Content Dependency of the Traffic Control in the Darwin Streaming Server

Karel De Vogeleer, Adrian Popescu, Markus Fiedler, and David Erman
School of Computing
Blekinge Institute of Technology
Karlskrona, Sweden
Email: {kdv,apo,mfi,der}@bth.se

***Abstract*—The Darwin Streaming Server employs stream thinning and transmission rate adaptation for traffic and congestion control. We investigate the relation of video content with the stream thinning and transmission rate adaptation process under 3GPP-adaptation. We observe that stream thinning is not only initiated upon fluctuation of network properties, but sometimes also upon the content of the streamed video. Moreover, we show that in extreme situations the video transmission is halted based merely on content dependent properties reflected in the RTSP feedback. This may have adverse consequences for the user experience. We put forward alterations to the 3GPP-adaptation traffic control of the Darwin Streaming Server that controls the stream thinning process. The alterations aim to avoid video freezes as a result of video content dependent properties. The results show that the probability of an interrupt-free playback is lower for the alterations than for the default streaming server algorithms.**

I. INTRODUCTION

Video-on-Demand (VoD) services are popular, companies as *YouTube* and *Netflix* exemplify this. Besides, VoD video streaming over the Internet sees an increase in application such as remote surveillance, security monitoring, smart home, environmental tracking, battlefield intelligence, distance learning, collaboration and interactive virtual environments, and others [1], [2].

For the transmission of video content over the Internet various protocols are used. The Real-time Transport Protocol (RTP) [3] is one of the most common open-source streaming protocols. RTP is also put forward by 3rd Generation Partnership Project (3GPP) in its Packet-switched Streaming Service (PSS) [4]. Adobe's Hypertext Transfer Protocol (HTTP) Dynamic Streaming system is an example of a proprietary streaming solution and is used in, e.g., Akamai's High Definition (HD) network. YouTube on the other hand encodes all videos in a Flash Video (FV) container for playback in Flash enabled browsers and uses RTP for its mobile variant.

Streaming protocols often use some kind of traffic and congestion control mechanisms. The aim of the congestion control is to ensure the timely delivery of video content under fluctuating bandwidth availability and to assure fair resource utilization between competing flows on the Internet. When the Transport Control Protocol (TCP) is used for media transportation, congestion control is handled by TCP itself. Though, many streaming protocols operate over User Datagram Protocol (UDP). In such cases it is not unusual that TCP like features, such as congestion control, are implemented at the application layer. For example Reliable UDP is used by the Darwin Streaming Server (DSS) [5] to create a reliable connectivity. The Reliable UDP feature must be explicitly enable in the Real Time Streaming Protocol (RTSP) SETUP request during client-server negotiation. Besides Reliable UDP, the DSS employs other methods to control the data flow to clients, i.e., traffic control. For example, traffic control is used in the DSS to control the amount of data in the jitter buffer on the client side. The DSS and a client participate in a closed loop feedback system. The streaming client shares information about the state of its jitter buffer to the DSS. Based on this information the DSS controls the *transmission rate* and the *quality level* of the requested media. By amending the transmission rate the DSS can control the growth and reduction of data in the jitter buffer. This may indirectly lead to an increase or decrease of the jitter buffer play-out time, especially for Variable Bit Rate (VBR) videos. Reducing the quality of the media is also know as *stream thinning*. Stream thinning removes information from the media stream resulting in a lower bit rate. This is possible because of the Group of Pictures (GOP) structure of video frames. When, for example, the available bandwidth temporarily decreases then it could be beneficial to reduce the bit rate of the video accordingly by dropping selected frames. The aim is to deliver at least some data before its play-out deadline. Stream thinning will not affect the timeline of the media frames but does affect the picture quality. Other bit rate adaption schemes may be used as well, e.g., Multi-Bit Rate (MBR) and Adaptive Multi-Rate (AMR). The purpose of the DSS traffic control mechanism is to assure that the jitter buffer on the client side doesn't overflow or underflow and that the jitter buffer play-out delay is close to a preferred delay. The DSS 3GPP-adaptation mechanism is the focus of this study. The rest of the paper is organized as follows. In section II we explain how the DSS' congestion and traffic control works. Section III elaborates on the content dependent behavior of the traffic and congestion control. We present measurements in section III that show that the DSS control decisions are content dependent and sometimes result in extreme behavior. In section IV-B we propose improvements to the DSS traffic control and evaluate its performance. We conclude with some remarks in section V.

