ($\lambda = 10$ req/min).

We now study different movie-assignment schemes using the token-tray allocation scheme; specifically, we compare the performance of TT/WQT with TT/FCFS (Token-Tray/First-Come-First-Served), TT/MRF (Token-Tray/Most-Revenue-First), and TT/RR (Token-Tray/Round-Robin). In all results follow, $\alpha = 2.0$ is the default value. We show in Fig. 5 R versus λ for these schemes. As λ increases, R also increases. Also shown is the "ideal line," the maximum R achievable when there were no user loss (given by $\lambda E[PPV]$). All schemes perform similarly (and close to the ideal) when λ is low as almost all requests are accepted. Among all the schemes, the revenue of TT/WQT is the highest. The performance of TT/RR is the lowest because TT/RR treats all movies fairly, resulting in popular movies not having enough channels, and leading to a higher overall loss rate and lower revenue rate.

We show in Fig. 6 p_l versus λ, with different movie-assignment schemes. As λ increases for all the schemes, p_l increases (as expected) and levels off to a certain value. Clearly, TT/WQT achieves the lowest p_l by taking into consideration the user waiting time (this is the main reason for its high revenue). Note also the high loss rate for TT/RR for high values of λ, due to its high loss of requests for popular movies.

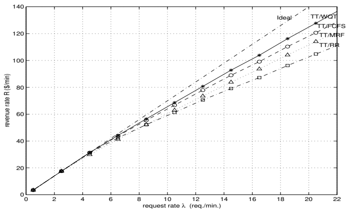

Fig. 5. R vs. λ for different movie-assignment schemes.

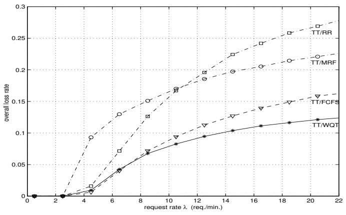

Fig. 6. Overall loss rate p_l vs. λ for different movie-assignment schemes.

We next examine the loss rate with respect to movie popularity in Fig. 7 with these assignment schemes, given $\lambda = 10.5$ req/min. As expected, TT/MRF shows the most discrimination against the unpopular movies, leading to high user loss for these movies. TT/RR shows uniformity in loss rate across movies. However, the loss rate is consistently high across all the movies, and from its high p_l and low R, we note that such uniformity is hard to justify. Our scheme, TT/WQT, achieves moderate uniformity in the loss rate.

We next examine the average user delay $\overline{D}$ versus λ in Fig. 8 for these schemes. As λ increases, $\overline{D}$ increases for all these schemes and then settles to a limiting value. The delay for TT/RR is the highest because movies have to wait for their turns and hence the delay for the popular movies is high. On the other hand, the overall average delay for TT/MRF is the lowest as it favors popular movies at the cost of unpopular movies. The delay for TT/WQT is relatively low among all the schemes.

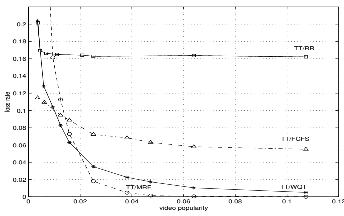

Fig. 7. Loss rate vs. movie popularity for different movie-assignment schemes.

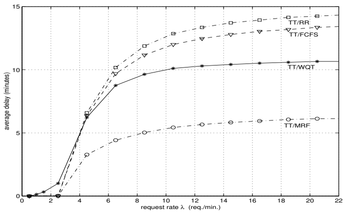

Fig. 8. $\overline{D}$ vs. λ for different movie-assignment schemes.

We show in Fig. 9 the average user delay across movies for different movie-assignment schemes ($\lambda = 10.5$ req/min). All the schemes except TT/RR discriminate against unpopular movies. TT/MRF achieves the lowest average delay across movies at the expense of the high loss rate for the unpopular movies (for unpopular movies, the users who get served eventually are those few who are willing to wait for a long time).

In summary, TT/WQT achieves the highest revenue and lowest user loss rate among the various schemes (including the traditional assignment schemes) by factoring in the user's PPV and its waiting time in movie assignment, while offering a reasonably good uniformity in user loss and user delay.

B.2 Comparison with Conventional Batching Policies

In this section, we compare the performance of TT/WQT with some of the traditional batching policies (First-Come-First-Served (FCFS), Maximum-Queue-Length (MQL), and Forced-Wait (FW) with the minimum waiting time of the first user in the request queue equal to 5 minutes, as well as Batch-Size-Based (BSB) scheme with the minimum batch size $M = 3$). We show in Fig. 10 R versus λ for these schemes. We again show the "ideal" (maximum) revenue corresponding to the case when there were no user loss. We see that TT/WQT outperforms all of the other traditional schemes. When λ is low, the BSB scheme does not perform well because users are likely to renege before the BSB's target batch-size ($M = 3$ in this case) is reached. As expected, when λ is high, the BSB's performance is similar to that of MQL. As observed before for MRF, the revenue collected using MQL is not as high as expected due to its failure in considering the user's waiting time, and hence many users, especially those requesting unpopular movies are lost, leading to uncollected revenue. The TT/WQT scheme, on the other hand, takes the waiting time and PPV into consideration and hence achieves the highest revenue among all of the other schemes.

