LIMITED DISTRIBUTION NOTICE
(NOT TO BE REMOVED FROM DOCUMENT):

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communication and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

On-Line Video on Demand

Sudhanshu Aggarwal* Juan A. Garay† Amir Herzberg†

Abstract

In this paper we formulate the problem of Video on Demand (VOD) from a resource allocation perspective. In particular, we introduce the *decision element* into a movie-vending environment, which complements the current approaches. In contrast with more the traditional resource allocation problems (such as machine scheduling and call control), the problem possesses the distinctive *batching* property, which stands for the feasibility of several requests being served by one resource (channel).

We investigate the problem in an *on-line* fashion, namely, having to accept or reject a request for a movie without the knowledge of future requests. We show upper and lower bounds on the *competitive ratio* of on-line movie scheduling algorithms for a variety of scenarios (an algorithm is called *competitive* if it performs, up to a constant factor, as well as its off-line, clairvoyant counterparts for the same problem). In particular, for the natural case of *refusal by choice* with *delayed notification*, we present a class of algorithms that exhibit, under certain conditions, an asymptotically optimal behavior.

We also compare the performances of the different algorithms under various distributions of requests over time, and evaluate the effect of various heuristics.

*Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139. E-mail: sonu@mit.edu. Work done while the author was visiting the IBM T.J. Watson Research Center.

†IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598. E-mail: {amir,garay}@watson.ibm.com.

1 Introduction

The area of interactive home video entertainment is actively developing. For example, hybrid networks that enable multi-media connections are being studied as a step toward all-digital video networks [10], as is the problem of bandwidth allocation strategies for combined analog/digital transmission of data over a CATV system [8]. Experiences and services trials are being performed, mostly for *near* VOD, which consists of broadcasting movies at fixed time invervals (e.g., every 15 min) [12, 13]. Also, special hardware and switches are been built for this purpose.

Most published works and on-going projects seem to share a common view on the basic architecture for VOD services, as depicted in Figure 1 in Appendix A. The movies are stored in a central *video server*, which may be connected to other servers over a high bandwidth WAN. The *video server* is connected by a high-capacity fiber link to local distribution centers (hubs), from which coax cables are used to broadcast to the households.

In this architecture, there appears to be two major bottlenecks:

- The limited number of broadcast channels available on the coax cable (shared by many homes). The number of channels per coax may be about 100. It appears that for economical reasons, it may be desirable to put a much larger number of homes on the same broadcast cable, and one may expect up to, say, 1000 concurrent users.
- The number of movies which the server is able to transmit concurrently (see, in particular, [9]).

Previous works and current implementations attempt to achieve, under these constraints, *Full Video on Demand* and/or *Near Video on Demand*:

- **Full Video on Demand:** Whenever a user's request for a movie arrives, it is immediately served, provided there is capacity (a channel) available, rejected otherwise.
- **Near Video on Demand:** A fixed set of movies is played regularly, at fixed time intervals.

Much attention has been given to the hardware requirements for such designs, and the quality of service, due to the to bottlenecks mentioned above. It is immediate that either policy may poorly utilize the available resources. Namely, a considerable number of feasible distribution of requests may cause the first approach to tie up the distribution resources, while the latter provides—by definition—a limited service, and a bad decision (e.g., what movies to show) directly translates into resource wastefulness. The two approaches are shown in Figure 2 in the Appendix.

The purpose of this paper is to investigate the intermediate terrain that lies between the two extreme policies. Indeed, we introduce the *decision element* in the video service architecture, which leads to the concept of:

- **Adaptive Video on Demand:** Upon arrival, a movie request is accepted and served (possibly with some delay), or rejected. The decision as to whether accept or reject (and the amount of delay) is made by a scheduling algorithm.

Note that the scheduling algorithm might specify that the request be rejected, even though there might be channels that are currently not being used.

The approach is also depicted in Figure 2. To the best of our knowledge, this is the first attempt so far to tackle VOD from an optimization perspective. Namely, the purpose of this paper is to optimize the decisions on which movie requests to accept, when to deliver them, and which requests to reject. We therefore try to maximize the utilization of a (limited) resource being shared by a number of users. Our motivation is to complement approaches based on predicting demand for movies (i.e., Near VOD), which are best suited for popular movies, but inappropriate for more unusual selections. This leads us to investigate the problem in an *on-line* fashion, as explained below.

The type of problem we have just described is not unrelated to the more "classical" problems of (parallel) machine scheduling (e.g., [5, 15]) call control [3, 6, 7], and virtual circuit routing [1, 2], to mention a few. However, our problem has the distinctive property of allowing the movie scheduling algorithm, under certain circumstances, to serve several users simultaneously on the same channel.

The general nature of the case of unusual selections is such that it is always possible to "second guess" decisions made in the past. In other words, a decision made previously to accept a request, may have been wrong from the service provider point of view, because it may cause a channel to be used up, thus preventing a subsequent more "valuable" set of requests (for example, for a more popular movie) to be rejected. Thus, the *on-line* nature of the problem (the fact that decisions are to be made when requests for movies arrive into the system without knowledge of future requests) might lead to significantly lower efficiency than would have been possible with full *off-line* knowledge of the entire pattern of request arrivals.

This motivates the emphasis we put on this issue. Namely, in this paper we investigate movie scheduling mechanisms in an *on-line* fashion. Generally speaking, an on-line problem is one in which an algorithm that solves it must handle a sequence of requests, satisfying each request without knowledge of the future requests. We evaluate on-line algorithms in terms of their *com-*

petitiveness [14]. [1] An algorithm is said to be κ-competitive, if its performance on *any* sequence of requests is within a factor κ of the performance of any other algorithm on the same sequence, including the *off-line* algorithms for the problem—which can "see" all future requests. A measure of performance of movie-scheduling algorithms is the *revenue* they produce. (We can assume that the operating costs are fixed, so the profit is a function of the revenue only.) Let c denote the number of available channels; m the number of movies; and u the number of users. Also let $\mu \stackrel{\text{def}}{=} u/c$. Our results are summarized in Table 1.

COMPETITIVE RATIO	*No refusal by choice*	*Refusal by choice*	
		Immediate Notification	*Delayed Notification*
Lower bound	μ	$\frac{u}{c+1} \approx \mu$	$\Omega(\ln \mu)$
Algorithm	μ	μ	$O(\ln \mu)^{\dagger}$

Table 1: Our results

† For adequate values of c.