II. TRAFFIC AND CONGESTION CONTROL

The DSS 3GPP-adaptation mechanism over UDP is the focus of this study. Defined by 3GPP-PSS [4], 3GPP-adaptation comprises a number of protocols and functionalities for reporting jitter buffer information to a streaming server via the Real-time Transport Control Protocol (RTCP). The DSS implementation of 3GPP-adaptation bases its traffic and congestion control on four parameters that are supplied by the streaming client via the RTCP on a periodical basis, more specifically, via Receiver Reports (RRs). A one second reporting interval is not unusual for RTCP operations. The following parameters are used by the DSS traffic and congestion control:

- BUFFER USAGE RATIO – the currently used percentage of allocated memory space for the jitter buffer;
- BUFFER DELAY RATIO – a percentage indicating the total time that would be needed to play all data in the buffer relative to a target play-out time;
- PACKET LOSS – the number of measured packet losses by the client; and,
- ROUND-TRIP TIME – the time in-between transmission of the last Sender Report (SR) and the arrival of a RR at the streaming server, subtracted by the time between the arrival of the SR and the transmission of the RR at the client's side.

For a Constant Bit Rate (CBR) video, the *buffer usage* and the *buffer delay* will behave similar. In the case of a VBR video however, the buffer usage and the buffer delay will vary based on the content of the video, but some dependance will be present. Conditional on the said parameters, the DSS will react to some extend in two different manners, as mentioned before: *transmission rate adaption* and/or *stream thinning*. Figure 1 shows the decision process as implemented by the DSS 6.0.3 defined in `RTPStream3gpp.cpp`. A flow chart of the decision process can be found in the work of Catalán [6]. The DSS adjust the transmission rate so that the play-out delay of the jitter buffer is as close to the target play-out delay as possible. Figure 1a shows that the traffic control adjusts the inter-departure time of individual packets in four different steps: +50%, +10%, -50%, and -20%. At the start of a streaming session the transmission rate adaptation is not activated. Instead, the streaming server schedules the packets 50% earlier until the jitter buffer is filled up to 70% or 15 s have passed since the streaming started. After one of these events, the traffic control as depicted in Figure 1a is initiated.

The transmission rate is adjusted by the DSS based on the inter-departure time between the past transmitted packets. The inter-departure time for the a packet is the sum of its default inter-departure time and an offset. The offset is a variable maintained by the DSS that holds the accumulative sum of offset adjustments calculated in the past. The offset for a particular packet is the previous inter-departure time multiplied by a percentage as given in Figure 1a plus the previous offset.

Stream thinning is used to control the population of the jitter buffer. The source code of the DSS describes seven quality levels for the stream thinning process. Zero is the default or original media quality, increasing the level results in a lesser media quality. Table I shows the properties of the seven quality levels. The result of thinning is a reduction of the video streams's bit rate. When bandwidth availability decreases, stream thinning is useful to deliver the media with minimal effect on the delivery timeline. In the source code of the DSS, level six overwrites the properties of level five.

The quality level is governed by the decision process in Figure 1b. The area marked by *increase* leads to the quality level increases of the video. This happens one step at a time. The area marked by *thin* (right shaded) reduces the quality level by half. The DSS source code talks about *thinning down aggressively*. Furthermore, the area *steady* will not alter the quality level of the video.

TABLE I: Quality levels used by the stream thinning process in the Darwin Streaming Server 6.0.3.

