Analytic Evaluation of Quality of Service for On-Demand Data Delivery

Hongfei Guo (guo@cs.wisc.edu)
Haonan Tan (haonan@cs.wisc.edu)

Abstract

Quality of service (QoS) measured as balking probability and average waiting time is of great interest for on-demand data delivery service providers. In this study, we develop and validate two analytical models, a balking model and a waiting model, for evaluating balking probability and average waiting time respectively for media servers with limited bandwidth. Based on in-depth analysis of simulation results, we also propose an interpolation of the average service time in the waiting model when measuring the Patching protocol. Compared to simulation, the balking model captures the trend of balking probability with changing server bandwidth; the waiting model yields reasonably accurate results when measuring the Hierarchical Multicast Stream Merging protocol; the interpolation improves the accuracy of the waiting model when measuring the Patching protocol. Finally, using those two models, we evaluate and compare two multicast protocols – Patching and HMSM – and suggest server bandwidth C* as a reasonable trade-off between QoS and server utilization.

1. Introduction

Multicast network protocols are widely used to save the server bandwidth for on-demand multimedia data delivery. As the market grows, the quality of service of multimedia servers gains more and more attention from service providers. In this study, we develop and validate two analytical models, balking model and waiting model, for evaluating balking probability and average waiting time respectively for media servers with limited bandwidth, which can be effectively used to help service providers choose the appropriate server bandwidth to optimize the cost-effectiveness of the stream media service.

Since a server has only a finite capacity, the incoming media requests are no longer contention-free. To address the cost-effectiveness issue, it is necessary to have measures that reflect the quality of the stream media service. Different measures are used in different models. For the balking model where requests withdraw whenever immediate service is not available, the customer balking probability is the key measure; for the waiting model where clients are patient enough to wait for service, the mean waiting time is our measure of interest.

Using those two analytic models we compare and evaluate two multicast protocols: grace patching and hierarchical merging. Since the core CMVA equations derived here are essentially protocol independent, it is straightforward to extend the modeling methodology to include other protocols such as dynamic skyscraper and send-latest receive earliest [HuSh97].

Several multicast protocols have been modeled analytically, which led to some quantitative results on the average bandwidth requirement [EaVZ00b]. Yet to our knowledge, so far there are no analytic models that measure the QoS of multimedia servers with limited bandwidth.

The main contribution of this paper is mainly threefold. Firstly, we develop two analytic models which ease the decision making during server design phase by providing QoS metrics (balking probability and waiting time) that both users and service providers concern about. Secondly, based on in-depth examination of estimated results against simulation results, we propose an interpolation of the average service time in the waiting model when measuring the Patching protocol, which further improves model accuracy. Finally, both qualitative and quantitative results are presented, based on which a reasonable bandwidth C* as a trade-off between QoS and server utilization is suggested.

The rest of this paper is organized as follows. Section 2 provides the background information regarding the two protocols. Section 3 elaborates on how CMVA equations are established for those two different models. We present validation results and analysis in Section 4. Then we compare and evaluate two multicast protocols in section 5. Section 6 concludes our study and gives a list of some possible future work.

2. Two Protocols For On-Demand Data Delivery

To be self-contained, we include a brief overview of the two multicast protocols used throughout our discussion.

2.1 Patching

The scheme is well described in [EaVZ00b]. In response to a given client request, the server delivers the requested file in a single multicast stream. A client that submits a new request for the same file sufficiently soon after this stream has started begins listening to the multicast, buffering the data received. Each such client is also provided a new unicast stream (i.e., the "patch" stream) that delivers the data that the new request has missed. Both the multicast stream and the patch stream deliver data at the file play rate. This requires that the client have a receive bandwidth twice the file play rate. The patch stream enables immediate playback on the client's side. Once the playback reaches the buffered point, the patch stream can terminate and both the old and the new requests can share the multicast stream.

A common configuration of the patching scheme is that each stream transmits data at normal file playback rate and the client has a receiving bandwidth twice the playback rate so that the client can listen to two server streams at the same time. Given this setting and infinite server capacity, the optimized average required bandwidth for this protocol is proved [EaVZ00b] to be:

$$R_{\text{optimized patching}} = \sqrt{2N_i + 1} - 1, \qquad (2.1.1)$$

where Ni is the expected number of incoming requests for file i during the playback of that file. In fact this is a function of server throughput on file i and the file playback duration T and we denote it as f(Xi, T) for generality. The algorithm itself is fairly simple. However, the choice of the threshold is important to attain the optimal bandwidth because it is also a function of Ni. If the request rate varies greatly from time to time, the threshold has to be adjusted accordingly to stay optimal. On the other hand, with the threshold constraint, this protocol does not take full advantage of an existing multicast stream. It is less aggressive in stream merging compared to hierarchical merging, which is described next.