Regarding the terminology of Table 1, one important property of Adaptive `VOD` is the possibility of a request being rejected even though there is available capacity to serve it. We call this property *refusal by choice*. We show that if refusal by choice is prohibited, then the competitive ratio of any Adaptive `VOD` algorithm cannot improve compared to the Full `VOD` approach. Another relevant parameter is the maximum time the system is allowed to spend (respectively, the time a user can be made to wait) before the user is informed of the decision on his/her request (i.e., accepted or rejected). We thus distinguish between systems with *immediate* or *delayed notification*. For the more natural case of refusal by choice with delayed notification, we show a lower bound of $\Omega(\ln \mu)$ on the performance of any algorithm, and introduce a class of algorithms (which we call `Pre-partition`) that, for big enough number of channels and of notification time, exhibit an asymptotically optimal behavior. In particular, we analyze the *Harmonic* version, which outperforms the more straightforward exponential algorithms by a constant factor in the worst case.

We note that in reality, the sequence of requests may depend on the decisions of the algorithm. For example, suppose that once a movie is scheduled, it is offered at a discount to new viewers willing to see at the already scheduled time. It is likely that the probability of new requests to this movie would temporarily increase until the scheduled time, and then it may drop. We have also

[1] A second motivation for tackling a new problem with this type of analysis is the hope of achieving a deeper, meaningful understanding. Indeed, this has allowed us to come up with a `VOD` scheme that we believe has the potential of practical development [4].

compared the performances of the different algorithms under various distributions of requests over time, and evaluated the effect of various heuristics. Our studies of these more realistic scenarios show an even stronger performance advantage of Harmonic compared to Exponential, Near, and variants of Full VOD. Some of these results are discussed at the end of the paper.

Finally, we remark that we consider the special case of *non-preemptive* video scheduling algorithms, and work exclusively with *flat* charging policies, although many of our results can be generalized to hold for other policies. (Figure 3 depicts two possible charging policies.) The remainder of the paper is organized as follows. In Section 2 we provide a more technical description of the model and necessary definitions. Our lower bounds and algorithms for the on-line case are presented in Section 3. We conclude with the presentation of some results for other scenarios and directions for future work.

2 Model and Definitions

In this work we concentrate on the problem of efficient sharing of a limited broadcast capability, controlled and supplied by a single server. This model is represented in Figure 4. This seems to be a reasonable model for the two likely bottlenecks for VOD identified in the introduction, namely the shared coax cable and the output capacity of the video server (see Figure 1). The model considers finite, fixed sets of movies, users and channels, denoted respectively M, U and C. Denote the cardinality of these sets by the corresponding lower case letter, i.e. m, u and c.

A *request* is a tuple $(time, user, movie)$ where $time$ is a positive number, $user \in U$ and $movie \in M$. Similarly, a response is a tuple $(time, user, movie, channel, servetime)$ where $channel \in C$ and $servetime$ is a real number. If $servetime$ is negative, we say that the response is negative (or a refusal).

The system has three positive parameters related to time:

- T, the duration of a movie (for simplicity assume it is the same for all movies). For simplicity, we also assume that no user can place more than one request in each interval of length T.
- τ, the maximal delay between a request and the start of the movie (if it is accepted),
- ν, the maximal delay between a request and its response (notification).

We assume $\nu \leq \tau \leq T$. A sequence of requests (t_i, u_i, m_i) is *valid* if the time values t_i are monotonously increasing and there is at least time T between any two subsequent requests of the same user.

Intuitively, the movie scheduling algorithm determines the responses at any moment, based on the requests up to that moment. More formally, a *scheduling algorithm* is a function from a time, a state, and a sequence of requests (all before the given time), to a time (for next wakeup, if no requests would occur before), a new state, and possibly one or more responses.

We consider different classes of movie scheduling algorithms, based on the definitions above. An algorithm $\mathcal{A}$ is said to *refuse by choice* if in some run it has a response $(t_i, u_i, m_i, c_i, s_i)$ with $s_i < 0$, while there is some free channel at t_i. A channel c_k is *free* at time t if there is no response $(t_k, u_k, m_k, c_k, s_k)$ with $t_k \leq t$ and for which $[s_k, s_k + T]$ intersects with $[t, t + T]$. An algorithm $\mathcal{A}$ is said to perform *immediate notification* if each request must be "decided upon" as soon as it is presented. (Note that if a request is accepted, $\mathcal{A}$ must serve it within time τ.) Algorithms that do not always notify immediately are said to perform *delayed notification*, and in this case they have to notify within the parameter ν defined above.

We now give a more technical definition of competitive ratio. A measure of performance of movie schedulers is the *revenue* they produce. A *revenue function* is a function that maps integers (delay) to the rationals (income). The revenue of a response $(t_i, u_i, m_i, c_i, s_i)$ corresponding to a request (t_j, u_i, m_i) is $rev(s_i - t_j)$ if $s_i \geq 0$, and 0 otherwise. The revenue of a run α, denoted $rev(\alpha)$, is the sum of the revenues of all the requests in the run. Let $\mathcal{A}(\rho)$ denote the run produced by algorithm $\mathcal{A}$ when presented with request sequence ρ (we assume that $\mathcal{A}$ is deterministic). The *competitive ratio* of online algorithm $\mathcal{A}$ is defined as

$$\phi(\mathcal{A}) = \sup_{\mathcal{A}^*, \rho} \frac{rev(\mathcal{A}^*(\rho))}{rev(\mathcal{A}(\rho))}$$

where $\mathcal{A}^*$ is an offline algorithm and ρ is a sequence of requests. It is sometimes convenient for the analysis to describe things in terms of a game between a player (the online algorithm) and an *adversary* (the offline algorithm), whose goal is to produce a strategy (i.e., a request sequence) that would force the player to perform poorly.

3 Our Results

We now show upper and lower bounds on the competitive ratio $\phi(\mathcal{A})$ of online movie scheduling algorithms $\mathcal{A}$ (see Table 1). Intuitively, there must exist algorithms with a competitive ratio of μ, because by "bunching up" requests, the adversary can serve at most u requests for every c requests the online algorithm serves. We first formalize this argument and show that there indeed exists a simple algorithm $\mathcal{A}_1$ with a competitive ratio of μ. This algorithm is essentially the same as the Full `VOD` approach.

We then show that any algorithm $\mathcal{A}$ which is *not allowed refusal by choice* must have a competitive ratio of at least μ. Thus, no online strategy can, in the worst case, "gain" anything by bunching up requests with respect to the simple strategy $\mathcal{A}_1$, unless it is allowed to reject requests. Next, we show that even refusal by choice, by itself, is not enough to improve worst-case performance. The timing of *notification* also determines how well an algorithm can do. Specifically, we show that an algorithm that performs *immediate notification* must have a competitive ratio of at least $\frac{u}{c+1}$ (which is nearly equal to μ). Thus, such algorithms do not have any advantage, in the worst case, over those that do not refuse by choice.

