# RECEIVER-BASED OPTIMIZATION FOR VIDEO DELIVERY OVER WIRELESS LINKS

*Carri Chan†, John Apostolopoulos ‡,Yan Li §, Nick Bambos ¶*

† EE Dept, Stanford University
Stanford, CA, USA
*cwchan@stanford.edu*

‡Hewlett-Packard Laboratories
Palo Alto, CA, USA
*japos@hpl.hp.com*

§Qualcomm
Campbell, CA, USA
*liyan@stanfordalumni.org*

¶ EE Dept, Stanford University
Stanford, CA, USA
*bambos@stanford.edu*

## ABSTRACT

We consider transfer of video frames over a time-varying wireless channel. When the channel is good, the transmitter can send frames at a higher rate than the receiver can consume them via playout. In that case, we introduce the idea of admitting new frames even when the receiver buffer is full, by selectively evicting frames already in the buffer; we can also control the playout rate, so as to optimize the tradeoff between video distortion and the time to freeze when the channel turns bad and frames arrive at a lower rate than should be played out. The decision/control problem of whether to admit a new frame, which already stored one to evict to accommodate the new one, and at what rate to play out frames is formulated within a dynamic programming framework, and an interesting connection to the Knapsack problem is made. Application of the idea in a relevant simple system shows significant performance gains, indicating that it is a promising approach for improving video delivery performance over challenging wireless channels.

## 1. INTRODUCTION

Video delivery over wireless links is a theoretically interesting and practically important problem which has received considerable attention in recent years. From the application layer, the wireless link may often appear as a 'reliable' channel but with time-varying throughput. This time-varying throughput can lead to buffer underflow and the associated highly undesirable video freeze at the receiver, when the throughput drops below the receiver's consumption rate for a duration longer than what can be compensated for by the receiver's buffer. A variety of approaches have been proposed to overcome this problem, including sender-driven rate-distortion (R-D) optimized streaming where the sender intelligently selects which packets to transmit to minimize the distortion at the receiver subject to the available throughput [1], [2]. Another approach is the receiver-based adaptive media playout (AMP), where the playout rate at the receiver is reduced when the buffer begins to underflow in order to reduce the probability of underflow [3], [4]. Joint R-D optimized packet scheduling, and transmitter power control at the sender, as well as (content-aware) adaptive playout at the receiver has been studied in [5].

In this paper, we examine the case of a simple transmitter, and a receiver which has a limited buffer size and simple decoder which only allows the ability to play or freeze. We propose a receiver-based optimization framework for maximizing the time to receiver underflow and associated undesirable video freeze by pre-emptively evicting frames from the receiver buffer in an R-D optimized manner in order to receive and store additional frames which would increase the playout time.

To put the control/decision dilemma into perspective, consider the following scenario. When the channel is good the receiver may be able to transfer packets to the receiver at a much higher rate than the latter can consume them. In that case, after the receiver buffer fills up, the standard approach is to stop the transmitter from sending more packets that would be dropped due to a full receiver buffer. We examine an alternative idea, as follows. When the channel is good, the transmitter may selectively keep sending frames/packets even when the receiver buffer is full. The receiver may then elect to keep a received frame and evict one already in its full buffer to keep its newly admitted one. The eviction/admission decision is done with an eye towards minimizing the distortion and maximizing the time to freeze when in a subsequent phase the channel turns bad and the receiver buffer drains as new frames/packets arrive sparingly. For example, a 'naive' way to evict existing frames in a full buffer to accommodate new ones is to substitute every other existing frame with a newly arrived one and play each frame twice (hence, reduce the playout rate to half the normal one). A proper way to formulate the decision/control dilemma of whether to admit a new frame, which stored one to evict to accommodate it, and at what rate to play out frames in the buffer is within a dynamic programming framework which can lead to a systematic way for obtaining the optimal control and deriving justified practical heuristics. An interesting connection to the Knapsack problem can also be established, which could find applications to other video transmission optimization scenarios.

This paper continues in Section 2 by describing the general model and problem formulation, followed by presenting the optimal solution achievable by a dynamic programming formulation. A special case of the general framework is then presented, for the case of H.264 coded video, which demonstrates the tradeoff between distortion and freezing.

## 2. GENERAL MODEL AND PROBLEM FORMULATION

In this section we present a novel modeling framework for receiver based optimization and provide the optimality formulation. We embed the problem within a Markov decision process framework and use dynamic programming to compute the optimal control.