LEVEL	DESCRIPTION
0	default movie quality
1	no B-frames
2	75% P-frames
3	50% P-frames
4	25% P-frames
5	key frames only
6	key frames only + one P-frame

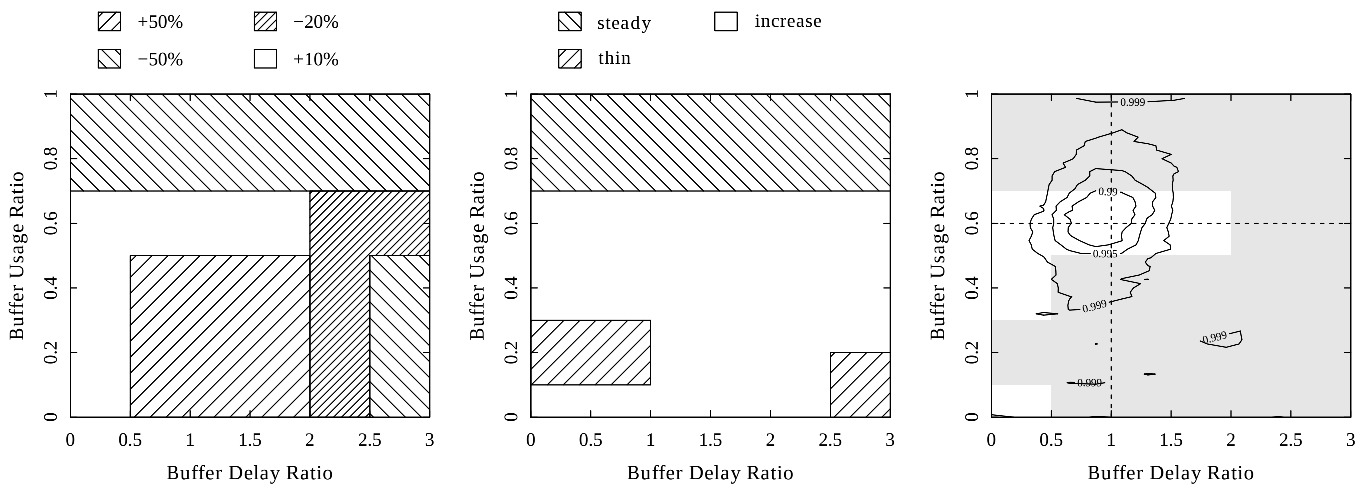
The decision process in Figure 1 is overwritten when *packet loss* occurs. For at least three consecutive RTCP cycles, the quality level is decreased one step and maintained or further degraded if the process from Figure 1 yields a lower quality. The inter-departure time offset is increased by 50% on the event of a packet loss, as per the source code: *not to drain down the buffer*. The code foresees an exception, if the Round-trip time (RTT) is smaller than 10 ms the inter-departure time offset is increased by 20%. In this case the packet loss is considered as plausibly a random packet loss.

The DSS also employs a *maximum quality level* which serves as a ceiling for the quality level. In other words, the maximum quality level limits the quality level of the decision process in Figure 1. The maximum quality level is set when *large RTT* values are measured. If four consecutive large RTTs are measured the maximum quality level is decreased. The maximum quality level is increased when at least m small RTTs values are measured. m is computed as half the number of better quality levels currently available plus one.

The difference between traffic and congestion control is not clearly defined in the DSS source code. Though, the anticipation of the DSS on RTTs and packet losses may be viewed as the congestion control and the reaction to play-out delay and buffer usage as the traffic control.

III. CONTENT DEPENDENT BEHAVIOR

When the bit rate of a VBR video and the future state of the Internet is know, the exact trajectory of the RTCP parameters can be mapped in the usage-delay plots of Figures 1a and 1b. Similarly, if the state of the Internet during playback shows small stochastic variations then the trajectory of the usage-delay mapping will be not identical but similar. The trajectory may be deemed deterministic. We conducted some experiments in the study on the Quality of Delivery (QoD) of variable bit rate video and observed that the jitter buffer exhaust during streaming often at the exact same time instants in the movie. Exhaustion at the same time instant would imply that the jitter buffer exhaustions are content dependent on the DSS traffic/congestion control behavior. To analyze this observation more closely we set up a set of experiments. We stream a video from a DSS 6.0.3 connected to a router with a fixed link to a Samsung Galaxy S and a HTC Dream over a