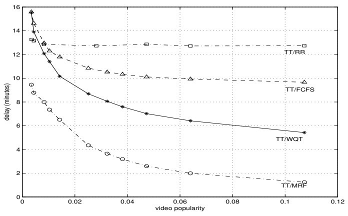

Fig. 9. Average user delay across movies for different movie-assignment schemes ($\lambda = 10.5$ req/min).

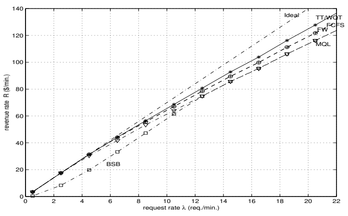

Fig. 10. R vs. λ for different batching policies.

We show in Fig. 11 p_l versus λ for different schemes. Clearly, TT/WQT achieves the lowest p_l. Regarding the BSB scheme, the loss rate first decreases and then increases. This is because of the following. When λ is low and hence the interarrival time of requests is long, many users would have reneged before the target batch-size is reached, thereby causing a high loss rate. As λ increases, it takes a shorter duration to collect a batch and hence fewer users renege. However, as λ increases further, channels begin to run out and BSB's performance resembles that of MQL. Note that, when the arrival rate is low, FW still has a loss rate. This is because, even though there are many free channels, the requests are still forced to be delayed and hence may leave the system. As λ further increases, the loss rate decreases slightly before it rises again because the relative number of users leaving the system as compared to the total number of arrivals decreases, because batching is possible and some users (especially those requesting popular movies) enjoy lower delay than the forced waiting time. We show in Fig. 12 the loss rate with respect to movie popularity for $\lambda = 10.5$ req/min. While the BSB and

MQL schemes discriminate against unpopular movies, FCFS and FW achieve somewhat uniform loss rate (though their high loss rates for the popular movies lead to high overall loss rates). Our policy, TT/WQT, achieves an uniformity somewhere in between.

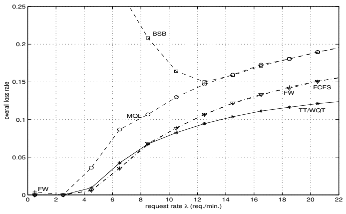

Fig. 11. Overall loss rate p_l vs. λ for different batching policies.

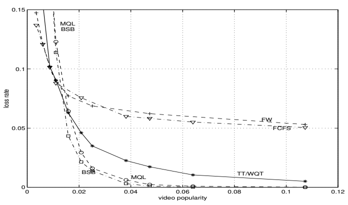

Fig. 12. Loss rate vs. movie popularity for different batching policies ($\lambda = 10.5$ req/min).

We show in Fig. 13 $\overline{D}$ versus λ for these schemes. The overall average delays for the MQL and BSB schemes are lower because users requesting popular movies enjoy very low delay. When the request rate is low, the average delay of the BSB scheme is high because users have to wait for a long time before the minimum batch size can be reached. When the arrival rate is low (e.g., $\lambda \leq 3$ req/min), TT/MQT has similar but slightly higher delay than FCFS due to its gated release of tokens (and hence channel allocation). At low values of λ, the users suffer delay in FW (equal to the forced waiting time of the first user in a request queue). However, when λ increases, the average delay actually decreases slightly (before it rises again). This is because some users (especially the ones requesting popular movies) are able to enjoy lower delay due to batching. When λ is high, the delays for both FW and FCFS are the highest because they allocate many channels to the unpopular movies, causing high delay for the users requesting popular movies. Our policy, TT/WQT, achieves intermediate average delay among all the policies. We show in Fig. 14 average user delay across movies for different batching policies ($\lambda = 10.5$ req/min). FW and FCFS achieve a more uniform but higher average delay across movies. On the other hand, the MQL and BSB schemes favor popular movies in order to offer a low overall average delay, at the cost of high request loss rate for unpopular movies.

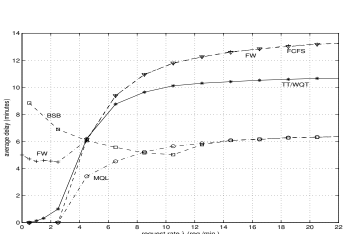

Fig. 13. $\overline{D}$ vs. λ for different batching policies.

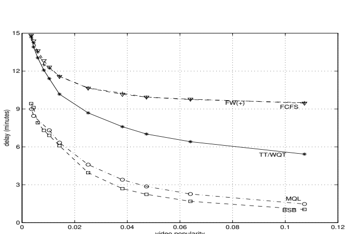

Fig. 14. Average user delay vs. movie popularity for different batching policies ($\lambda = 10.5$ req/min).