### 2.1. System Description

We consider a system shown in Fig. 1, which is comprised of a transmitter (Tx) and a receiver (Rx) communicating over a wireless link. Time is slotted and is indexed by $t = 0, 1, 2, 3, ....$ We assume either the entire video sequence of $N$ frames is pre-stored at the transmitter or arrives to the transmitter through a backbone access network. For the latter case, we assume the backbone access network is not the bottleneck. Each frame has size $s_n$ in bits. The receiver is equipped with a buffer of size $B$, where received packets are queued up while waiting to be played out.

The condition of a wireless channel varies with time. This is typically due to both the attenuations in the received signal strength (path-

This work was conducted when the third author was affiliated with the Electrical Engineering Dept. at Stanford University.

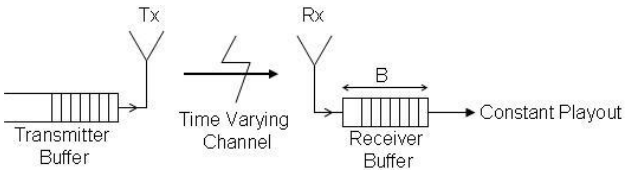


**Fig. 1**. Receiver based Optimization

loss, shadowing and fast fading), the varying interference levels caused by other users, and the mobility of nodes. Without loss of generality, we model the wireless channel as a Markov chain with transition probabilities $q_{ij}$; in channel state $i$, the bandwidth available to the video stream is $R_i$. The reasoning behind this model is the following. Fast fading, slow shadowing, path loss and interferences all affect the signal to interference/noise ratio (SIR); in turn, the SIR dictates the physical transmission rate and packet error rate, thus the channel throughput. Assuming that the physical and MAC layers use coding and retransmissions to combat channel variations, the wireless channel appears to the application layer as error-free but with time-varying throughput. The throughput $R_i$ in state $i$ can be calculated as $R_i^p(1-PER)$, where the $R_i^p$ is the physical channel rate after the coding, and $PER$ is the packet error rate with the corresponding codes; this is reasonable if the channel varies slower than a packet's transmission time, which is the case for low mobility or in-home environments.

We assume that the transmitter has no knowledge of the content of the video frames. This is common in many practical scenarios. For example, the stream may be encrypted or the transmitter has no capability to access the MPEG/H.264 packet headers. At each time slot, the transmitter transmits as many video frames as the channel throughput allows. We assume at the end of each time slot, the receiver sends the reception acknowledgements to the transmitter, indicating the results of the transmission. We further assume these acknowledgements are small in size, do not consume any channel throughput, and are received with no errors.

At each time slot, the receiver plays out one frame stored in the buffer. If the frame to be played out, say frame $n$, is not available, the transmitter will use error concealment techniques to estimate the missing frame. One such technique is to reuse the previous frame. Due to the absence of frame $n$, the video quality suffers distortion $d_n$. We further assume distortion values are additive across multiple missing frames. This is a reasonable assumption if the losses are sparse. If the receiver buffer runs empty, the receiver will terminate the playout and proceed to the rebuffering process. We denote the time at which the receiver buffer runs empty before all $N$ video frames have been played as the time to freeze $T_f$. The viewer would like to view the entire video stream without any playout freeze. However, given the limited buffer space and time-varying channel throughput, playout freeze may be unavoidable. One solution is to implement receiver buffer control as mentioned earlier.

In this paper we propose to allow the receiver to drop frames that were previously stored in the buffer and replace them with newly arrived frames. Combined with error concealment techniques, this control extends the actual playout duration of the frames in the buffer. For example, suppose the buffer can store 6 frames. We can either store frames 1 to 6, offering a playout duration of 6 frame periods. Or we can store frames 1, 3, 5,..., 11, and use error concealment techniques to recover the missing frames. This effectively offers playout duration of 12 frames, thus reducing the risk of playout buffer underflow. However the video quality suffers in the latter case. Clearly there exists a trade-off between the distortion incurred due to frame loss and the reduction in buffer underflow risk. The latter can be measured as time to freeze $T_f$. This trade-off can be explored by implementing the receiver buffer control.