Finally, we consider the case of algorithms which are allowed to refuse by choice together with *delayed notification*. As expected, they may perform substantially better than those with immediate notification. More specifically, we show a lower bound of $\Omega(\ln \mu)$ on the performance of such algorithms, and present a simple algorithm ($\mathcal{H}$, for *Harmonic*) which matches this bound in the case $\nu = \tau$. Proofs of the theorems are omitted from the main body of the paper, and appear in Appendix B.

3.1 No refusal

In this section we show that a competitive ratio of μ is easy to achieve by giving a simple "naive" algorithm that achieves this ratio (and which is not allowed to refuse requests by choice). For simplicity, we consider the special case of the maximum service delay τ being 0; the proof is easily extended to any arbitrary delay τ. Consider the following algorithm $\mathcal{A}_1$: Given request sequence $\rho = r_1, r_2, r_3, ..., r_n$ (as defined in the previous section), $\mathcal{A}_1$ assigns r_i to any channel that is free at time t_i; otherwise r_i is rejected.

Theorem 1 *Algorithm $\mathcal{A}_1$ is μ-competitive.*

The following theorem shows that, under the circumstances, this is the best that can be done.

Theorem 2 *Let $T > \tau$. Any algorithm $\mathcal{A}$ that is not allowed to refuse by choice must have a competitive ratio $\phi(\mathcal{A}) \geq \mu$.*

3.2 Refusal by choice and immediate notification

One might wonder whether an algorithm, when allowed to reject movies, could perform substantially better. As the following theorem shows, that is not the case: Refusal by choice, by itself, is not enough to improve worst-case performance. The timing of notification also determines how well an algorithm can do.

Theorem 3 *Any algorithm $\mathcal{A}$ capable of refusal by choice, but which is required to notify immediately, must have a competitive ratio $\phi(\mathcal{A}) \geq \frac{u}{c+1}$.*

The above lower bound can be readily extended to non-zero (but small) values of ν, namely, for all $\nu < \frac{\tau}{c\kappa}$. We leave the details for the full version of the paper.

3.3 Refusal by choice and delayed notification

In this section we give lower and upper bounds on the competitive ratio of movie scheduling algorithms that are allowed refusal by choice and delayed notification. For the case of the upper bound we also assume big, but yet reasonable, values of c. We first show a lower bound of $\ln \mu/2$ on the competitive ratio of such algorithms for the case of short notification times (specifically, $\nu < \frac{2\tau}{\mu}$).

Let $\mathcal{A}$ be an online algorithm. We prove the result using a particular type of game between $\mathcal{A}$ and its adversary $\mathcal{A}^*$. Suppose we need to show that $\mathcal{A}$ cannot have a competitive ratio of κ (which may be a function of u and c). The structure of the game will be as follows. The adversary will present $\mathcal{A}$ with several sets $R_1, R_2, ...$ of requests in quick succession. When presented with each set of requests, $\mathcal{A}$ has to schedule just enough to maintain a competitive ratio of κ. However, the adversary chooses the request sets in such a manner that it is able to force $\mathcal{A}$ to accept "cheap" requests (i.e., requests which require a relatively large number of channels to serve), and thereby "block" its channels to further, more remunerative requests, which are served by the adversary. If the adversary $\mathcal{A}^*$ ends up serving at least κ times more requests than $\mathcal{A}$, we will have shown that $\mathcal{A}$ cannot have a competitive ratio of κ. The proof details can be found in Appendix B. This allows us to state:

Theorem 4 *Let $\nu < \frac{2\tau}{\mu}$. Any algorithm $\mathcal{A}$ capable of refusal by choice and delayed notification must have a competitive ratio $\phi(\mathcal{A}) > \ln \frac{\mu}{2}$.*

The $\Omega(\ln \mu)$ bound can be extended to hold for all $\nu \leq \tau$, at the expense of a bigger user space. (Of course, the bound can also be readily strengthened to μ for small values of c.)

3.3.1 Prepartition algorithms

In this section we present algorithm $\mathcal{H}$ which, when allowed delayed notification and refusal by choice, and given a large enough (but reasonable) number of channels, matches (asymptotically) the lower bound of the previous section. More specifically, $\phi(\mathcal{H}) \leq 1 + 8H_\mu$, where H_μ is the μth harmonic number.

Roughly speaking, $\mathcal{H}$ divides the c channels into μ sets $S_1, S_2, ...S_\mu$. Each set S_i contains $\frac{c}{iH_\mu}$ channels (in fact, $|S_i| = \lfloor \frac{c}{iH_\mu} \rfloor$). Thus, the total cardinality of the sets is $\frac{c}{H_\mu} \cdot (1 + 1/2 + ... + 1/\mu) = c$. The algorithm stipulates that any transmission made on a channel in set S_i must serve at least i requests (see Figure 5).

$\mathcal{H}$ schedules the arriving requests as follows: Each movie m_j is associated with a distinct *queue* Q_j of requests for that movie. Initially, $Q_j = nil$ for all j. When a request for m_j is made, it is immediately inserted into Q_j. The purpose of Q_j is to accumulate as many requests as possible for movie m_j, so that it can serve all of them with a single transmission at the end. Each queue Q_j is associated with a "start time" $start(Q_j)$, representing the time at which the first request in the queue was made.

At time $start(Q_j) + \nu$, the algorithm decides whether to serve the requests in Q_j (say k in number). If there exists a free channel in a set S_i with index $i \leq k$, it immediately notifies and begins a transmission for m_j on one such channel, thus serving all k requests, and Q_j is re-initialized to *nil*. If, on the other hand, there exists no such channel, $\mathcal{H}$ rejects only those requests in Q_j made at time $start(Q_j)$, deletes them from Q_j, and resets $start(Q_j)$ to the time of the earliest request now in Q_j. We are able to state the following.

Theorem 5 *Let $\nu = \tau$ and $c > \mu \ln \mu$. Then $\phi(\mathcal{H}) = O(H_\mu)$, where H_μ is the μth harmonic number.*

4 Discussion

In this paper we have formulated the problem of Video on Demand (VOD) from a resource allocation perspective. We have investigated the problem in an *on-line* fashion, and established upper and lower bounds on the competitive ratio of on-line movie scheduling algorithms for a variety of scenarios. For the natural case of refusal by choice with delayed notification, we leave open achieving a logarithmic upper bound for the case of smaller notification times. A certain type of algorithm introduced in [4] looks promising in this direction, and is the subject of current work.

