AN ANALYTIC STUDY OF STREAM TAPPING PROTOCOLS

Jehan-François Pâris

Department of Computer Science
University of Houston, Houston, TX 77204-3010

Darrell D. E. Long

Department of Computer Science
University of California, Santa Cruz, CA 95064

ABSTRACT

We present the first analytic study of stream tapping protocols, a family of protocols that provide the most efficient way to distribute videos on demand at low to medium request arrival rates, say, less than ten requests per hour for a two-hour video. The main results of this study are analytical solutions for the optimal operational points of stream tapping, stream tapping with small client buffers, stream tapping with partial preloading and stream tapping with proactive streams. In addition we introduce a new stream tapping protocol with batching that caps the bandwidth requirements of stream tapping at high to very high arrival rates.

1. INTRODUCTION

Despite its attractiveness as a concept, video-on-demand (VOD) has yet to make a significant impact on the home entertainment market. One of the reasons behind this state of affairs is the high demands that VOD makes on video servers. Videos in MPEG-2 format require the delivery of around 5 Mb/s of data per stream. A video server that allocates a separate stream of data to each request requires 5 Gb/s to accommodate 1,000 concurrent users.

This situation has led to numerous proposals aiming at reducing the bandwidth requirements of VOD services. These proposals can be broadly classified into two groups. Proposals in the first group are said to be *proactive* because they broadcast each video according to a fixed schedule that is not affected by the presence–or absence–of requests for that video. They are also known as *broadcasting* protocols [4, 6, 9, 10].

Other solutions are purely *reactive*: they only transmit data in response to a specific customer request. Unlike proactive protocols, reactive protocols do not consume bandwidth in the absence of customer requests [1−3, 5, 7, 8]. Of all reactive protocols, stream tapping is the one that has spawned the most variants thanks to its good performance and its simplicity.

Given the reactive nature of stream tapping, previous studies of the protocol [1, 2, 8] have relied on discrete simulation to evaluate the bandwidth requirements of the protocol. While these studies produced reliable estimates of the bandwidth requirements of the numerous protocol variants, they did not provide the same wealth of information, as analytical solutions would do. In particular, they offered no help in setting the optimal protocol settings.

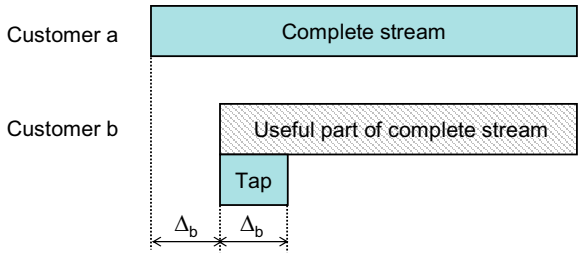

Figure 1: How stream tapping works

We present the first systematic analytical study of the performance of stream tapping with or without partial preloading and introduce a new stream tapping protocol with batching that caps the bandwidth requirements of the protocol at high to very high arrival rates.

2. STREAM TAPPING

Stream tapping [1, 2] requires each customer set-top box to have a buffer capable of storing at least 10 minutes of video data and to be able to receive data at at least twice the video consumption rate. This buffer will allow the set-top box to "tap" into streams of data on the server originally created for other clients, and then store these data until they are needed. In the best case, clients obtain most of their data from an existing stream.

In particular, stream tapping defines two types of streams. *Complete streams* read a video in its entirety and are the most commonly tapped. *Full tap streams* can be used if a complete stream for the same video started $\Delta \leq b$ minutes in the past, where b is the size of the client buffer, measured in minutes of video data. In this case, the client begins receiving the complete stream right away, storing the data in its buffer. Simultaneously, it receives a full tap stream and uses it to display the first Δ minutes of the video. After that, the client will consume directly from its buffer, which will then always contain a moving Δ-minute window of the video.

Clients that can receive data at three times the video consumption rate, they can use an option of the protocol called *extra tapping*. Extra tapping allows clients to tap data from any stream on the VOD server, and not just from complete streams. Figure 1 shows some two sample customer requests. Since customer *a* is the first customer, it is serviced by a complete stream, whose duration is equal to the duration D of the video. Since customer *b* arrives Δ_b

minutes after customer a, it can share $D - \Delta_b$ minutes of the complete stream and only requires a full tap of duration Δ_b minutes.

3. OUR ANALYSIS

We will focus our study of on the performance of *stream tapping* protocols without extra tapping.