In the case that new frames arrive and the buffer has enough space to store them, the decision is simple and the new frames are added to the buffer. However, if the buffer is full, the receiver must make a decision: ignore the incoming frames and wait for them to be retransmitted later or evict some frame(s) already in the buffer in order to allow for the addition of the incoming frames. Clearly, eviction incurs distortion in video quality. However, eviction also extends the playout time of the video frames stored in the buffer. Given a limited buffer size at the receiver, this may be necessary to maximize the time to freeze. If the channel enters a deep fade, connectivity will be lost, the buffer will eventually be depleted and the playout will freeze. By extending the playout time of the frames stored in the buffer, the receiver extends the length of time of a deep fade it can avoid freezing over.

On one hand the receiver wants to playout the video stream with the least amount of distortion, which means playing every single received frame. On the other hand, the receiver wants to maximize the time to freeze, which means evicting frames to extend the buffer's playout time. We seek the optimal buffering policy to jointly optimize these goals.

### 2.2. Discussion of Costs

In order to explore the trade-off described in Section 2.1, we define the performance cost functions here.

*Distortion Costs.* The distortion cost $d_n$ is incurred if frame $n$ is dropped by the receiver. Assuming additive distortion, a minimum distortion of 0 corresponds to all frames $1, 2, 3, ..., N$ being played out and a maximum distortion of $\sum_{n\in(1,N)} d_n$ corresponds to all frames being dropped. The distortion associated with each frame is transmitted to the receiver, e.g., as part of the packet header as described in [6].

*Time to Freeze.* We define $T_f$ as the time after the start of playout at which the receiver buffer hits empty and the playout must freeze. Then at time slot $T_f$, a cost $\Phi(T_f)$ is included to reflect the desire to maximize the time to freeze. $\Phi(T_f)$ is nonnegative function that is decreasing in $T_f$. Recalling that $N$ is the number of video frames to be transmitted, if $T_f = N$, then the buffer empties after all video frame slots have passed and $\Phi(N) = 0$. Beyond these two general properties, we do not assume any specific structure for the function $\Phi(T_f)$, which can be adapted depending on the system model.

### 2.3. System State and Optimal Control

The objective is to receive and playout video frames while minimizing the overall cost incurred in the process. The system state to be tracked in each time slot is:

$$(i, m, \mathbf{p}) \tag{1}$$

where, $i$ is the channel state which specifies how many frames can be transmitted in the time slot, $m$ is the index of the next video frame to be transmitted, and $\mathbf{p} \in \mathbf{P}$ is an indicator vector of which video frames are stored in the receiver buffer. $\mathbf{p}_n = 1$ if frame $n$ is in the buffer and $\mathbf{p}_n = 0$ otherwise.

At each time slot, the receiver plays out and removes the played frame from the buffer. The control applied in each time slot is $\tilde{\mathbf{p}}$–the new buffer state. If no frame is played out and no new frames are added to the buffer, either because the transmitter did not send any frames or because the receiver chooses to ignore the transmitted frames, then $\tilde{\mathbf{p}} = \mathbf{p}$. In the other case, $\tilde{\mathbf{p}} = \mathbf{p}$ except for the following elements: If frame $n$ was played out, then $\tilde{\mathbf{p}}_n = 0 \neq \mathbf{p}_n$. If $\Psi$ is the set of frames that are evicted from the buffer, $\tilde{\mathbf{p}}_n = 0, \forall n \in \Psi$. Likewise, if $\Omega$ is the set of frames that are added to the buffer, $\tilde{\mathbf{p}}_n = 1, \forall n \in \Omega$.

Given this formulation, the system simply becomes a controlled Markov chain, and hence, we can develop a Dynamic Programming

recursion to compute the optimal control. Let $J^t(i,m,\mathbf{p})$ be the cost-to-go, that is the minimum expected cost incurred until termination, given that the optimal control is used, $t$ video frame time slots have past and the current state is $(i,m,\mathbf{p})$. $J^t(i,m,\mathbf{p})$ satisfies the following functional recursive equations, with $t \in \{1,2,3,...,N-1\}$, $i \in \mathcal{I}$, $\mathbf{p} \in \mathbf{P} \setminus \mathbf{0}$ (does not hold when the buffer is empty):