In the previous section we have analyzed VOD from a worst-case perspective. We have also compared the performances of the different algorithms under various distributions of requests over time, and evaluated the effect of various heuristics. For example, it is expected that the number of requests for movies in a VOD system would peak during "prime time" hours. Define the *load* of such a system as the number of incoming requests in a given time interval. Alternatively, this can be expressed as $x \cdot c$, where x is the *loading factor*. Figure 6 depicts one plausible load variation over a period of 8τ.

Let π denote the probability that a user will place a request for a movie, given that the movie is to be shown in some channel. Since we are concerned with unusual selections/rare movies, π would typically be a small number. Figure 7 shows the performance (i.e., expected number of users served) as a function of π achieved by the different approaches: Near, Full, Full$^+$ VOD, and Harmonic from the previous section. (Full$^+$ is the extension of Full that waits for time τ before serving a request.) In the example, $c = 1000$ and $u = 10000$. The comparison is done under the load conditions of Figure 6. For the purpose of Harmonic, we model the probability of i users showing interest for a (rare) movie to be shown in some channel as $\frac{\pi^i}{i!}$. The poor performance of Near VOD (i.e., $\pi \cdot c$) is expected, as the method is aimed at popular movies. Harmonic also substantially outperforms the more straightforward Exponential algorithm (not shown).

Finally, we remark that in this work we have implicitly assumed that all requests for movies are made in "real time," i.e., ideally the customer would like the request to be honored "as soon as possible." A natural extension is to allow for *reservations*, i.e., requests which are done substantially ahead of the time desired for the delivery of the movie. Some additional natural extensions would be to investigate requests (movies) of different length (cf. [11]), different capacity requirements (e.g., HDTV vs. regular TV quality), dynamically changing capacity requirements (e.g., multi-media documents), and multiple stages of video servers and capacity-limited channels.

Acknowledgements

The authors would like to thank Anna Karlin for helpful comments on a preliminary version of the manuscript.

References

[1] B. Awerbuch, Y. Azar and S. Plotkin. Throughput-Competitive On-Line Routing. *Proc. 34th Annual IEEE Symp. on Found. of Computer Science*, 32–40, 1993.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-Line Load Balancing with Applications to Machine Scheduling and Virtual Circuit Routing. *Proc. 25th Annual ACM Symp. on Theory of Computing*, 623–631, 1993.

[3] B. Awerbuch, Y. Bartal, A. Fiat and A. Rosen. Competitive Non-Preemptive Call Control. *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, 312–320, January 1993.

[4] A. Bar-Noy, J. Garay and A. Herzberg. Shared Video on Demand. Submitted to *INFOCOM 95*.

[5] R.L. Graham. Bounds on multiprocessing anomalies. *SIAM Journal of Applied Mathematics*, 17:263–269, 1969.

[6] J. A. Garay and I. S. Gopal. Call Preemption in Communication Networks. *Proc. IEEE INFOCOM '92*, 1043–1050, May 1992.

[7] J.A. Garay, I. Gopal, S. Kutten, Y. Mansour and M. Yung. Efficient On-line Call Control Algorithms. *Proc. 2nd Israeli Symp. on Theory of Computing and Systems*, Netania, Israel, 285–293 June 1993.

[8] Colin J. Horton. 110 channels without boundaries: why restrict options? *Communications Engineering and Design*, 19/1:29–30, January 1993.

[9] Steven M. McCarthy. Integrating Telco Interoffice Fiber Transport with Coaxial Distribution. *Proc. SPIE - Int. Soc. Opt. Eng.*, 1786:23–33, November 1993.

[10] John Koegel and Andrzej Syta. Routing of multi-media connections in hybrid networks. *Proc. SPIE - Int. Soc. Opt. Eng.*, 1786:2–10, November 1993.

[11] Richard J. Lipton and Andrew Tomkins. Online Interval Scheduling. Private communication.

[12] Kenneth Metz. Next generation catv networks. *Proc. SPIE - Int. Soc. Opt. Eng.*, 1786:184–189, November 1993.

[13] Chris Sell. Video on demand internal trial. *Proc. SPIE - Int. Soc. Opt. Eng.*, 1786:168–175, November 1993.

[14] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 32:652–686, 1985.

[15] D. Shmoys and J. Wein and D. Williamson. Scheduling Parallel Machines On-line. *Proc. 32nd IEEE Symp. on Foundtations of Computer Science*, 131–140, October 1991.

Appendix A: Figures

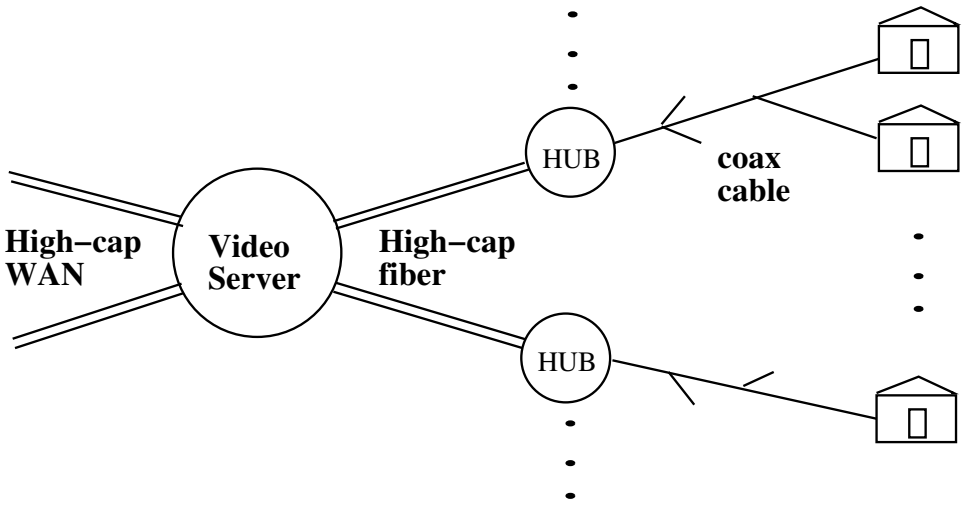

Figure 1: Video-on-Demand architecture.