3.1. Stream tapping

We consider first stream tapping without extra tapping for a video of duration D that is being accessed at a rate λ. We assume that we will restart a new complete stream whenever the length of the next full tap exceeds βD where $0 < \beta \leq 1$ is a parameter to be determined. In other words, we will wait for an incoming request, start a complete stream of duration D, tap this stream over a time interval of duration βD then restart the process. During this time interval, the server will process an average of $\lambda\beta D$ requests in addition to the request that prompted the complete stream. Hence the average number of requests sharing the same complete stream is

$$n_{avg} = 1 + \lambda\beta D .$$

Since the lengths of the $\lambda\beta D$ full tap streams in the group will be uniformly distributed over the interval $(0, \beta D]$, the average duration of each of these streams will be equal to $\beta D/2$. The total duration of the streams required for processing these $1 + \lambda\beta D$ requests will thus be

$$W = D + (\lambda\beta D)(\beta D/2) = D + \lambda\beta^2 D^2/2$$

and the average request service time T_{avg} will be

$$T_{avg} = \frac{W}{n_{avg}} = \frac{D + \lambda\beta^2 D^2/2}{1 + \lambda\beta D} . \quad (1)$$

Differentiating the above expression with respect to β, we obtain

$$\frac{\lambda D^2(\beta^2\lambda D + 2\beta - 2)}{2(\beta\lambda D + 1)^2},$$

the only positive root of which is

$$\beta_{opt} = \frac{\sqrt{2\lambda D + 1} - 1}{\lambda D} . \quad (2)$$

This solution has the interesting property that $T_{avg}(\beta_{opt}) = \beta_{opt} D$, meaning that the optimal point for restarting a new complete stream is reached when the cost of the new tap stream becomes equal to the average service cost of all previous requests serviced by the current complete stream.

The average bandwidth requirements B_{ST} of the stream tapping protocol are then given by multiplying the average request service time T_{avg} by the request arrival rate λ:

$$B_{ST} = \lambda T_{avg} = \frac{\lambda D + \beta^2(\lambda D)^2/2}{1 + \beta\lambda D} . \quad (3)$$

Fig. 2 displays the bandwidth requirements of the stream tapping protocol for a two-hour video when the request arrival rate varies between one and one thousand requests per hour: the solid line represents the values

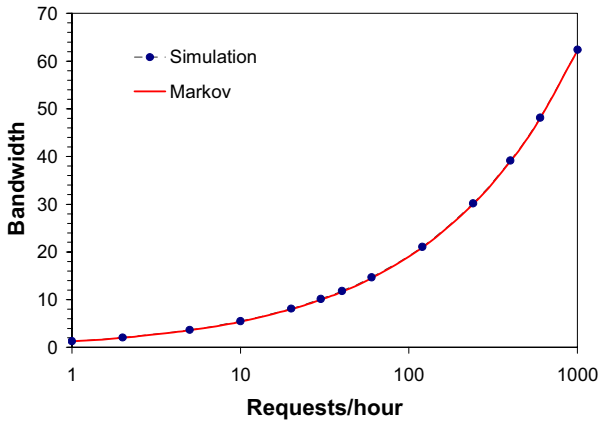

Fig. 2. Bandwidth requirements of the stream tapping protocol for a two-hour video.

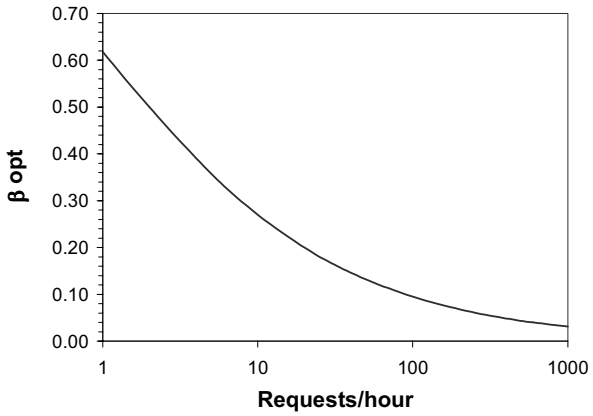

Fig. 3. Fraction of a two-hour video that the client must be able to store in order to achieve the best performance at different request arrival rates.

predicted by Eq. (2) while the small circles correspond to values obtained by simulating one million requests. As we can, the two techniques reach nearly identical results.