$$\begin{aligned} J^t(i,m,\mathbf{p}) = d_t\mathbf{p}_t + \mathbf{p}_t \min\{&\sum_j q_{ij} J^{t+1}(j,m,\mathbf{p}^{t^-}), \\ &\min_{n\le i, \tilde{\mathbf{p}}\in\tilde{\mathbf{P}}} \sum_j q_{ij} J^{t+1}(j,m+n,\tilde{\mathbf{p}})\}+ \\ &(1-\mathbf{p}_t)\min\{\sum_j q_{ij} J^{t+1}(j,m,\mathbf{p}), \\ &\min_{n\le i, \tilde{\mathbf{p}}\in\tilde{\mathbf{P}}} \sum_j q_{ij} J^{t+1}(j,m+n,\tilde{\mathbf{p}})\} \end{aligned} \tag{2}$$

where $\tilde{\mathbf{P}}$ is the set of possible buffer states given the previous buffer state $\mathbf{P}$, the playout of frame $t$, and the arriving frames $m, m+1, ..., m+i$. $\mathbf{p}^{t^-}$ denotes the buffer state minus the $t^{th}$ frame.

The first minimization corresponds to when frame $t$ is included in the buffer and is first played out. Then the decision is to either ignore the incoming frames and the system evolves to state $J^{t+1}(j,m,\tilde{\mathbf{p}})$ or to include a subset, of size $n$, of the arriving frames, $[m, m+i]$, after evicting some existing frames making the system evolve to state $J^{t+1}(j,m+n,\tilde{\mathbf{p}})$. Likewise, the second term which is preceded by $(1-\mathbf{p}_t)$ corresponds to when frame $t$ is not in the buffer for playout. The decisions are analogous to when it is included.

We must also consider the boundary conditions.

$$\begin{aligned} J^t(i,N,\mathbf{p}) = \sum_j q_{ij}\{&\mathbf{p}_t J^{t+1}(j,N,\mathbf{p}^{t^-})+ \\ &(1-\mathbf{p}_t)J^{t+1}(j,N,\mathbf{p})\} \end{aligned} \tag{3}$$

$$J^t(i,m,\mathbf{p}=\mathbf{0}) = \Phi(t) \tag{4}$$

It can be shown that given the current buffer state and the incoming video frames, the optimal eviction policy is given by the 0/1 Knapsack problem [7]. Given the subset, $\mathfrak{F}$, of data frames, we must choose which frames to put into the limited capacity buffer and which not to choose. The knapsack problem will give the optimal solution of which frames should be included in the buffer to minimize the distortion of the subset $\mathfrak{F}$. Minimizing this distortion corresponds to minimizing the playout distortion. This makes sense because if a frame will not fit in the buffer, it will never be played out and its distortion will be incurred at playout. By using the knapsack solution to optimize the contents of the buffer at each time slot, the playout will also be optimized. For a reminder, the knapsack solution is shown. We number the available frames from $1,2,...,M$. Let $A_b(i)$ be the optimal solution given a buffer with capacity $b$ and we only allow inclusion of frames $1,2,...,i$. If the knapsack size is $B$, the optimal solution is given by $A_B(M)$. Then:

$$A_b(i) = \begin{cases} 0, i=0 \text{ or } b=0; \\ \min\{A_{b-s_i}(i-1)+d_i, A_b(i-1)\}, \text{otherwise.} \end{cases} \tag{5}$$

Let $K_B(i,m,\mathbf{p})$ be the optimal knapsack solution given the receiver buffer state $\mathbf{p}$ and the arriving frames given by $m$ and $i$. Then the DP recursion from Eqn. 2 becomes:

$$\begin{aligned} J^t(i,m,\mathbf{p}) = d_t\mathbf{p}_t + \mathbf{p}_t \min\{&\sum_j q_{ij} J^{t+1}(j,m,\mathbf{p}^{t^-}), \\ &\min_{n\le i} \sum_j q_{ij} J^{t+1}(j,m+n,K_B(i,m,\mathbf{p}^{t^-}))\}+ \\ &(1-\mathbf{p}_t)\min\{\sum_j q_{ij} J^{t+1}(j,m,\mathbf{p}), \\ &\min_{n\le i} \sum_j q_{ij} J^{t+1}(j,m+n,K_B(i,m,\mathbf{p}))\} \end{aligned} \tag{6}$$

### 2.4. Special Case

We look at a special case of our general framework, which allows us to highlight fundamental tradeoffs and properties of the general problem. In particular, in order to focus on the relationship between distortion and time to freeze, we consider the following problem setup. We assume that all frames have the same size: $s_n = k$ for all $n$. In this case, the 0/1 knapsack solution is equivalent to the greedy solution. With this assumption, our eviction policy can be reduced to the eviction of frames with the smallest distortion cost. We also assume that all frames transmitted at each time slot are acknowledged at the end of the time slot via an error-free control channel.