2.2 Hierarchical Multicast Stream Merging (HMSM)

There are two features that distinguish HMSM [EaVZ00a, EaVZ00b] from patching. One is that every stream is a multicast stream so that any two streams for the same file can merge; the other is that the server uses dynamic programming to generate a merge plan in real-time. The merge plan gets updated every time a new stream is initialized for a new request to reflect the optimal scheme for all the requests that are currently served by the system. This algorithm has a much larger search space for optimality compared with patching. The configuration for HMSM is more flexible. But for the sake of protocol comparison, the same setup is used as in previous section and this is denoted by HMSM(2, 1). The bandwidth requirement for HMSM(2, 1) can be reasonably well approximated [EaVZ00b] by

$$R_{\text{HMSM(2, 1)}} = 1.62 \ln(N_i / 1.62 + 1), \qquad (2.2.1)$$

which is significantly lower when Ni is large. No wonder HMSM is a much more sophisticated algorithm. It employs greater server computing power to lower the bandwidth.

3. Customized Mean Value Analysis (CMVA)

This section covers the analysis of two different types of client request behavior. In the balking model a customer does not wait. Once a request cannot be satisfied immediately, the customer request will simply be dropped. In contrast, in the waiting model each customer request has to wait until it is served. Two waiting requests can coalesce if they ask for the same media file, which means they will share a multicast stream when the service becomes available. Therefore, coalescing also helps ease the server load and it happens regardless of the delivery protocol used by the server. This will be discussed in more detail shortly.

3.1 Customer Balking Model

In the balking scenario, there are a fixed number of streams in the server system. At any given moment, a stream can and only can be in one of two possible states: waiting state (i.e. waiting for customers) and active state (i.e. serving a customer). At the moment a customer arrives, if there is no waiting stream, it simply leaves. Otherwise, a waiting stream will switch to active state, beginning to serve the request. Once the stream terminates, it returns into waiting state.

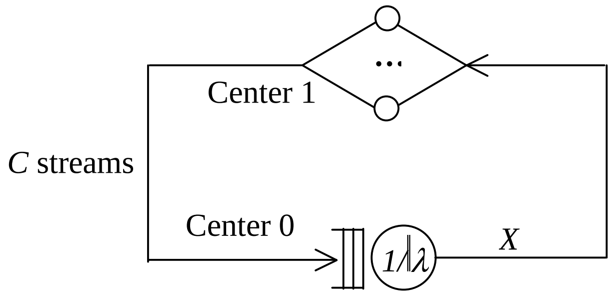


Figure 1. The balking model

In order to model the balking probability, we restate the scenario, yet from the streams' perspective. Initially, waiting streams wait for customers in a queue. An incoming customer activates a stream in the queue, reducing the queue length by one. Once the stream terminates, it returns to the waiting queue again.

We model this scenario by a closed queuing network with a fixed number of streams circuiting in it. The waiting state of a stream is represented by a fictitious SSFR center, which we call center 0. And the active state of a stream is represented by a fictitious delay center, which we call center 1.

The SSFR center models the mean time for a stream waiting for customers. Similarly, the delay center models the mean transmit duration for each file in the server. Eager et al. gave out simple formulas for the average server bandwidth used to deliver stored media files on-demand, as a function of client arrival rate and file length, according to different protocols. Solving those equations iteratively, together with the CMVA equations, the utilization of the SSFR server can be calculated. We assume there are different categories of files on the server, and that the access pattern of each such category of files follows a zipfian distribution. In this simple model, we assume there is only one category. We shall show how to extend this model to include multiple categories shortly.

We use the following notation:

Input Parameters

• C	server capacity
• λ	external customer arrival rate
• M	number of file categories

	For i = 1, 2, …, M
K_i	the total number of distinct files in category i
T_i	mean duration of the entire file in category i
q_i	zipfian parameter in category i
P_i	probability of accessing category i files

Output Parameters

S_1	average service time at center 1
R_0	mean residence time at center 0
X	system throughput.
	For i = 1, 2, … #files on the server
p_i	fraction of customer requests for file i
C_i’	average b/w for file i
S_{1i}	mean service time of file i streams at center 1
S_0	mean service time at center 0
Q_0	mean queue length at center 0
X_i	throughput of streams serving file i
P_B	mean incoming costumer balking probability

Required minimum bandwidth equation: Given a protocol, the average server bandwidth used to deliver stored media files on-demand can be expressed as a function of client arrival rate and file length [EaVZ00b]. Therefore,

$$C' = f(p_i X, T_i), \qquad (3.1.1)$$

where f should be substituted by (2.1.1) and (2.2.1) for patching and HMSM respectively.

