Disruption-Tolerant Content-Aware Video Streaming

Tiecheng Liu and Srihari Nelakuditi
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC, 29208
{tiecheng,srihari}@cse.sc.edu

ABSTRACT

Communication between a pair of nodes in the network may get disrupted due to failures of links/nodes resulting in zero effective bandwidth between them during the recovery period. It has been observed that such disruptions are not too uncommon and may last from tens of seconds to minutes. Even an occasional such disruption can drastically degrade the viewing experience of a participant in a video streaming session particularly when a sequence of frames central to the story are lost during the disruption. The conventional approach of prefetching video frames and patching lost ones with retransmissions is not always viable when disruptions are localized and experienced only by a few among many receivers. Error spreading approaches that distribute the losses across the video work well only when the disruptions are quite short. As a better alternative, we propose a disruption-tolerant content-aware video streaming approach that combines the techniques of content summarization and error spreading to enhance viewers experience even when the disruptions are long. We introduce the notion of "substitutable content summary frames" and provide a method to select these frames and also their transmission order to mitigate the impact of a disruption. In the event of a disruption, the already received summary frames are played by the client during disruption and near normal playback is resumed after the disruption. We evaluate our approach and demonstrate that it provides acceptable viewing experience with minimal startup latency and client buffer.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data Sharing*

General Terms

Algorithms

Keywords

Network disruption, Video streaming, Video key frames, Disruption-tolerant streaming

1. INTRODUCTION

A major objective in the design and operation of a network is to ensure that it is highly available and reliable. Unfortunately, *failures* occur relatively frequently due to various causes such as interface faults, router crashes and reboots, periodic maintenance, and accidental fiber cuts [8]. Besides these, wireless links are prone to failures due to external interference, channel fading, inclement weather etc. In addition, mobility induces failures in the case of mobile networks. Various routing protocols are designed to react to such failures and route around the failed links and nodes. However, recomputation or rediscovery of new routes, and resumption of forwarding after a route failure may take tens of seconds to minutes [1]. During this period, some destinations could not be reached and the packets to those nodes would be dropped. We refer to this duration of discontinuity in the communication between a pair of nodes in the network as *disruption*. While such disruptions are bad for any application, their impact is particularly severe on continuous media applications. Even an occasional disruption can drastically degrade the viewing experience of a participant in a video streaming session particularly when a sequence of frames central to the story are lost during the disruption. Therefore, it is imperative that we find ways to mitigate the impact of inevitable disruptions on the quality perceived by video streaming clients.

Several approaches have been proposed [7] to make video streaming error-resilient. Most of these approaches can recover from only a few bit errors or packet losses not from long disruptions. Scalable video coding [11] and adaptive streaming schemes [9] adjust video quality to match available bandwidth and/or loss rate of packets. But these schemes may not be able to cope with disruptions where the bandwidth suddenly drops to zero. Moreover, most of these schemes are not content-aware and therefore may skip the key frames as part of the adaptation.

The straightforward approach [3] of prefetching video frames and patching lost ones with retransmissions is restricted. Suppose prefetching is applied to deal with a potential disruption of one minute. Then it would introduce a startup latency in the order of a minute and also require a buffer at the client large enough to hold around 1800 video frames (assuming 30fps) which may not be acceptable considering the variety of video clients, especially the resource-constrained mobile devices. In addition, disruption is likely to be localized only to a few receivers among many receivers in a multicast session in which case retransmission may waste bandwidth or selective retransmission may complicate the protocol. Although there are more intelligent video prefetching and buffering techniques [2, 4] using priority queues or priority drop, how to select frames of different priorities at semantic level needs to be considered.

Error spreading technique [10] does not intend to correct errors, but instead to spread the error to the rest of video sequence. It reduces the loss of continuous frames by permuting frames so that the loss of continuous frames during video transmission results in the loss of isolated frames across the whole video sequence. This method suffers the same drawback of excessive initial delay and video buffer requirement. Since error spreading method spreads the error uniformly to the future video sequence, it also affects video quality significantly long after the end of disruption.

We propose a disruption-tolerant content-aware video streaming approach based on video content summarization. Our approach is based on the notion of "substitutable content summary frames", i.e., the set of frames that provide the visual summary of video content but that are substitutable in content by the frames around them. In this paper, we provide a method to select such summary frames and also their transmission order to mitigate the impact of a disruption. The proposed approach is ideally suited for scenarios where recovery of lost frames is not feasible. Essentially our approach provides a better tradeoff between the additional resources (time/bandwidth/buffer) consumed to deal with the disruption and the corresponding improvement in the perceived quality at the client.

2. DISRUPTION-TOLERANT STREAMING

The essential step in disruption-tolerant video transmission is to construct a disruption-tolerant video sequence so that in the event of disruption, it still provides helpful video content to clients.