In summary, for a system in which we want high revenue and low loss rate for a wide range of arrival rate (or fluctuating arrival rate), TT/WQT appears to be the best choice among all the traditional schemes examined. It also achieves a good performance in the uniformity of loss rates across movies. When the arrival rate is low, it offers low delay to the users, and when the arrival rate is high, it offers more deterministic delay (as the channel allocation is gated).

V. Conclusion

In this investigation, we have proposed an adaptive channel-allocation scheme and move assignment scheme TT/WQT. We have compared the performance of our TT/WQT policy with some other traditional batching policies (First-Come-First-Served, Maximum-Queue-Length, Forced-Wait, and Batch-Size-Based schemes). Our scheme achieves the highest revenue and lowest overall loss rate, and the loss rate across movies is fairly uniform and the user delay is reasonably low even at high arrival rate. Our results suggest that, in serving requests in a n_VoD system, the user pay-per-view and their waiting time should be taken into consideration in order to achieve high revenue and low loss.

References

[1] T. Little and D. Venkatesh, "Propespects for interactive video-on-demand," *IEEE Multimedia Magazine*, pp. 14-24, Fall 1994.

[2] V. O. K. Li and W. Liao, "Distributed multimedia systems," *Proceedings of the IEEE*, vol. 85, pp. 1063-1108, July 1997.

[3] F. A. Tobagi, "Distance learning with digital video," *IEEE Multimedia Magazine*, pp. 90-94, Spring 1995.

[4] A.D. Gelman, et al., "A store-and-forward architecture for video-on-demand service," *Canadian Journal of Electrical and computer Engineering*, vol. 18, pp. 37-40, January 1993.

[5] A. Dan, et al., "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, pp. 112-121, June 1996.

[6] C. C, Aggarwal, et al., "On optional batching policies for video-on-demand storage servers," in *Proceedings of the 3rd ICMCS*, (Hiroshima, Japan), pp. 312-316, 1996.

[7] A. D. Gelman and S. Halfin, "Analysis of resource sharing in information providing services," in *Proceedings of IEEE Globecom' 90*, pp. 312-316, June 1990.

[8] K. C. Almeroth, et. al, "Long term resource allocation in video delivery systems," in *Proceedings of IEEE Infocom' 97*, (Kobe, Japan), pp. 1333-1430, April 1997.

[9] E. L. Abram-Profeta and K. Shin, "Scheduling video programs in near video-on-demand systems," in *Proceedings of ACM International Multimedia Conference*, pp. 359-371, Nov. 1997.

[10] S.-H. G. Chan, et al., "Providing on demand video services using request batching," in *Proceedings of the IEEE ICC' 98*, (Altanta, Georgia), pp. 1716-1722, June 1998.

[11] S.-H. G. Chan and F. A Tobagi, "On achieving profit in providing near video-on-demand services," in *Proceedings of the 1999 IEEE ICC' 99*, (Vancouver, Canada), pp. 988-994, June 1999.

[12] S.-H. G. Chan, F. A Tobagi, and T.-M. Ko, "Bandwidth planning in near video-on-demand," in *Proceedings of the 1998 IEEE ISCAS' 98*, (Montery, CA), pp. 61-64, June 1998.

[13] E. Gelenbe and H. Shachnai, "On g-networks and resource allocation in multimedia systems," in *IEEE RIDE (Research in Data Engineering)*, (Orlando, Florida), pp. 104-110, Feb. 1998.

[14] A. K. Tsiolis and M. K. Vernon, "Group-guaranteed channel capacity in multimedia storage servers," *Performance Evaluation Review*, vol. 25, no. 1, pp. 285-297, 1997.

[15] L. Gouchik, et al., "Apdative piggy-backing: a novel technique for data sharing in video-on-demand storage servers," *ACM Multimedia Systems*, vol. 4, pp. 140-155, June 1996.

[16] S.-W. Lau, et al, "Merging video streams in a multimedia torage servers: Complexity and Heurestics ," *ACM Multimedia Systems*, vol. 6, pp. 29-42, Jan. 1998.

[17] C. Aggarval, et al., "On optimal paggybacking merging policies for video-on-demand systems," *Performance Evaluation Review*, vol. 24, pp. 200-209, May 1996.

[18] S. Sheu, et al., "Chaining: A generalized batching technique for video-on-demand systems," in *Proceedings of the IEEE ICMCS' 97*, (Ottawa, Canada), pp.110-117, June 1997.

[19] P. S. Yu, et al., "Design and Analysis of a look-ahead scheduling scheme to support pause-resume for video-on-demand applications," *Multimedia Systems*, vol. 3, no. 4, pp. 137-149, 1995.

[20] W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia*, pp. 51-62, October-December 1997.