CMVA equation: Using the mean value technique for queuing network models, the mean residence time at center 0

$$R_0 = S_0 \cdot \left(1 + \frac{(C-1)}{C} Q_0\right) \qquad (3.1.2)$$

After a stream becomes the head of the queue in center 0 (i.e. at the moment an arriving costumer activates the former head), the average time spent (i.e. service time) before it can leave the center (i.e. at the moment the next costumer arrives) equals the average customer inter-arrival time. Therefore

$$S_0 = \frac{1}{\lambda} \qquad (3.1.3)$$

Applying Little’s result, the average queue length at center 0

$$Q_0(C) = X \cdot R_0 \qquad (3.1.4)$$

The system throughput

$$X = \frac{C}{R_0 + S_1} \qquad (3.1.5)$$

Where the expected stream duration

$$S_1 = \sum_{i=1}^{K} p_i S_i \qquad (3.1.6)$$

After an activated stream leaves center 0, it serves file i with the probability $p_{i.}$ Therefore, the throughput of streams serving file i

$$X_i = p_i \cdot X \qquad (3.1.7)$$

Applying Little's result again, the mean service time of file i streams at center 1

$$S_i = \frac{C_i'}{X_i} \qquad (3.1.8)$$

The utilization of center 0

$$U = X \cdot S_0 \qquad (3.1.9)$$

Balking probability equation: When center 0 is being used, we can interpret it like there is at least one stream waiting in center 0. Therefore an incoming customer will be served. Otherwise, when center 0 is not being used, i.e., when queue length equals to 0, an incoming customer will not find any server, thus will leave without being served. Therefore the costumer balking probability

$$P_B = 1 - U \qquad (3.1.10)$$

3.2 Customer Waiting Model

As illustrated in Figure 2, the waiting model is an open system. The requests floating in are represented by a Poisson process. The stream media server is modeled by a multi-channel service center. In this model, requests queue up when the server does not have enough bandwidth to serve their needs. The input parameters for this model are the same as those for the balking model. The output parameters can be summarized as follows:

• W	mean waiting time for a request (not coalesced)
• U	system utilization
• S	overall mean stream duration estimate For i = 1, 2, …, #files on the server
• p_i	fraction of customer requests for file i
• S_i	mean stream duration for file *i*
• Q_i	mean number of waiting requests (not coalesced) for file *i*
• X_i	mean throughput of requests (not coalesced) for file *i*
• R_i	mean residence time of a request (not coalesced) for file *i*
• C_i'	average number of active streams for file *i*

- R_i' mean residence time adjusted for coalescing
- W_i' mean waiting time adjusted for coalescing

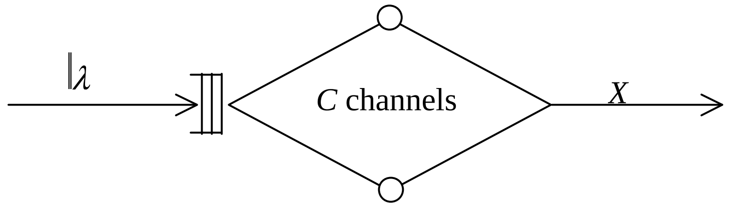


Figure 2. The waiting model

A distinct feature of this waiting model is that requests can be coalesced if an incoming request sees another request for the same file waiting in the queue. Consequently, the number of waiting requests for a single file cannot exceed one at any time. In the steady state, the mean waiting time for each request should be the same regardless of which file it is asking for. In other words, all requests are equal when they are waiting. For a request that cannot be coalesced with any previous requests, its waiting time can be approximated by

$$Wi \approx \frac{S}{C} \cdot \left(\sum_{\substack{j=1 \\ j \neq i}}^{K} Q_{ji} + U^{C}\right), \qquad (3.2.1)$$

where *S / C* estimates the server inter-departure time, which characterizes how fast the queue is moving forward. *S* is an average over *Si*’s weighted by the corresponding *Xi*.

$$S = \frac{1}{\sum_{i=1}^{K} X_i} \sum_{i=1}^{K} S_i X_i \, , \qquad (3.2.2)$$

Where S_i’s can be obtained in the same way as we did for the balking model. *U* is the server utilization and can be derived from C_i’s as

$$U = \frac{\sum_{i=1}^{K} C_i^{'}}{C} \, . \qquad (3.2.3)$$

Given the server utilization U, the probability for a randomly chosen channel being active is also U. By assuming independence among channels, UC is the probability that the server is full and cannot provide service immediately. However, generally this is not the case for a multi-channel server and UC is just a heuristic to reduce the complexity of the model. Q_i is the mean number of waiting requests that cannot coalesce for file *i* in the queue. Applying Little’s Law to the queue, we have

$$Q_i = X_i W \qquad (3.2.4)$$

At any time, the number of non-coalescing waiting requests for a certain file is either zero or one. Therefore, the expected partial queue length for file i

$$Q_i = \Pr[\text{queue length for file } i \text{ is one}] \, . \qquad (3.2.5)$$