2.1 Substitutable Content Summary Frames

In this paper, we provide a new notion of "substitutable content summary frames", for abbreviation, "summary frames". Similar to key frames selected by general video summarization approach [5], summary frames summarize the visual content of a video. But unlike key frames, they are not indispensable in term of understanding video content. The visual content of these summary frames can be substituted by that of non-summary frames. By displaying all the summary frames, a user perceives video content "snapshots", which aid the understanding of video content; by displaying all the non-summary frames, a user still gets a video sequence with quality very close to the original complete sequence.

For a video sequence S, let F be the set of all frames in the video. The target is to divide F into two sets: F_s, the set of summary frames, and ${F_s}^c$, the set of complementary frames (non-summary frames). F_s provides a visual summary of the video content and the content of frames in F_s can be replaced by a subset of F_s^c.

Let $d_{i,j}$ represent the content difference between frame f_i and f_j. In this paper, we define content difference as color histogram distance based on L^1 norm. However, this definition may be extended to other content differences for different video genres [6]. Define r as the ratio of the number of all frames to the number of summary frames. For a given summary ratio r, the algorithm of selecting summary frames is as follows.

1. Initialize summary frame set and complementary frame set: $F_s = F$, $F_s^c = \emptyset$.
2. Build a doubly linked list of all frames. Two frames are adjacent in the list only when they are temporally adjacent in the video sequence.
3. Measure content difference of all adjacent frames in the list.
4. Find the minimum content difference and delete one frame. Suppose $d_{j,k}$ is the minimum of all adjacent frame differences, f_j is adjacent to $\{f_p, f_k\}$, f_k is adjacent to $\{f_j, f_q\}$, we make a decision to drop frame: if $d_{p,j} < d_{k,q}$, we delete frame f_j form the linked list; otherwise we delete frame f_k.
5. Update F_s and F_s^c. Suppose f_j is the frame deleted in step 4, then $F_s = F_s \setminus \{f_j\}$, $F_s^c = F_s^c \cup \{f_j\}$. Since f_p and f_k become adjacent in the linked list, we measure their content difference.
6. Repeat step 4 and step 5 until the number of summary frames reaches N/r.

The process above is a greedy algorithm in nature. For a video of N frames, the memory usage is $O(N)$ and computational complexity is $O(N \log N)$.

Temporally adjacent summary frames are quite dissimilar in visual content, otherwise the content difference between them is very small, making one frame eligible to be removed from the set of summary frames in the process above. Thus our greedy approach reduces visual content redundancy of summary frames.

The selected summary frames are also content "substitutable". For each summary frame, according to our selection process, there exists one complementary frame with a relatively small content difference to the summary frame. At a certain iteration, the content difference between the summary frame and an adjacent frame is the smallest among all, making that frame be deleted from the list of summary frames. That complementary frame makes a good candidate in substituting the summary frame.

The summary ratio r determines how densely the summary frames are spaced. If r is too large, the summary frames are too sparse to cover enough visual content. If r is too small, the disruption-tolerant streaming introduces long startup latency and requires too much client buffer, as we will discuss in the next section. An appropriate value of r is in the range of $10^1 \sim 10^2$, based on our experimental observation. Within this range, r is determined by the size of video client buffer and the expected disruption time.

2.2 Disruption-tolerant Video Sequence

After a video sequence is divided into summary frame set F_s and complementary frame set F_s^c, a new frame sequence S^* is generated by combining the frames in F_s and F_s^c and re-arranging their orders.

Suppose S_s is the sequence of summary frames and S_c is the sequence of complementary frames. $S_s = \{f_{s_1}, f_{s_2}, \cdots, f_{s_{N/r}}\}$, $s_1 < s_2 < \cdots < s_{N/r}$. $S_c =$

$\{f_{c_1}, f_{c_2}, \cdots, f_{c_{N-N/r}}\}$, $c_1 < c_2 < \cdots < c_{N-N/r}$. In this paper, the subscription of f represents the its order in the original video sequence. For example, f_{s_i} is the s_i-th frame in the original video sequence.

We set two pointers p_s and p_c, pointing to first elements of S_s and S_s^c respectively. Suppose normal frame rate is λ. To mitigate a disruption of time T, we generate a new frame sequence S^* using the following method.

1. Initialize S^* as an empty sequence. Set $p_s = s_1$ and $p_c = c_1$.
2. If $p_s \leq T\lambda$, append frame f_{p_s} to the end of the sequence S^*. Move pointer p_s to the next summary frame in S_s.
3. Repeat step 2 until $p_s > T\lambda$
4. Append frame f_{p_c} to the end of sequence S^*, move pointer p_c to the next frame in S_s^c. If p_c reaches the end of S_s^c, terminate the process.
5. If $p_s \leq p_c + T\lambda$, append frame f_{p_s} to the end of the sequence S^* and move pointer p_s to the next frame number in S_s. Otherwise go back to step 4.