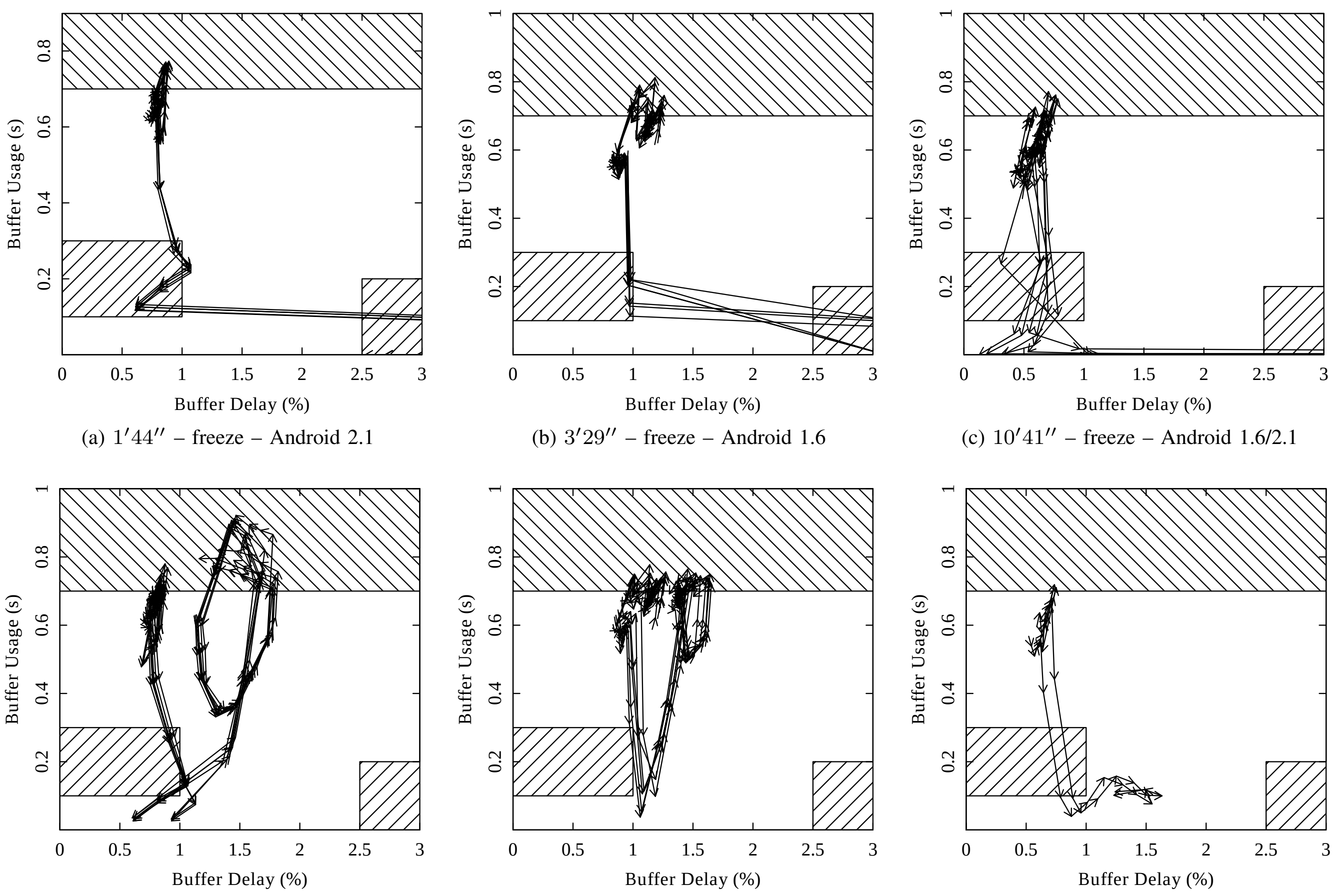
(a) Transmission rate adaptation (b) Stream thinning (c) RTSP feedback density example

Fig. 1: Visualization of the Darwin Streaming Server (DSS) traffic control decision process for (a) *transmission rate adaptation* and (b) *stream thinning*. The decision process is based on the parameters: *buffer usage* and *buffer delay ratio*. These parameters are reported by the client in real-time during streaming via RTCP feedback. Figure (c) shows an example of the parameter's densities reported by a client during the streaming of a 350 kbit/s H.264 AVC encoded video without audio over WLAN.