In addition, we can now estimate how the client buffer size $\beta_{opt} D$ required to minimize the server bandwidth is affected by the request arrival rate λ. As Fig. 3 shows, β_{opt} is a monotonically deceasing function of λ: a buffer capable of containing 20 percent of the video suffices to achieve optimal performance when λ exceeds 20 requests per hour while lower arrival rates require buffers capable of storing up to 62 percent of the video.

3.2. Stream tapping with a small buffer

We may encounter clients that can only store a fraction β_{max} of each video. In that case, we need to replace β_{opt} in all the previous equations by $\beta_{eff} = \min(\beta_{max}, \beta_{opt})$. This will result in a significant increase of the server bandwidth whenever β_{max}, is much smaller than β_{opt}. Returning to Fig. 3, we see that the impact of this restriction will mostly affect the performance of the protocol at low arrival rates as β_{opt} rapidly decreases when the arrival rate λ increases.

3.3. Stream tapping with partial preloading

Stream tapping with partial preloading [8] requires each client to receive and store in its buffer ahead of time the first few γD minutes of each video with $0 < \gamma < 1$. Hence a complete stream will only have to transmit the remaining $(1 - \gamma)D$ minutes of the video.

As before, we will restart a new complete stream whenever the length of the next full tap will exceed βD where $0 < \beta \leq 1$ is a parameter to be determined. The main difference is that all requests arriving within γD minutes after the request that prompted the complete stream will receive all the data they need from the complete stream. As a result, only the requests arriving more than γ minutes after the first request but less than $\gamma + \beta$ after it will require tap streams. During this time interval, the server will process an average of $\lambda(\beta + \gamma)D$ requests. Hence the average number of requests sharing the same complete stream is now

$$n_{avg} = 1 + \lambda(\beta + \gamma)D .$$

Since the lengths of the corresponding full tap streams will be uniformly distributed over the interval $(0, \beta D)$, the average duration of each of these streams will be $\beta D/2$. The average request service time is

$$T_{avg} = \frac{W}{n_{avg}} = \frac{(1-\gamma)D + \lambda\beta^2 D^2 / 2}{1 + \lambda(\beta + \gamma)D} .$$

and the derivative of that expression with respect to β has a single positive root

$$\beta_{opt} = \frac{\sqrt{\lambda^2\gamma^2 D^2 + 2\lambda D + 1} - \lambda\gamma D - 1}{\lambda D} .$$

From T_{avg}, we can compute the bandwidth requirements B_{STPP} of the stream tapping protocol with partial preloading by multiplying the average service time for a request by the request arrival rate, obtaining

$$B_{STPP} = \lambda T_{avg} . \quad (4)$$

Fig. 4 displays the bandwidth requirements of the stream tapping protocol for a two-hour video when various fractions γ of the video is preloaded. As we can see, partial preloading has most impact at high to very high request arrival rates. For instance, preloading the first six minutes of a two-hour video reduces the server bandwidth by only 13 percent when the video is requested 10 times per hour but would reduce the same bandwidth by at least 33 percent when the request arrival rate exceeds 60 requests per hour.

3.4. Adding proactive streams

Like all purely reactive video distribution protocols, stream tapping performs much better than proactive distribution protocols at low request arrival rates and much worse at very high request arrival rates. As a result, stream tapping is especially vulnerable to flash crowds caused by a large number of customers suddenly deciding they want to watch a specific video, which could overload the server.

Returning to Fig. 3, we notice the very low values of β_{opt} at high arrival rates. Consider, for instance, a server receiving one request per minute for a two-hour video. Its β_{opt} would be close to 0.125, which means that it would

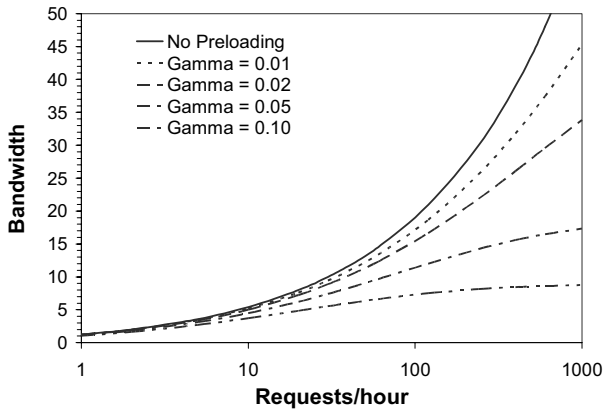

Fig. 4. Bandwidth requirements of the stream tapping protocol for a two-hour video when various fractions γ of the video are preloaded.