As easily seen from the process above, all frames are inserted into S^* without duplication. This process is linear in terms of the number of frames.

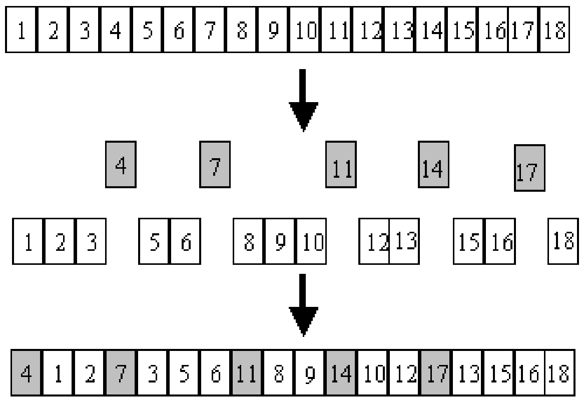

Figure 1: Illustration of the process of selecting content summary frames and constructing a disruption-tolerant sequence. The summary frames are represented using shaded rectangles. The bottom sequence shows the new disruption-tolerant video sequence, where $T\lambda = 4$.

Figure 1 illustrates the process of selecting summary frames and constructing disruption-tolerant video sequence S^*. In S^*, summary frames are sandwiched between complementary frames, and the positions of summary frames are always "ahead" of their positions in the original video sequence. When disruption occurs, the already transmitted summary frames provide visual summary for future video content. Because summary frames are "substitutable" in video content, the loss of summary frames during disruption has little impact on video quality after network resumes.

For the first frame to be displayed at client side, an average of $T\lambda/r$ summary frames need to be transferred first, resulting in a startup latency of time T/r. To properly display the video at client side, original video sequence needs to be reconstructed. At anytime the client receives a complementary frame, it has already received an average of $T\lambda/r$ summary frames that are going to be used to reconstruct video sequence in the future. So the client needs a video buffer of size $T\lambda/r$ to store the summary frames. In the event of disruption, the summary frames in video buffer provide a summary video content for a duration of T.

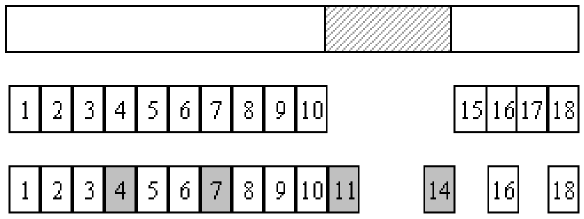

Figure 2: Illustration of the effect of disruption on unchanged video sequence and on disruption-tolerant sequence. The shaded region at top shows a network disruption of 4 frames. For unchanged video sequence, the disruption causes a content-gap of 4 frames, while for our disruption-tolerant video sequence, summary frames provide visual content during network disruption.

Figure 2 shows an example of how our disruption-tolerant video transmission behaves in the event of network disruption. A disruption occurs immediately after the video client receives frame f_{10} and lasts for a time of transferring 4 frames. The disruption causes the original sequence a loss of 4 continuous frames during disruption. In our disruption-tolerant video transmission, at the time of disruption, the client already received and buffered summary frames f_{11} and f_{14}. These two summary frames are displayed to provide useful video content during disruption.

3. ANALYSIS OF PERFORMANCE

We now analyze the performance of our approach in terms of initial delay, video buffer, and post-disruption effect.

3.1 Initial Delay and Video Buffer

Let us assume that network bandwidth is equal to video playback data rate. To overcome a disruption for a duration of T, our approach requires an initial delay of T/r and a client buffer to hold $T\lambda/r$ frames. Prefetching technique incurs an initial delay of T and requires a client buffer of $T\lambda$ frames. Since disruption can last for a relatively long duration and some mobile clients have very limited buffer, the initial delay and client buffer requirements of prefetching are unacceptable in some cases.

Error spreading technique [10] permutes the frame order to spread errors. To reduce error from a loss of 3 continuous frames to 1 frame loss, error spreading method changes a video sequence of $1-2-3-4-5-6-7-8$ to $1-4-7-2-5-8-3-6$. So it requires the same amount of client buffer as in prefetching technique, and incurs an initial delay of time T. Compared with these two techniques, our approach reduces initial delay and client buffer by a factor of r, which would be in the order of tens.