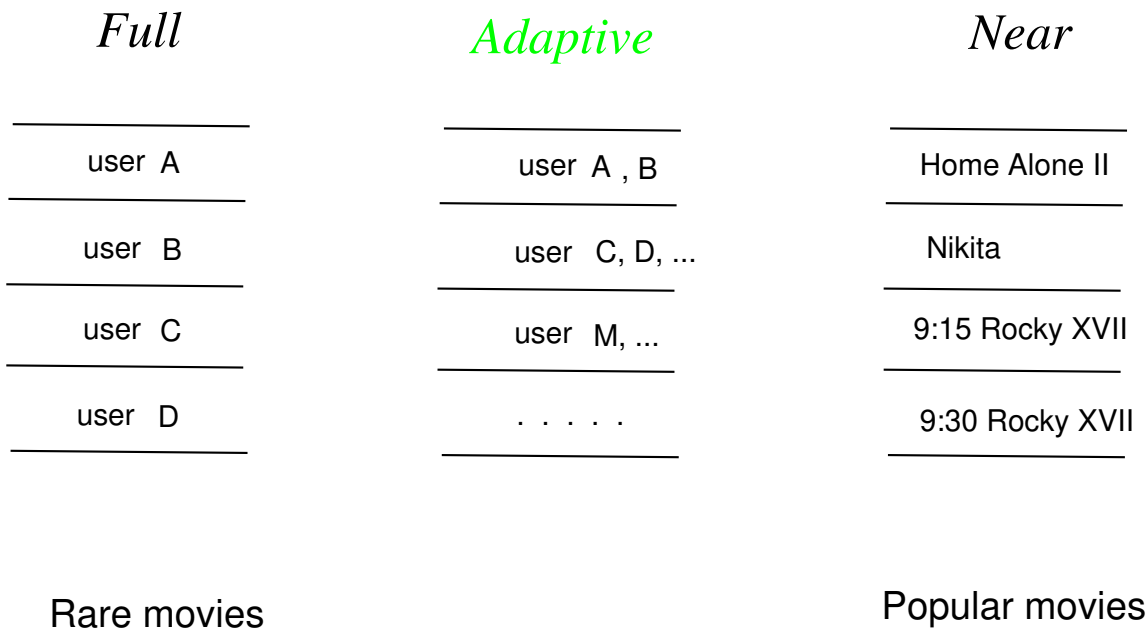

Figure 2: VOD approaches.

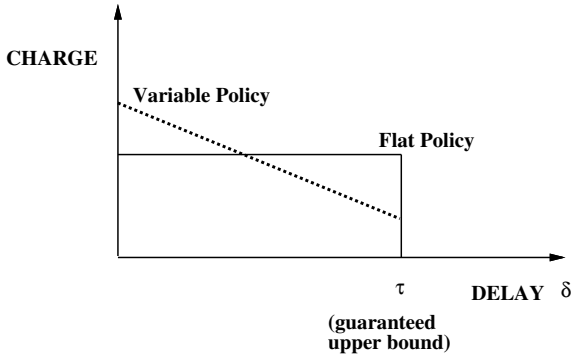

Figure 3: Two possible charging policies.

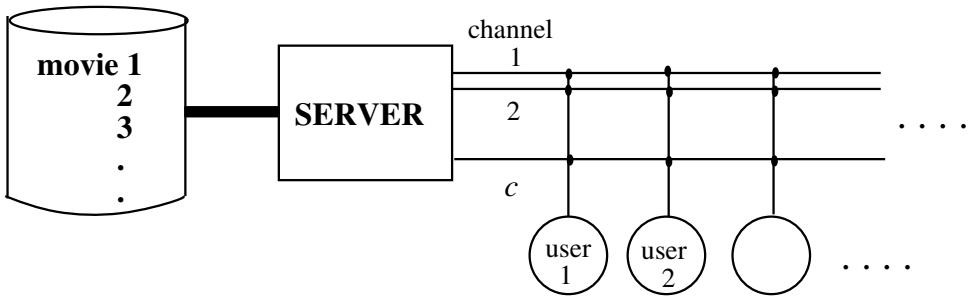

Figure 4: Model for Video On Demand.

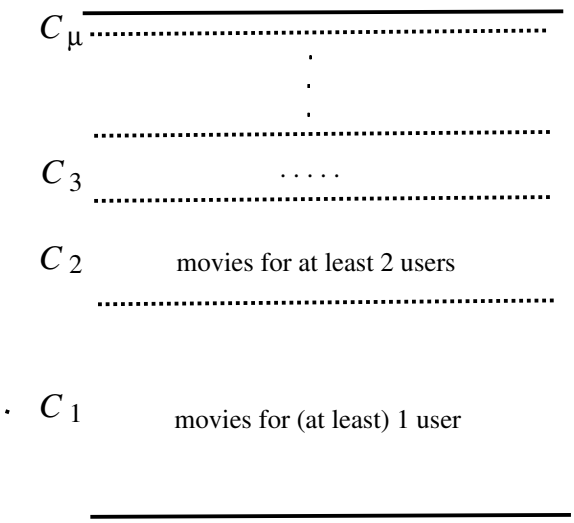

Figure 5: The *Harmonic Prepartition* algorithm.

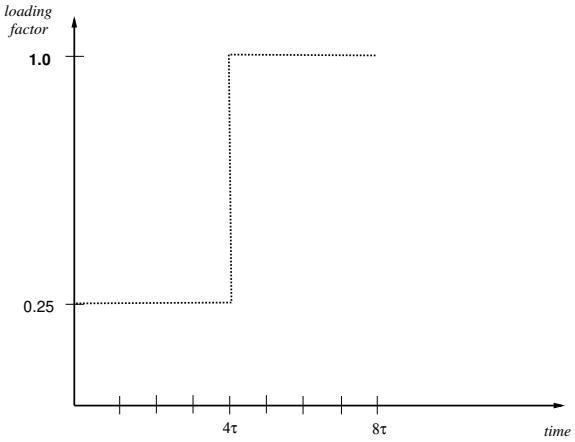

Figure 6: Load variation in a VOD system.