Wireless Local Area Network (WLAN) 802.11n network. The WLAN is solely used by the streaming client and positioned such to receive with optimal signal strength, the wireless connection is thus quasi optimal. We streamed the video *Elephants Dream*, released under the creative commons license [7]. The Samsung device runs a retail Android 2.1 version using PacketVideo's RTP implementation PVPlayer CORE/6.506.4.1 OPENCORE/2.02 and the HTC runs a customized Android 1.6 version with PacketVideo's PVPlayer CORE/8.000.1.1 OPENCORE/2.06. The DSS runs on a 32 bit Dell over Ubuntu 8.10 and the 2.6.27 Linux kernel. The video streamed is H.264 AVC encoded with MPEG Streamclip [8] at an average bit rate of 250 kbit/s in Quarter Common Intermediate Format (QCIF) without audio support and an I(119P) GOP structure. The compressed video is hinted by MP4BOX [9] with the Maximum Transmission Unit (MTU) set to 1440 bytes.

The video is streamed thirty times to both devices and the RTCP feedback is recorded on the server side. Only fourteen runs out of sixty completed without any jitter buffer exhaustion during playback. The other runs have at least one jitter buffer exhaustion or aborted playback before the end of the movie. Figure 1c shows the two dimensional density of the RTCP usage-delay feedback data reported by the client to the server. It is observed that the during the sampling of the buffer state by the client the play-out delay is close to the target and the buffer usage is on an average around 60%.

We focus specifically on three points in the time-space of the experiments, i.e., $1'44''$, $3'29''$ and $10'41''$. During these time instants the jitter buffer starves for several seconds with a given probability. Figure 2 plots the trajectory of the RTCP feedback in the delay-usage diagram during the listed time instances. We selected to plot only the trajectories of the videos that manifest no jitter buffer starvation before the times $1'44''$, $3'29''$, and $10'41''$ respectively. This ensures the synchronization of the different trajectories. The arrows show the direction in which the state of the jitter buffer is changing. The top three plots in Figure 2 show jitter buffer exhaustions, the bottom three plots show an interrupt-free playback. Due to the deterministic nature and content dependency of the system, the feedback trajectories are similar with some stochastic variations. We see in the Figures 2 that in both cases, the jitter buffer usage drops to a level between 30 % and 5 %. The trajectory of the feedback in the bottom Figures rises back to normal levels after the the drop. The trajectory shows low buffer usage, however the buffer doesn't exhaust. In the three top Figures however, the feedback trajectory does not rise back but instead plumbs even more. Simultaneously the play-out delay rises significantly. The latter is the indication of a jitter buffer exhaustion. The difference between the top and the bottom plots is that in Figures 2a, 2b and 2c the feedback trajectory passes through the shaded area defined by the buffer's $delay < 1$ and $0.1 < usage \leq 0.3$, in the *stream thinning* decision graph. When RTCP RRs feedback reports values within this area the video quality level is decreased by half, the DSS comments mentions: *thin down aggressively*. In this case the quality drops, e.g., from quality level 0 to 3, resulting in a slower population of the jitter buffer. Quality level 3 implies that 50% of the movie's P-frames are dropped. Given that the jitter buffer was already sparsely populated the client decoder retrieves data faster than the streaming server supplies, resulting eventually in a jitter buffer exhaustion. On the average 10 seconds after the jitter buffer exhaustion events, the jitter buffer recovers from the situation. In the meantime the quality of the video and streaming rate is restored by the streaming server to optimal settings. Similar observations can be made for the Figure 2c. However, as the freeze here is close to the end of the video, the jitter buffer doesn't have the time to recover before the end of the video playback. Besides reducing the quality level of the video in the said area, the inter-departure time offset is increased (see Figure 1a) with



(a) $1'44''$ – freeze – Android 2.1 (b) $3'29''$ – freeze – Android 1.6 (c) $10'41''$ – freeze – Android 1.6/2.1

(d) $1'44''$ – seamless – Android 2.1 (e) $3'29''$ – seamless – Android 1.6 (f) $10'41''$ – seamless – Android 1.6/2.1

Fig. 2: The trajectory of the RTCP feedback for the Elephants Dream movie for (a,d) $1'25'' < t < 1'55''$, (c,d) $3'20'' < t < 3'50''$ and (b,e) $10'25'' < t < 10'45''$. The trajectories of Android 2.1 and 1.6 are combined in Figure (c,f). Figures (d), (e) and (f) show a seamless playback of the movie whereas Figure (a) shows a freeze at $t = 1'44''$, (b) at $t = 3'29''$ and (c) at $t = 10'41''$.