To limit the search space we constrain the dropping pattern so that we never drop 2 consecutive frames. We measure distortion in the mean-squared error sense. Each frame has a unique distortion value which is calculated by incrementally dropping frames, decoding the modified video stream and comparing it to the original video content. Due to the limited dropping patterns, there may be instances when newly arriving frames cannot be admitted into the buffer. In this case, the channel resources are wasted. To fully utilize resources, we implement a DP-based heuristic. When the channel is good, the next frame to be played out is missing, and the receiver cannot evict frames to admit newly arriving ones, the receiver will request the missing frame from the transmitter. Otherwise, it will follow the optimal solution given by the DP. This addition will decrease the total number of dropped frames.

In order to closely examine the tradeoff between distortion and time to freeze, we include a weight $W$ on the terminal cost $J^t(i,m,\mathbf{p}=\mathbf{0}) = W\Phi(t)$. By increasing $W$, we place more value on extending the playout time and allow for more distortion to be incurred.

Given these constraints, the optimality equation (2) can be dramatically simplified. Due to the limited space, we neglect it here.

For these simulations we have a receiver buffer size of 10 frames and a total of 200 media frames to be transmitted. The foreman video sequence is pre-encoded using H.264/MPEG-4 AVC with a single leading I frame followed by 199 P frames. Missing frames are estimated using previous frame error concealment. Our time-varying channel alternates between allowing 2 or 0 frames to be transmitted in each time slot. When the channel is in the good state, the transmission rate is twice the playout rate; hence, the system can quickly fill-up the buffer when the channel is in a good state. The steady-state probabilities of being in each channel state is $\frac{1}{6}$ for 0 frames and $\frac{5}{6}$ for 2 frames. The expected duration of being in the bad and good states is 4 and 20 frames, respectively. In this experiment we examine the performance of the optimal policy, computed over 1000 realizations of the channel, as we vary W to examine the range of tradeoffs between incurred distortion and time to first freeze.

Fig. 2 shows the tradeoff between distortion and extending playout time. Observe that as the time to buffer underflow increases, distortion is increased. Evictions must take place in order to extend playout time, but evictions also lead to incurred distortion due to the missing frames. If no control is used the expected time to playout freeze is approxi-

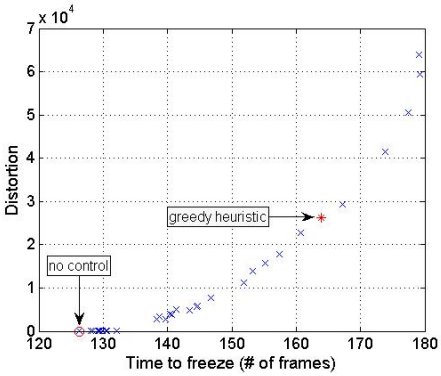


**Fig. 2**. Distortion vs. Time to Freeze tradeoff

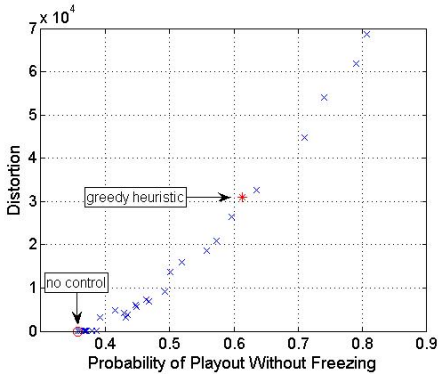


**Fig. 3**. Distortion vs. Probability of not freezing during playout.

mately 125 frames, as identified by the circle. By intelligently evicting frames the time to freeze can be significantly increased, at the cost of incurred distortion from the missing frames.