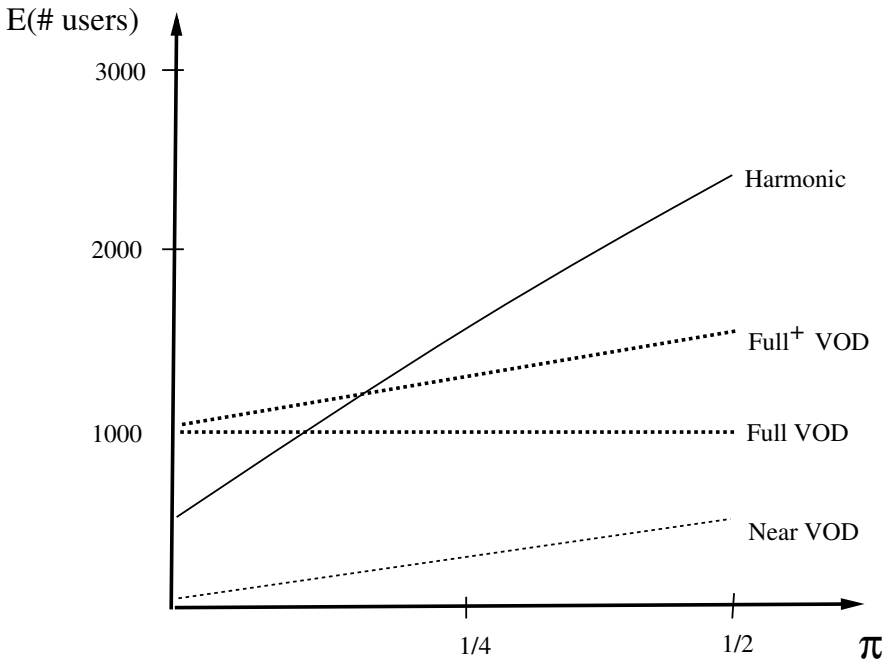

Figure 7: Experimental comparison of VOD approaches.

Appendix B: Proofs

We repeat here the statements of the theorems for convenience.

Theorem 1: *Algorithm $\mathcal{A}_1$ is μ-competitive.*

Proof: Each request in ρ is either accepted or rejected. Let n_A and n_R be the number of accepted and rejected requests, respectively. Thus, $n = n_A + n_R$.

If request r_i is rejected, it is rejected because exactly c previously accepted requests are being served at that instant. For each of these c requests r', we say r_i *is rejected because of* r'. Define the "rejection graph" to be the graph whose vertices consist of all n requests, and for any pair of vertices r_a and r_b, there is a link connecting them if and only if r_a is rejected because of r_b, or vice-versa. (Note that each rejected request must have exactly c incident links, and that the total number of links incident on all accepted requests must equal the total number of links incident on all rejected requests.)

Now we claim that each accepted request cannot have more than $u - c$ incident links. For suppose there exists an accepted request r_i which has $u-c+d$ incident links, i.e., $u-c+d$ requests (comprising, say, set R) are rejected because of r_i. Consider the latest request in set R, say, request r_l at time t_l. Note that $t_l < t_i + T$. Thus, there were exactly $u - c + d$ rejections in the time interval $[t_i, t_l]$. Further, since r_l was rejected, there were exactly c acceptances in the time interval $(t_l - T, t_l]$. It follows that there were at least $(u - c + d) + c = (u + d)$ requests in $(t_l - T, t_l]$, which contradicts the fact that each user of the u users may have at most one request during $[t_l - T, t_l]$.

Thus, the total number of links incident on accepted requests is at most $n_A(u - c)$, and the number incident on rejected requests is exactly $n_R c$. As a consequence, we have $n_A(u - c) \geq n_R c$, or $n_A u \geq nc$, which implies $\frac{n}{n_A} = \phi(\mathcal{A}_1) \leq \frac{u}{c}$. □.

Theorem 2: *Let $T > \tau$. Any algorithm $\mathcal{A}$ that is not allowed to refuse by choice must have a competitive ratio $\phi(\mathcal{A}) \geq \mu$.*

Proof: Let $\mathcal{A}$ be an algorithm that does not refuse by choice. We construct request sequences ρ that would necessitate a competitive ratio of at least $\mu = u/c$.

Let ρ be a request sequence for $\mathcal{A}$, which begins with a set of c requests for c different movies, presented to $\mathcal{A}$ at time 0. Assume there is no additional request till time ν. Algorithm $\mathcal{A}$ must schedule all of the first c requests until time ν, since it does not refuse by choice. Furthermore, the schedules should all start before time τ.

We now have two mutually exclusive cases:

1. Suppose there at least one movie is scheduled before τ; let m_i be such a movie. Then ρ would contain $u - c$ requests for the same movie m_i, all arriving exactly at time τ. However, at this

time all channels are already scheduled, and since we do not allow preemption, the online algorithm would have to refuse all of these movies.

Thus, $\mathcal{A}$ honors a total of c requests in its run. On the other hand, obviously there exists an offline algorithm $\mathcal{A}^*$ which can serve all requests. Thus, $\phi(\mathcal{A}) = u/c$.

2. Suppose that $\mathcal{A}$ schedules each of the c movies to begin at time τ. The request sequence ρ would continue with $u - c$ requests at time $T - \frac{\tau}{2}$ for some movie. Of course, algorithm $\mathcal{A}$ cannot honor any of these requests since all channels are still occupied at that time. Thus, $\mathcal{A}$ honors a total of c requests in this run.

   On the other hand, there exists an offline algorithm $\mathcal{A}^*$ that honors all requests by honoring the first set immediately and the second set at time T (within the permitted delay). Thus, again, $\phi = u/c$.

□

Theorem 3: *Any algorithm $\mathcal{A}$ capable of refusal by choice, but which is required to notify immediately, must have a competitive ratio $\phi(\mathcal{A}) \geq \frac{u}{c+1}$.*

Proof: Let $\mathcal{A}$ be an algorithm that is al!l lowed to refuse requests by choice, but which always notifies immediately. Let it have a competitive ratio $\phi(\mathcal{A}) = \kappa$. The proof is by contradiction. It is based on the fact that if $\mathcal{A}$ has a lower competitive ratio, the adversary can construct a request sequence that makes $\mathcal{A}$ violate the ratio. The adversary first presents a series of "baits" such that $\mathcal{A}$ is forced to fill up its channels (with non-lucrative, single-user transmissions) in order to maintain its competitive ratio. Once $\mathcal{A}$ is filled up, the adversary then presents as many requests as possible; thus, the adversay can take all of these while the $\mathcal{A}$ can take none.

More specifically, the adversary first presents a request sequence $\rho' = r_1, r_2, ...r_l$. Each request r_i is a request for movie m_i made at time $i\epsilon$, where ϵ is some arbitrarily small interval of time. The adversary constructs the request sequence according to the following rules:

1. If r_i is rejected by $\mathcal{A}$, $m_{i+1} = m_i$;
2. if r_i is accepted by $\mathcal{A}$, $m_{i+1} \notin \{m_1, m_2, ...m_i\}$; and
3. the total number of different movies requested in ρ' is c.