restart a complete stream every 15 minutes. Let us assume that (a) customers always watch the video in sequence without fast-forwarding and (b) their set-top box can receive video data at at least three times the video consumption rate. We could then halve the durations of these complete streams by introducing a single proactive channel that would continuously repeat the second half of the video. As a result, a client serviced by a tap stream would simultaneously receive data from three sources, namely, the complete stream it is tapping, its dedicated tap stream and the channel broadcasting the last half of the video. The bandwidth requirements of a stream tapping protocol with one proactive channel broadcasting the last half of the video is then given by

$$B_{ST+1} = 1 + \frac{\lambda D/2 + \beta^2(\lambda D/2)^2/2}{1 + \beta\lambda D/2} .$$

As we have shown previously [7], adding more proactive streams can further reduce the duration of the complete streams. With two proactive streams, we could partition the video into four segments and broadcast three of them on the proactive streams. Adding a third proactive stream would then allow us to partition the video into ten segments and broadcast nine of them on the proactive streams, leaving only the first ten percent of the video to be distributed reactively. Fig. 5 displays the bandwidth requirements of the stream tapping protocol for a two-hour video and one, two or three proactive streams.

3.5. Stream tapping with batching

A simpler solution to this problem is to enforce a maximum request service rate λ_{max} of, say, one or two requests per minute. This would mean that a request arriving less than $1/\lambda_{max}$ minutes after the previous request would be delayed by at most $1/\lambda_{max}$ minutes and that all requests arriving in that interval would be serviced by the same server stream.

Hence, we define the effective request service rate λ_{eff} as the minimum of the request arrival rate λ and the maximum request service rate λ_{max}, that is,

$$\lambda_{eff} = \min(\lambda, \lambda_{max})$$

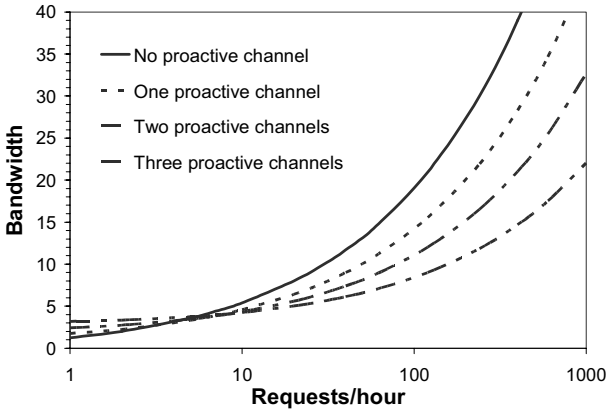

Fig. 5. Bandwidth requirements of the stream tapping protocol for a two-hour video with and without proactive channels.

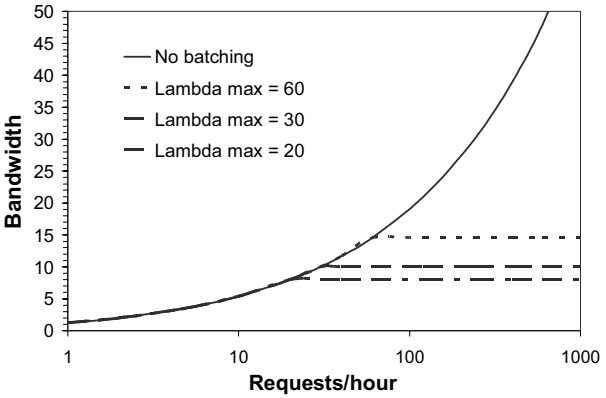

Fig. 6. Bandwidth requirements of the stream tapping protocol for a two-hour video when the request service rate is limited to 20, 30 or 60 requests per hour.

As before, we will restart a new complete stream whenever the length of the next full tap exceeds βD where $0 < \beta \leq 1$ is a parameter to be determined. The main difference is that the $\lambda\beta D$ requests arriving on the average within βD minutes after the complete stream will never result in more than $\lambda_{\max}\beta D$ service streams. In addition, that complete stream will now be likely to serve more than one request whenever $\lambda > \lambda_{max}$. As a result, the average number of requests sharing the same complete stream will now be

$$n_{avg} = \max(1, \lambda / \lambda_{\max}) + \lambda\beta D$$

and the total duration of the streams required for processing these n_{avg} requests will be

$$W = D + \lambda_{eff}\beta D(\beta D/2) = D + \lambda_{eff}\beta^2 D^2/2 .$$

Hence, the average request service time will be

$$T_{avg} = \frac{W}{n_{avg}} = \frac{D + \lambda_{eff}\beta^2 D^2/2}{\max(1, \lambda/\lambda_{\max}) + \lambda\beta D} . \quad (5)$$