We also looked at a greedy heuristic to mimic the performance of the DP algorithm. At each time step, the receiver tries to play out the next frame. If it is available and the channel is good we will admit new frames (and evict others) only if the admitted frames have higher associated distortion than the frames to be evicted from the buffer. If the next frame is not available and the channel is good, we request the missing frame as well as the next frame that would normally be transmitted. If it is not available and the channel is bad we incur distortion. Note that unlike the DP-based algorithm, there is no limit on the eviction patterns. The operating point of this policy are denoted by an asterisk. This policy clearly outperforms the no control policy and performs nearly as well as the DP-based algorithm. It may seem that this greedy policy should have the longest time to freeze and lowest probability of freezing as it is frequently admitting and evicting frames. Despite the limits on eviction in the DP-based algorithm, it can extend the playout time even more than the greedy algorithm which suggests that a DP-based algorithm with no limits will easily outperform this algorithm.

In the next set of experiments, rather than terminating service upon underflow, the receiver rebuffers until it is full. In this scenario, the receiver will playout until the very last media frame has been transmitted and reaches the front of the receiver queue. Instead of trading off between distortion and time to first underflow, the tradeoff is now between distortion and the probability of playing out the entire sequence without freezing. Note that the receiver policy that is implemented is given by the DP solution solved by Eqn. 2. This is not the optimal policy for this scenario as we would have to introduce costs to represent the frequency of underflow into the original DP formulation. However, using our solution is a reasonable heuristic to explore these tradeoffs.

Fig. 3 shows the tradeoff between distortion and the probability of not freezing. If we allow for high probability of freezing during playout, MSE distortion can be diminished. Freezing is also a form of distortion and certain viewers may prefer MSE distortion to increase in order to avoid playout freezes. We can see that the policy given by no control, as shown by the circle, has no incurred MSE distortion but is very likely to freeze. Depending on the relative importance between MSE distortion and freezing distortion, the receiver can optimize its policy to maximize the viewer's quality of service despite the time-varying wireless channel. In fact, we can increase the probability of not freezing from $35\%$ with no control, to $80\%$ by intelligently dropping only $7\%$ of video frames.

By allowing for distortion to be incurred during video playout, playout time can be extended and freezing can be minimized. 200 video frames corresponds to just over $6.5$ seconds of video playout. So even with this challenging wireless channel and small buffer at the client, the proposed algorithm allows us to intelligently tradeoff distortion to increase the probability of playing out the entire video stream without freezing, and to perform this tradeoff in an optimized manner. This is in contrast to the no control case, where the probability of freezing is about $65\%$.

The aforementioned simple and special case of the general framework, using H.264 coded video, illustrates the tradeoff between distortion and freezing which can be achieved by receiver-based optimization.

## 3. CONCLUSION

This paper proposed a receiver-based technique for maximizing the time to receiver underflow and undesirable video freeze by pre-emptively evicting frames from the receiver buffer in an R-D optimized manner. This leads to a tradeoff between reduced probability of freezing versus incurred distortion that results from the evicted frames. The optimal policies to achieve this tradeoff are determined using a dynamic programming formulation which solves a knapsack problem at the receiver at each time step. Determining the optimal solution is complex because of the large state space, and therefore this submission includes illustrative examples of the optimal performance for relatively small receiver buffer sizes. Future directions may include the design of justified low-complexity heuristics for achieving high-quality (though sub-optimal) performance for arbitrary receiver buffer sizes.

## 4. REFERENCES

[1] P. Chou, Z. Miao," Rate-distortion optimized streaming of packetized media," submitted to *IEEE Trans. on Multimedia*, Feb. 2001.

[2] B. Girod, J. Chakareski, M. Kalman, Y.J. Liang, E. Setton, R. Zhang, "Advances in Network-adaptive Video Streaming", in *Proc. of IWDC 2002*, Sept. 2002, Capri, Italy.

[3] M. Kalman, E. Steinbach, B. Girod, "Adaptive Media Playout for Low Delay Video Streaming over Error-Prone Channels," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 841 - 851, June 2004.

[4] M. Kalman, E. Steinbach, and B. Girod, "Rate-distortion optimized video streaming with adaptive playout," in *IEEE ICIP*, vol. 3, pp. 189-192, Sept. 2002.

[5] Y. Li, A. Markopoulou, J. Apostolopoulos, Nicholas Bambos. "Joint Packet Scheduling and Content-Aware Playout Control for Video Streaming over Wireless Links", in *IEEE MMSP*, Oct. 2005.

[6] J. Apostolopoulos, "Secure Media Streaming & Secure Adaptation for Non-scalable Video", in *IEEE ICIP*, Oct. 2004.

[7] T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduction to Algorithms*. The MIT Press. Cambridge, MA. 2001.