Thus, the adversary presents individual requests for movie m (at instants separated by ϵ) until one of these is accepted by $\mathcal{A}$, then it presents a group of individual requests for another movie m' until one is accepted, and so on, for a total of c request groups for c different movies. The total number of requests accepted by $\mathcal{A}$ is c (the last one in each of the c groups).

On the other hand, by "bunching up" the requests in each group and transmitting a single movie to serve an entire group, the adversary can accept each one of these j requests. Since $\mathcal{A}$ must maintain a competitive ratio of κ, $\mathcal{A}$'s ith acceptance must occur by request $r_{i\kappa}$. Thus, the total number of requests is $l \leq c\kappa$. Note that since there are c groups of requests of total size $l \leq c\kappa$, there must exist a group G of size $g \leq \kappa$.

Having "filled up" all the channels of $\mathcal{A}$, the adversary now presents a request sequence $\rho'' = r_{l+1}, r_{l+2}, ...r_u$ for any one of the movies not requested so far. It accepts all the requests from the earlier sequence ρ', except for the $g \leq \kappa$ requests in group G. Thus, at this point the adversay has one channel free; it accepts all the requests in ρ'' and serves them on this channel.

The online algorithm $\mathcal{A}$ has served exactly c requests, while the adversary has served $(l-g)$ requests from ρ' and $(u-l)$ from ρ'', for a total of $(u-g) \geq (u-\kappa)$. Thus, to preserve a competitive ratio of κ, a necessary condition is

$$\kappa \geq \frac{u-\kappa}{c} \ ,$$

which reduces to

$$\kappa \geq \frac{u}{c+1} \ .$$

□

Theorem 4: *Let $\nu < \frac{2\tau}{\mu}$. Any algorithm $\mathcal{A}$ capable of refusal by choice and delayed notification must have a competitive ratio $\phi(\mathcal{A}) > \ln \frac{\mu}{2}$.*

We first describe the game in more detail, and establish some technical facts. Suppose there is a κ-competitive online movie scheduler algorithm $\mathcal{A}$. Consider the following sequence of requests. At time 0, request set R_1 is presented, which contains one request for each of c movies. $\mathcal{A}$ must eventually serve at least $\frac{c}{\kappa}$ of them (in fact, $\lceil \frac{c}{\kappa} \rceil$) to maintain a competitive ratio of κ, otherwise $\mathcal{A}^*$ could serve all of them and have a ratio greater than κ. We say that $\mathcal{A}$ has *committed* to at least $\frac{c}{\kappa}$ requests. Note that $\mathcal{A}$ must do so by time ν.

At time ν, two requests for each of c movies are presented which were not of the $\frac{c}{\kappa}$ already committed. More precisely, for each of the $c - \frac{c}{\kappa}$ movies not committed there is an additional request, plus two new requests for $\frac{c}{\kappa}$ movies which were not requested before. Let the additional requests presented at time ν comprise set R_2. Again, $\mathcal{A}$ must serve at least $\frac{c}{\kappa} \cdot \frac{1}{2}$ movies (each with 2 requests), otherwise $\mathcal{A}^*$ could satisfy all and accrue $2c$. In other words, the adversary would have gotten revenue of $2c$ while $\mathcal{A}$'s revenue would be $\frac{c}{\kappa}$ from the movies taken by time ν plus *less than* $\frac{c}{\kappa}$ from the additional movies taken by time 2ν (two movies per channel, but less than $\frac{c}{2\kappa}$ channels).

This generalizes to request set R_i at time $(i-1)\nu$ in the following way: At time $(i-1)\nu$ there are i requests for each of c movies which are not yet being served. Again, old outstanding requests

are "reused." After set R_{i-1} at time $(i-2)\nu$ is presented, $\mathcal{A}$ has to serve $\frac{c}{(i-1)\kappa}$ movies, each with $i-1$ requests. There are $c - \frac{c}{(i-1)\kappa}$ movies left, with $i-1$ outstanding requests; the adversary will just add one additional request for each one of these movies. For the remaining $\frac{c}{(i-1)\kappa}$ channels, the adversary will produce i requests for new movies. Thus, the total number of new requests is $(c - \frac{c}{(i-1)\kappa}) + i \cdot \frac{c}{(i-1)\kappa} = c + \frac{c}{\kappa}$. Again, $\mathcal{A}$ must satisfy at least $\frac{c}{\kappa} \cdot \frac{1}{i}$ of these requests, otherwise the adversary could satisfy them all and beat the competitive ratio. If this was the case, the adversary would have gotten revenue of $i \cdot c$, while $\mathcal{A}$'s revenues would be $\frac{(i-1)c}{\kappa}$ from the movies taken before time $(i-1)\nu$, plus *less than* $\frac{c}{\kappa}$ from the movies taken from R_i at time $(i-1)\nu$ (i movies per channel but less than $\frac{c}{i \cdot \kappa}$ channels). Thus, the adversary would achieve a ratio greater than κ.

This process continues until $\mathcal{A}$ exhausts its supply of channels. This happens after the kth set when

$$\sum_{i=1}^{k} \frac{c}{i \cdot \kappa} \geq c \ ,$$

hence

$$H_k = \sum_{i=1}^{k} \frac{1}{i} \geq \kappa \ , \tag{1}$$

where H_k is the kth harmonic number.

We also need to make sure for our argument to hold that the number of requests is not too large to exhaust the supply of users in a period of length T. The size of the first request set is c, while the size of each of the $k-1$ subsequent sets is $(c + \frac{c}{\kappa})$. Thus, the total number of requests needed by this strategy is $c + (k-1)(c + \frac{c}{\kappa})$, which must be less than u. I.e.,

$$c\ [1 + (k-1)(1 + \frac{1}{\kappa})] \leq u \ ,$$

which imposes the following condition on k:

$$k + \frac{k-1}{\kappa} \leq \mu \ . \tag{2}$$

Once the adversary has exhausted the online algorithm's supply of channels after presenting R_k, it can serve all the remaining users on one channel. In order for the adversary to take advantage of the situation by accepting requests that $\mathcal{A}$ cannot, we also require that the acceptance/rejection notifications for the kth set of requests be issued *before* time T. This will require:

$$k\nu < \tau \ . \tag{3}$$

Just after set R_k is presented, there exist kc unsatisfied requests (k requests for each of c movies). $\mathcal{A}$ has already committed to a total of $\frac{(k-1)c}{\kappa}$ requests. $\mathcal{A}$ serves $\frac{c}{\kappa}$ more requests at the kth round,

for a total of $\frac{kc}{\kappa}$ requests. On the other hand, $\mathcal{A}^*$ serves all the kc unfulfilled requests. In addition, it presents set R_{k+1} at time $k\nu$, which contains as many requests as possible (to make a total of u requests), all for one of the movies that are not been served by $\mathcal{A}$. At this point, the only requests not served by $\mathcal{A}^*$ are the $\frac{(k-1)c}{\kappa}$ requests that $\mathcal{A}$ committed to. Thus, the adversay is able to serve a total of $u - \frac{(k-1)c}{\kappa}$ requests.