By forced flow law plus the fact that any new requests for file *i* will be coalesced if there is already a request for file I waiting. Hence the throughput should satisfy

$$X_i = (1 - Q_i)\, p_i \lambda\,. \qquad (3.2.6)$$

We can replace Xi in (3.2.4) with (3.2.6) and rearrange terms to get a more intuitive form

$$Q_i = \frac{W\lambda_i}{1 + W\lambda_i}, \qquad (3.2.7)$$

where $W\lambda i$ is the expected number of requests that coalesces with a non-coalescing request. Thus (3.2.7) can be interpreted as the fraction of coalescing requests for file i. With this set of equations together with (3.1.1) and (3.1.8), we are able to solve the system iteratively. Once the system is solved, it is easy to see that the mean residence time for a request that does not coalesce is

$$R_i = W + S_i\,. \qquad (3.2.8)$$

The remaining task is to find out what the mean waiting time and the mean residence time for all the incoming requests should be. For a request that cannot coalesce with previous requests, it expects to see $W\lambda i$ requests for the same file arrive during its waiting period. All these requests coalesce into a single request. If we assume the waiting time of the first request to be deterministic with value W, the mean waiting time for those coalescing requests would be W/2, thus the overall average waiting time could be computed as

$$W^{'} = \frac{W\lambda_i \cdot \frac{W}{2} + W}{1 + W\lambda_i}\,. \qquad (3.2.9)$$

Unfortunately, the mean waiting time for the first request is a random variable instead of a fixed number. This equation only gives an approximation of the overall waiting time and we expect it to be more accurate when the variance of waiting time distribution is small. Similarly, if we assume the waiting time distribution to be exponential (or rather, a distribution with a coefficient of variation of 1), the "residual waiting time" for each coalescing request is still expected to be W. Therefore,

$$W = W' \qquad (3.2.10)$$

could be used as an alternative to approximate the overall waiting time. In general, the second moment of the waiting time distribution is needed to quantify W' more accurately. For simplicity, in this paper we only discuss the above two approximations.

Once W' is calculated, the mean residence time for these requests can be modified as

$$R_i^{'} = W^{'} + S_i\,. \qquad (3.2.11)$$

Note that the formula is valid only when coalescing truly occurs, i.e., $\frac{1}{\lambda_i} \le W_i$. In case the arrival rate is too low, coalescing is not likely and (3.2.8) would be more appropriate for the mean residence time estimate.

3.3 A Simple Interpolation

We consider using protocol-specific information to fine-tune the analytical model for Patching. In fact, when a request waits for too long (longer than the threshold), the server has to open a new stream that lasts for T, which may make the stream duration deviate greatly from the estimated value. In order to reflect this in the model, a simple interpolation on stream duration is proposed:

$$S_i' = \frac{g(X_i)}{W_i + g(X_i)} \cdot S_i + \frac{W_i}{W_i + g(X_i)} \cdot T \,, \qquad (4.1.1)$$

where g(Xi) is the optimized threshold. Intuitively, the interpolated mean stream duration is simply a weighted average between the duration obtained in the previous section and the full file playback duration. The waiting time and the optimal threshold are used as weights. The full playback duration would be favored more if the waiting time is longer than the threshold, and vice versa.

4. Validation

We validate our models against some existing simulation results [EaVZ99]. The comparison results show that the balking model captures the trend of balking probability with changing server bandwidth; the waiting model yields reasonably accurate results when measuring the Hierarchical Multicast Stream Merging protocol; the interpolation improves the accuracy of the waiting model when measuring the Patching protocol.

4.1 Balking Model

Figure 3 is the result for the balking model. Dotted lines are simulation outcomes and solid lines are CMVA solutions. The server has 20 files with equal lengths. The total request arrival rate is 2000 per file playback duration. File request frequencies conform to the Zipfian(0) distribution. This configuration will be used throughout the validation.

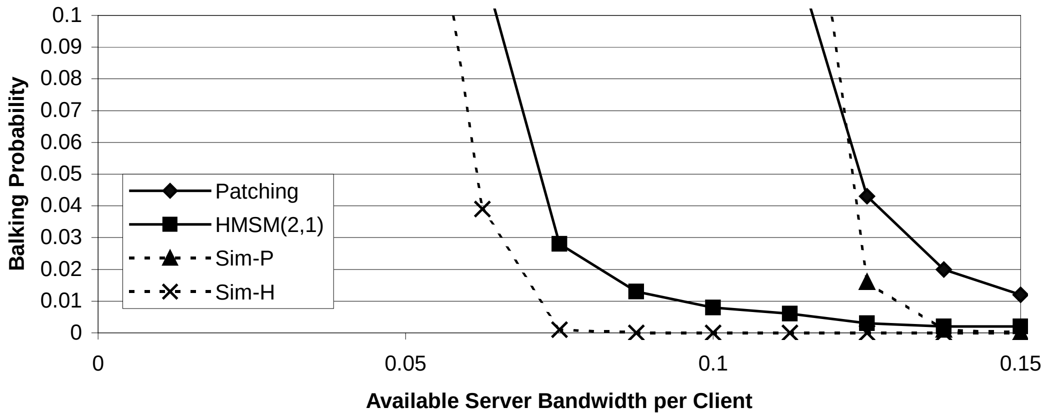
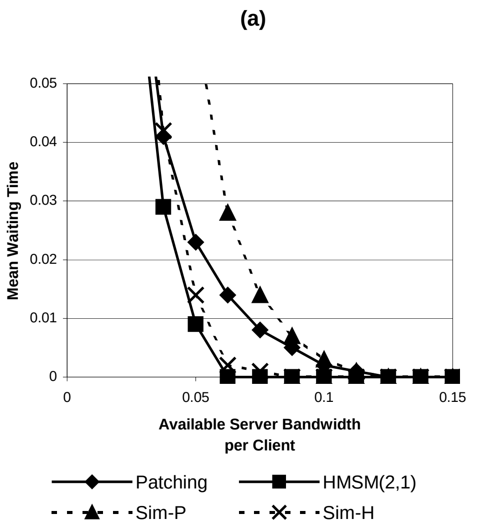