3.2 Post-disruption Effect

In the event of network disruption, some content sent out by server is permanently lost. If normal transmission scheme

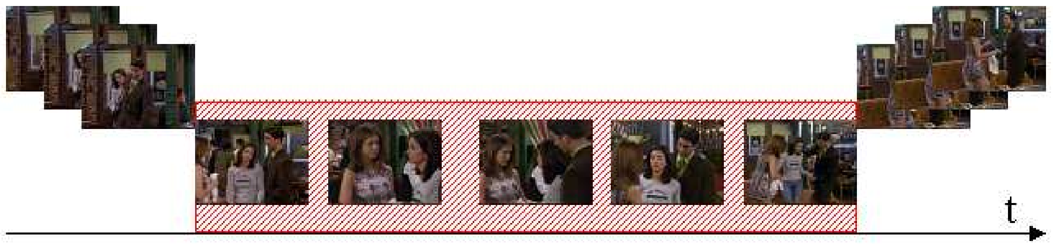

Figure 3: Experimental result of our disruption-tolerant video streaming in the event of a network disruption of 15 seconds. The shaded region shows the duration of network disruption. Our disruption-tolerant streaming displays 5 summary frames during the 15 second disruption.

or prefetching technique is applied, immediately after network recovers, the client can receive and display a video sequence of the same quality. For our proposed method and error spreading [10] technique, the lost frames affect video quality even when network resumes because during network disruption, the lost video frames may also include those intended to be used after disruption. The loss of "future" frames during disruption affects post-disruption video quality, a phenomenon we refer as "post-disruption effect".

In our disruption-tolerant video sequence, an interruption of time T can only affect the quality of video sequence for less than T time after disruption. Since only summary frames are displayed during disruption, the average frame rate during disruption is λ/r. After disruption, because all the frames except some summary frames are displayed, the average frame rate is $(1 - 1/r)\lambda$. Considering summary ratio r usually is a large number, this rate is very close to normal frame rate λ. Moveover, because our selected summary frames are "substitutable" by complementary frames, the loss of video content is minimal. So the overall effect of video quality after disruption is almost neglectable.

Error spreading technique, on the other hand, has a larger "post-disruption effect" because the loss of frames is spread uniformly to other frames. Since the lost frames may also have some significant content, the video quality after disruption is degraded considerably. Although spreading error to more frames increases frame rate after disruption, it still inevitably introduces a prolonged post-disruption effect.

4. EXPERIMENTAL RESULTS

We make an experiment on one episode (20 minutes) of sit-com video. We set summary ratio r as 100 and generate a disruption-tolerant video sequence that reduces the effect of a disruption of up to 15 seconds. We simulate network disruption by intentionally disconnecting and reconnecting network at random times. The displayed frames during one 15 second disruption are shown in Figure 3. The disruption-tolerant scheme requires an average client buffer of 5 frames. The buffer requirement on average is $1.2MB$ in our transmission of uncompressed frames.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a content-aware approach to generating disruption-tolerant video sequence for transmission. We introduced the concept of "substitutable content summary frame" and provided a method to select these frames. We also analyzed its performance in terms of initial delay, video buffer usage, and post-disruption effect. We plan to conduct a thorough evaluation of our approach and compare it with other error spreading and periodic broadcast based schemes [7] for handling burst errors. We also intend to extend our approach to scenarios where some amount of patching is feasible.

6. REFERENCES

[1] C. Alattinoglu, V. Jacobson, and H.Yu. Towards milli-second IGP convergence. IETF Internet Draft, Nov. 2000. draft-alaettinoglu-ISIS-convergence-00.txt.

[2] W. chi Feng. On the efficacy of quality, frame rate, and buffer management for video streaming across best-effort networks. In *Journal of High Speed Networks*, pages 199–214, 2002.

[3] F. Fitzek and M. Rei. Prefetching protocol for continuous media streaming in wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10).

[4] C. Krasic, J. Walpole, and W. chi Feng. Quality-adaptive media streaming by priority drop. In *NOSSDAV*, 2003.

[5] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. In *European Conference on Computer Vision*, 2002.

[6] T. Liu and J. R. Kender. Rule-based semantic summarization of instructional videos. In *International Conference on Image Processing*, 2002.

[7] A. Manhanti, D. L. Eager, M. K. Vernon, and D. J. Sundaram. Scalable on-demand media streaming with packet loss recovery. *IEEE/ACM Transactions on Networking*, 11(2):195–201, 2003.

[8] A. Markopulu, G. Iannaccone, S. Bhattacharya, C.-N. Chuah, and C. Diot. Characterization of failures in an IP backbone. In *Proc. IEEE Infocom*, Mar. 2004.

[9] B. Vandalore, W. chi Feng, R. Jain, and S. Fahmy. A survey of application layer techniques for adaptive streaming of multimedia. *Real-Time Imaging*, 7:221–235, June 2001.

[10] S. Varadarajan, H. Q. Ngo, and J. Srivastava. Error Spreading: A Perception Driven Approach Orthogonal to Error Handling in Continuous Media Streaming. *IEEE/ACM Trans. Networking*, 10(1), Feb. 2002.

[11] F. Wu, shipeng Li, and Y.-Q. Zhang. A framework for efficient progressive fine granularity scalable video coding. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(3):332–344, 2001.