In order for algorithm $\mathcal{A}$ to preserve a competitive ratio of κ, we must have

$$\kappa \geq \frac{u - (k-1)\frac{c}{\kappa}}{\frac{kc}{\kappa}} = \frac{u\kappa - (k-1)c}{kc} = \frac{\kappa}{k}\mu - (1 - \frac{1}{k}) \ ,$$

which implies

$$(1 - \frac{1}{k}) \geq \kappa(\frac{\mu}{k} - 1) \ ,$$

or

$$\kappa \leq \frac{1 - 1/k}{\mu/k - 1} \ . \tag{4}$$

We now proceed to prove the bounds on the competitive ratio.

Proof: Suppose there exists an online algorithm $\mathcal{A}$ with competitive ratio $\phi(\mathcal{A}) = \kappa \leq \ln \mu/2$. The adversary chooses $k = \lfloor \mu/2 \rfloor$. Since $\forall k, \ln(k+1) \leq H_k$, we have $\ln\lceil \mu/2 \rceil \leq H_k$. Thus, $\kappa \leq H_k$, and constraint (1) is satisfied.

Regarding constraint (2), since $\kappa \geq 1$ (by definition), we have $(k + \frac{k-1}{\kappa}) \leq 2k$, always. If $k = \lfloor \mu/2 \rfloor$, we obtain $2k \leq \mu$, so constraint (2) is also satisfied. Thus the "game" can be completed. Finally, the assumption (stated earlier) that $\nu < 2\tau/\mu$ allows constraint (3) to be satisfied. We now check whether (4) is satisfied. The RHS of (4) is now

$$\frac{1 - \frac{1}{\lfloor \mu/2 \rfloor}}{\frac{\mu}{\lfloor \mu/2 \rfloor} - 1} \leq \frac{1 - \frac{1}{\mu/2}}{\frac{\mu}{\mu/2} - 1} = 1 - \frac{2}{\mu} \ .$$

Thus, the RHS is always smaller than 1. Since the LHS of (4), κ, is larger than 1, (4) fails to hold, and therefore no algorithm can have a competitive ratio of $\kappa \leq \ln \mu/2$. □

Remark: The $\Omega(\ln \mu)$ bound can be extended to hold for all $\nu \leq \tau$, at the expense of a bigger user space.

Theorem 5: *Let $\nu = \tau$ and $c > \mu \ln \mu$. Then $\phi(\mathcal{H}) = O(H_\mu)$, where H_μ is the μth harmonic number.*

Proof: We define the *saturation level* at instant t to be the highest i such that *all* channels in sets $S_1, S_2, ...S_i$ are occupied at time t. The saturation level of an *interval* of time is the *highest* saturation level achieved during the interval.

Consider any execution of $\mathcal{H}$ in response to a request sequence ρ, and a corresponding execution of an offline algorithm $\mathcal{A}^*$ in response to the same request sequence. We divide the entire execution

into *intervals* $I_0, I_1, ...I_l$ of length T (the duration of a movie). Interval I_j spans the half-open time interval $[jT, (j+1)T\ )$.

For all $j = 0, 1, ...l$, let $A(j)$ be the number of requests accepted by $\mathcal{H}$ in interval I_j, and $R(j)$ the number of requests that were *rejected by* $\mathcal{H}$ *in* I_j *but accepted by* $\mathcal{A}^*$ *in its execution.* Let σ_j be the saturation level of interval I_j. Thus, there exists some instant $t_j \in I_j$ in which all channel sets $S_1, S_2, ...S_{\sigma_j}$ are "full." Since such requests must have been accepted no earlier than T before t_j, we have, $\forall j = 1, 2, ..., l$,

$$A(j-1) + A(j) \geq \frac{\sigma_j c}{H_\mu} - \frac{\sigma_j(\sigma_j + 1)}{2} \geq \frac{\sigma_j c}{2H_\mu}\ , \tag{5}$$

given our assumption on the number of channels. Along similar lines, $A(0) \geq \frac{\sigma_0 c}{2H_\mu}$.

Now consider $R(j)$. By definition, all requests in $R(j)$ were rejected by $\mathcal{H}$ during I_j, so each such request was *made* in the interval $[jT - \nu, (j+1)T - \nu)$. The adversary $\mathcal{A}^*$ could thus serve these requests anytime during the interval $[jT - \nu, (j+1)T)$. Hence the offline can utilize upto $2c$ transmissions to serve these requests (≤ 2 on each channel).

Assume first $\sigma_j < \mu$. Each such transmission by $\mathcal{A}^*$ must contain $\leq \sigma_j$ requests from $R(j)$. For if there existed an offline transmission serving more than σ_j requests from $R(j)$, $\mathcal{H}$ too would be in a position to accept these requests, since its saturation level in that interval was σ_j. Hence we have, $\forall j$,

$$R(j) \leq 2\sigma_j c\ . \tag{6}$$

From (5) and (6) above, we have, $\forall j$,

$$4H_\mu(A(j-1) + A(j)) \geq R(j)\ . \tag{7}$$

The case $\sigma_j = \mu$ is handled similarly, yielding the same result. Let now A be the total number of requests accepted by $\mathcal{H}$, and let R be the total number of requests rejected by $\mathcal{H}$ but accepted by $\mathcal{A}^*$. Note that $A = A(0) + A(1) + ... + A(l)$, and similarly, $R = R(0) + ... + R(l)$. Summing up the $(l+1)$ inequalities of the form (7) (and making the necessary adjustments), we obtain

$$4H_\mu \cdot (2A) \geq R\ ,$$

or

$$R \leq 8H_\mu A\ . \tag{8}$$

Now, from the definitions, $rev(\mathcal{A}^*) \leq A + R$, while $rev(\mathcal{H}) = A$. Hence

$$\phi(\mathcal{H}) = \frac{rev(\mathcal{A}^*)}{rev(\mathcal{H})} \leq 1 + \frac{R}{A} \ . \tag{9}$$

Hence, from (8),

$$\phi(\mathcal{H}) \leq 1 + 8H_\mu \ .$$

□