Figure 3. Balking Model Validation

It can be seen that although the analytical model captures the trend of the balking probability as server bandwidth increases, it consistently overestimates the balking probability when the bandwidth gets sufficiently large. In some cases the relative error could be huge. This is somewhat surprising because this model should be exact. We cannot come up with an explanation at the moment this paper is due. For the time being, we can consider the model to be at least qualitatively valid.

4. 2 Waiting Model

The mean waiting time is given in terms of the percentage of the full playback duration of a file. Figure 4 gives a comparison between waiting time approximation (3.2.9) and (3.2.10). It appears that the latter is closer to the simulation results. For HMSM, both approaches perform reasonably well. In contrast, neither way is satisfactory for grace patching when the bandwidth is relatively low. Due to the fact that request coalescing can only reduce the mean waiting time, we suspect that the model has overlooked some important aspect of the patching protocol.



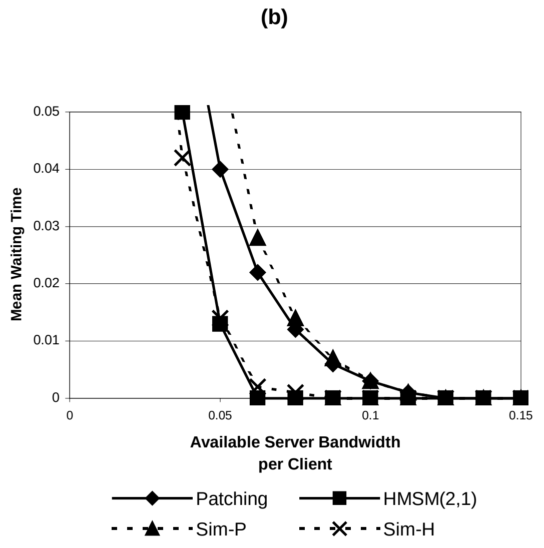


Figure 4. Waiting Model Validation
(a) Deterministic Approximation (b) Exponential Approximation

Further comparison of various server statistics revealed two sources of errors. One is the mean stream duration for each file. Table 1 shows the discrepancy between the analytical model and the simulator when the server bandwidth is 0.05 channel/request. For conciseness, only the overall measure and measures for the three most frequently visited files are listed. The model seems to underestimate this quantity by up to 74% for an individual file and by 56% overall.

Table 1. Mean Stream Duration Comparison at 0.05 channel/request

Duration	Access Probability	Model	Simulator
File 1	.278	.249	.959
File 2	.139	.253	.947
File 3	.093	.258	.869
Overall	1.00	.284	.650

4.3 Interpolation

The effect of the simple service time interpolation is shown in Figure 5 and the improvement is significant. However, there is still room for further improvement. Another source of error comes from the probability that the server is full, which is approximated by UC. Simulation results suggest that this heuristic sometimes can be off by 20% to 30%. Although other more sophisticated mechanisms exist to model this probability [Klei76], we would rather not do so because we cannot further improve our result significantly even if we use the value obtained from the simulator when the server load is high. The average number of active streams is underestimated (as well as the server utilization) and the mean throughput is overestimated. Our speculation is that the formula for bandwidth calculation has to be adjusted to some extent when being adapted to our model. There is no proof for this though.