When $\lambda \leq \lambda_{max}$, λ_{eff} is equal to λ, max $(1, \lambda/\lambda_{max})$ is equal to one, and Eq. (5) becomes identical to Eq. (1): Since our model only considers average cases, it assumes that batching does not affect the performance of the protocol as long as $\lambda \leq \lambda_{max}$. This not true in reality because actual request interarrival times fluctuate around their average $1/\lambda$ and will occasionally exceed $1/\lambda_{max}$.

When $\lambda \leq \lambda_{max}$, Eq. (5) becomes

$$T_{avg} = \frac{W}{n_{avg}} = \frac{D + \lambda_{\max}\beta^2 D^2/2}{\lambda/\lambda_{\max} + \lambda\beta D} . \quad (6)$$

and the derivative of that expression with respect to β has a single positive root

$$\beta_{opt} = \frac{\sqrt{2\lambda_{\max} D + 1} - 1}{\lambda_{\max} D} . \quad (7)$$

In that range of request arrival rates, the bandwidth requirements B_{STB} of the stream tapping protocol with batching is given by

$$B_{STPP} = \lambda T_{avg} = \frac{\lambda_{\max} D + \beta_{opt}^2 \lambda_{\max}^2 D^2/2}{1 + \beta_{opt}\lambda_{\max} D}$$

which does not depend on λ.

Fig. 6 displays the bandwidth requirements of the stream tapping protocol with batching for a two-hour video and selected values of $\lambda_{\max}$.

4. CONCLUSIONS

We can make three main conclusions from our study of stream tapping protocol. First, the client buffer requirements of the stream tapping protocol rapidly decrease when the request arrival rate increases. For instance, a buffer capable of containing 20 percent of the video suffices to achieve optimal performance when the request arrival rate exceeds 20 requests per hour while lower arrival rates require buffers capable of storing up to 62 percent of the video. Second, partial preloading can effectively reduce the distribution costs of very popular videos but has little or no impact on the distribution costs of less popular videos. Finally, batching requests at high arrival rates can actually cap the bandwidth requirements of the protocol at a reasonable value.

REFERENCES

[1] S. W. Carter and D. D. E. Long. "Improving video-on-demand server efficiency through stream tapping." *Proc. 5th ICCCN Conf.,* pp. 200–207, Sep. 1997.

[2] S. W. Carter and D. D. E. Long. "Improving bandwidth efficiency on video-on-demand servers." *Computer Networks and ISDN Systems*, 30(1–2):99–111, Mar. 1999.

[3] D. Eager and M. K. Vernon. "Dynamic skyscraper broadcast for video-on-demand." *Proc. 4th Int'l Workshop on Advances in Multimedia Information Systems*, pp. 18–32, Sep. 1998.

[4] K. A. Hua A. and S. Sheu. "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. *Proc. ACM SIGCOMM '97 Conf.,* pp. 89–100, Sep. 1997.

[5] K. A. Hua, Y. Cai, and S. Sheu. "Patching: a multicast technique for true video-on-demand services." *Proc. 6th ACM Multimedia Conf.*, pp. 191–200, Sep. 1998.

[6] L. Juhn and L. Tseng. "Harmonic broadcasting for video-on-demand service." *IEEE Trans. on Broadcasting*, 43(3):268–271, Sep. 1997.

[7] J.-F. Pâris, S. W. Carter and D. D. E. Long. "A Reactive Broadcasting Protocol for Video on Demand." *Proc. 2000 MMCN Conf.*, pp. 216–223, Jan. 2000.

[8] J.-F. Pâris. "A Stream Tapping Protocol with Partial Preloading." *Proc. 9th MASCOTS Symp.*, pp. 423–430, Aug. 2001.

[9] J.-F. Pâris. "A simple low-bandwidth broadcasting protocol for video on demand." *Proc. 7th ICCCN Conf.*, pp. 690–697, Oct. 1999.

[10] S. Viswanathan and T. Imielinski. "Metropolitan area video-on-demand service using pyramid broadcasting." *Multimedia Systems*, 4(4):197–208, Aug. 1996.