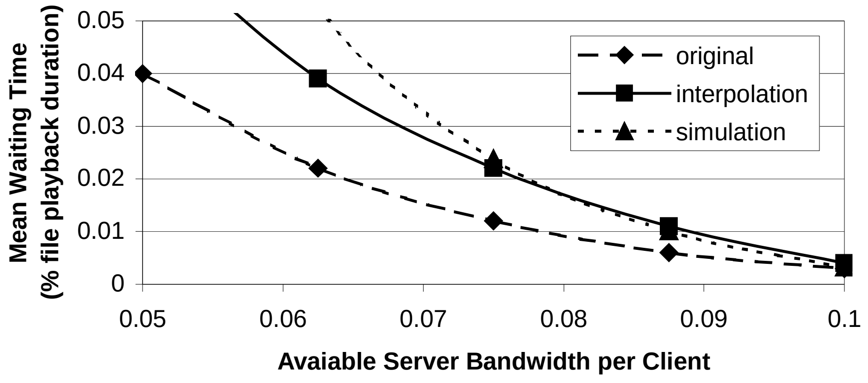


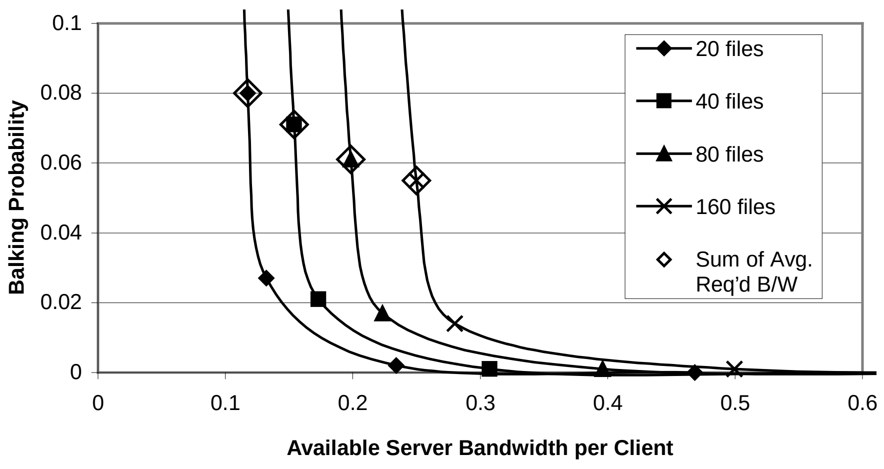
Figure 5. Service Time Interpolation for Optimized Patching

5. Protocol Evaluation

We designed and conducted a series of experiments using those two analytic models to compare and evaluate Patching protocol and HTSM protocol. Not surprisingly, the experiment results show significant impact of different multicast protocols and other system parameters on the quality of service, which can serve as a guideline for choosing appropriate server bandwidth. (Note the data presented below are calculated using the equations derived in section 3.1 and section 3.2.)

Figure 6 illustrates the outcome for the balking model. By comparing two plots vertically, we notice a left shift of curves when HMSM is used instead of patching, which indicates its lower consumption of server bandwidth. Within a single plot, multiple curves show that the requirement on bandwidth grows much more slowly than the increase in number of media files on server. The diamond shaped points are the sum of the average server bandwidths required for immediate delivery of each file [EaVZ99] and we use C* to denote this. For example, for a server with 80 files (other configurations agree with our default setting), C* is 396 for patching and 291 for HMSM(2, 1). These spots are strategic points in that the balking probabilities at these points are already reasonably low. Additional bandwidths slightly beyond (say within 10% above) these points can still be very effective in reducing customer balking. But further increase of the server bandwidth would be much less fruitful since the balking probability curves flatten out really quickly.

(a)



(b)

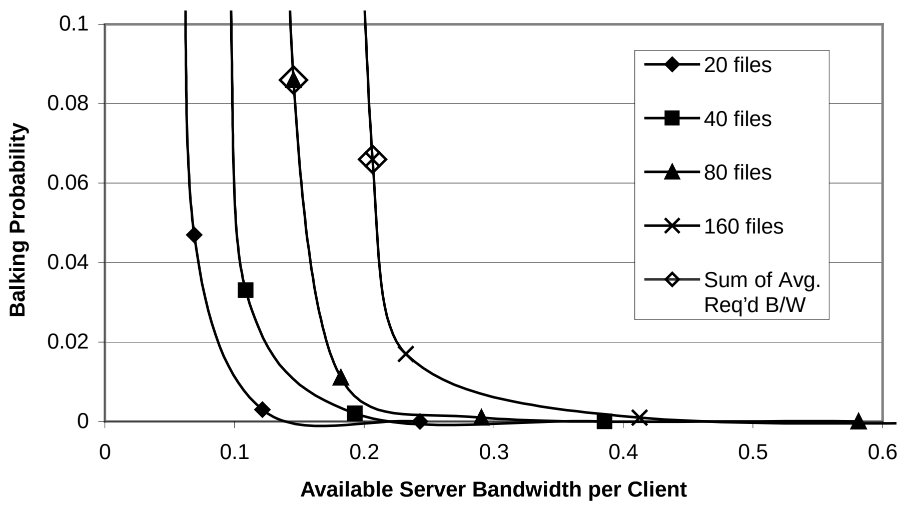
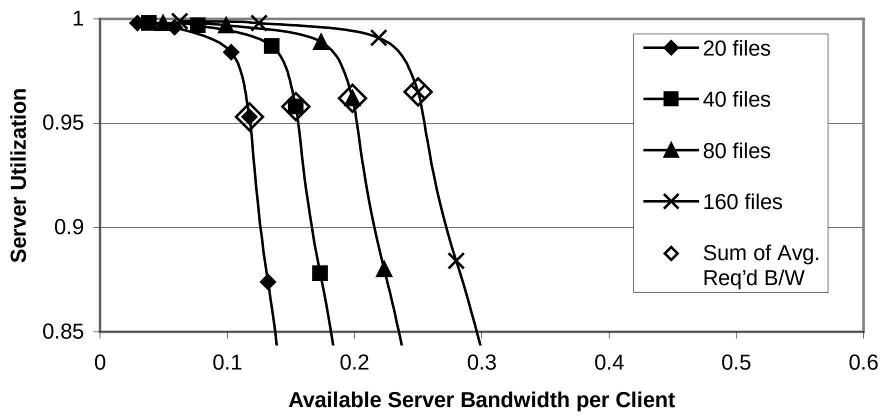


Figure 6. The Impact of Protocols on Bulking Probabilities
(a) Patching (b) HMSM

From a stream media service provider's perspective, it is always desirable to maximize the server utilization. To address this issue, Figure 7 plots the utilization as a function of the server bandwidth, depending on the multicast protocol used and the number of files on the server. Again, we can find the diamond-shaped spots remarkable: the server operates at these bandwidths with high utilization and further investment in bandwidth can degrade utilization drastically.

(a)



(b)

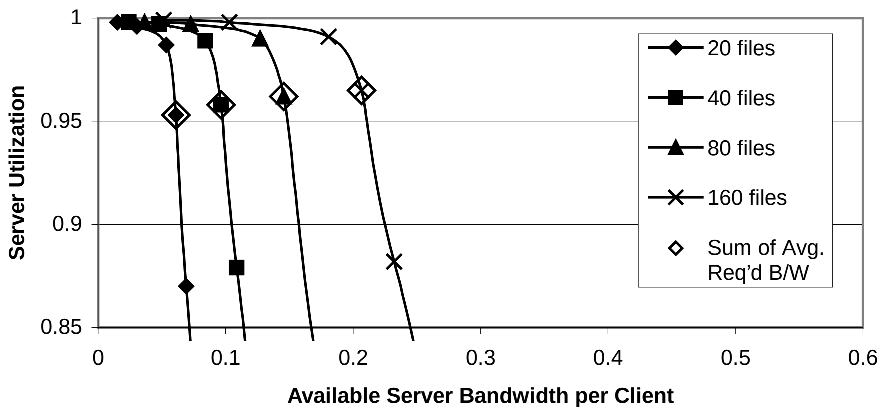
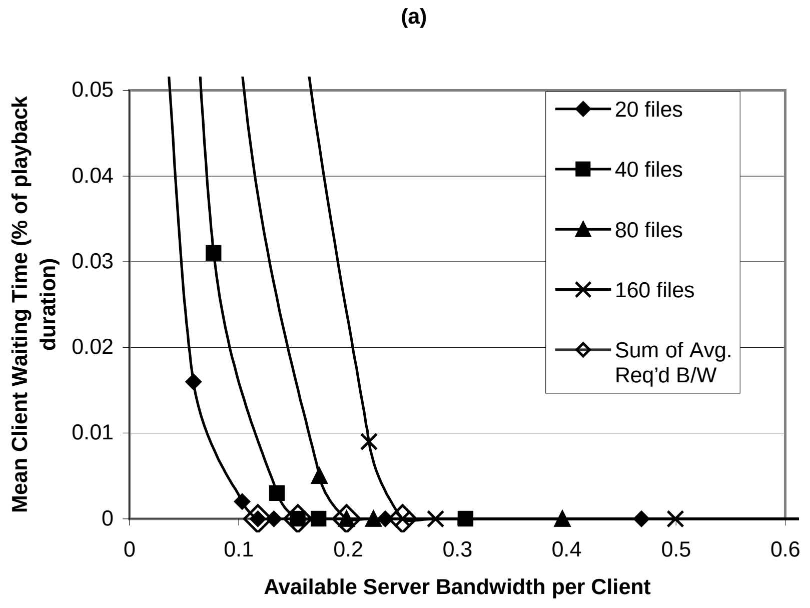


Figure 7. The Impact of Protocols on Server Utilization
(a) Patching (b) HMSM

Similarly, the association between mean request waiting time and server bandwidth is presented in Figure 8. Our findings are consistent with those for the balking model. For the same reason, we omit the utilization analysis for the waiting model.



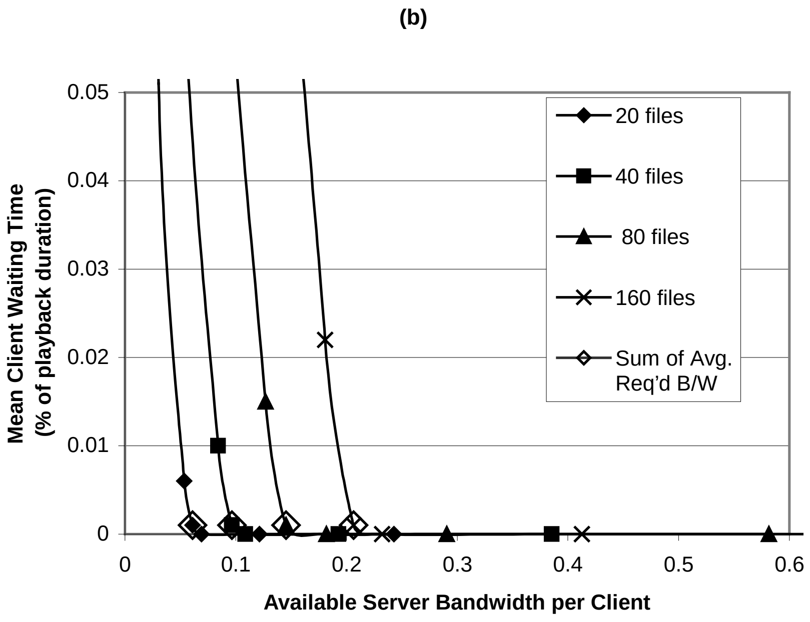


Figure 8. Impact of Protocols on Mean Waiting Time
(a) Patching (b) HMSM

Figure 9 might be more relevant to the cost-effectiveness argument: the higher the load on the server, the more customers to lose, and vice versa. The tradeoff is always there. The trend does not vary much given different numbers of media files. The argument is also applicable to the waiting model (not shown). This is essentially a guideline for system designers to choose the appropriate server bandwidth according to their design policies.

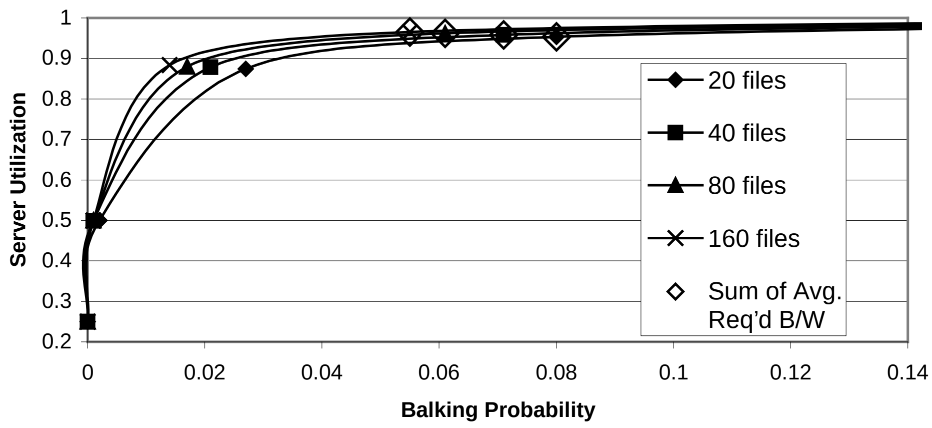


Figure 9. The Tradeoff: Utilization vs. Balking

6. Conclusion And Future Work

In this study, we use CMVA techniques to develop two analytical models, balking model and waiting model, for evaluating balking probability and average waiting time for media servers with limited bandwidth. Based on in-depth analysis of simulation results, we also propose an interpolation of the average service time in the waiting model when measuring the Patching protocol.

Compared to simulation, although the balking model captures the trend of balking probability with changing server bandwidth, it can incur a large relative error when the server utilization is low. The waiting model performs reasonably well for the HMSM protocol, yet for patching, it demonstrates some peculiarity. The simple service time interpolation yields a more accurate estimate of the mean stream duration, which in turn helps the estimation of the mean waiting time.

Based on evaluation results, it is reasonable to take C* as "sweet spots" in the system design space. And as we expected, HMSM is always more effective than patching in bandwidth saving.

Future work can be conducted from the following aspects to further improve and complete this study:

- Investigating why the balking model predictions deviate from simulation results when the server capacity gets large.
- Further model debugging on the waiting model for the optimized patching protocol.
- Applying the analytical models to media servers with multiple categories of files (different playback durations, different popularities and so forth).

Acknowledgements

Special thanks to Professor Mary Vernon for all the help she gave, and to Derek Eager for providing us with the simulator.

References

[CaLo97] Carter, S. and Long, D., “Improving Video-on-Demand Server Efficiency Through Stream Tapping”, *Proc. 6th ICCCN’97*, Las Vegas, NV, Sept. 1997, pp. 200-207.

[EaVZ99] Eager, D., Vernon, M., and Zahorjan, J., “Optimal and Efficient Merging Schedules for Video-on-Demand Servers”, *Porc. 7th ACM Int’l. Multimedia Conf. (ACM MULTIMEDIA ‘99)*, Orlando, FL, Nov. 1999, pp. 199-202.

[EaVZ00a] Eager, D., Vernon, M., and Zahorjan, J., "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand", *Proc. Multimedia Computing and Networking 2000 (MMCN'00)*, San Jose, CA, January 25-27, 2000.

[EaVZ00b] Eager, D., Vernon, M., and Zahorjan, J., "Minimizing Bandwidth Requirements for On-Demand Data Delivery", *Technical Report #1418a*, Computer Science Dept., University of Wisconsin.

[GoLM96] Golubchik, L., Lui, J., and Muntz, R., “Adaptive Piggybacking: A Novel Technique for DataSharing in VideoOn -Demand Storage Servers”, *ACM Multimedia Systems Journal*, 4(3):140--155, 1996.

[HuSh97] Hua, K., and Sheu, S., “Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems”, *Proc. ACM SIGCOMM’97 Conf.*, Cannes, France, Sept. 1997, pp. 89-100.

[Klei76] Kleinrock, L., Queueing Systems, vol. 1, John Wiley & Sons, 1976.