50% in the majority of the trajectories in Figure 2. Figure 2d shows a trajectory that passes through the shaded area and the jitter buffer content drops for about 3 seconds to about 5% usage. In this particular cases the jitter buffer was able to recover from an imminent exhaustion, as the streaming server subsequently restored the quality level back to 0.

The reason for sudden drops in buffer usage is due the VBR nature of the video. Some scenes contain few motion or even sometimes still frames, e.g., black screens to build up suspense and tension. In such situation the video requires less information to be coded, hence resulting in a temporary drop in bit rate. The drop in the video bit rate is reflected in a decreasing buffer usage. Then the client reports this usage drop to the streaming server. If, by coincidence the RTCP trajectory goes trough the shaded area, the DSS will halve the video quality. Given that the video already has a low bit rate, the stream thinning effect reduces the bit rate even more. This may result in a jitter buffer underflow, or temporal extreme low buffer usage while the play-out delay is acceptable. It is noted that during these periods of low buffer usage the smooth playback is more vulnerable for adverse network effects. The problem here is that the DSS doesn't differentiate between buffer usage fluctuations resulting from the inherent video bit rate and network effects. Hence, the video may be thinned or suspended by the DSS streaming server based on video content dependent factors.

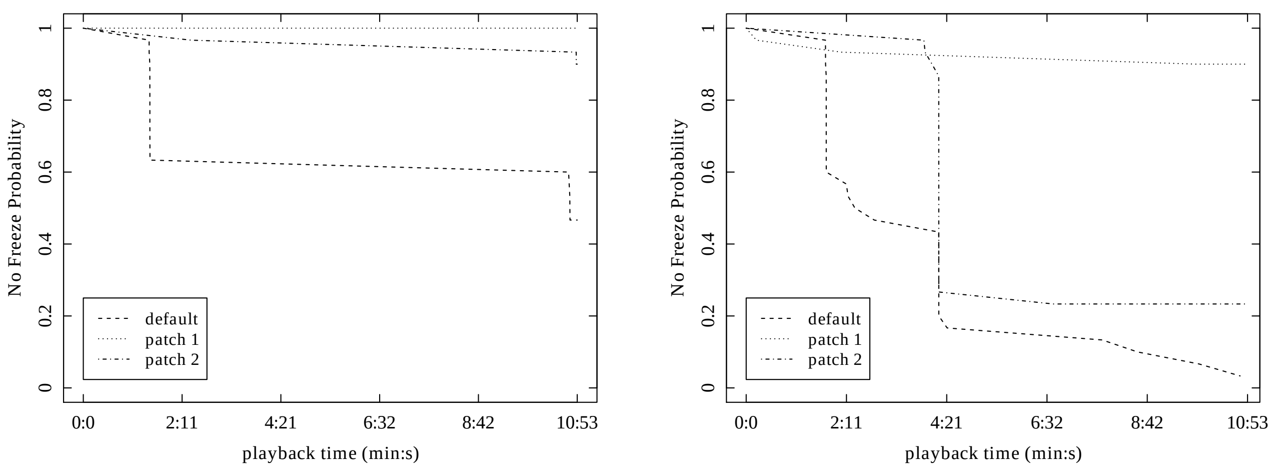
IV. Improvement to the Darwin Streaming Server's traffic control

A. Changes to the DSS traffic control algorithm

In the previous section we highlighted that the shaded area defined by *buffer delay* < 1 and $0.1 <$ *buffer usage* ≤ 0.3 in the *stream thinning* decision process may create unwanted effects, i.e., jitter buffer exhaustion. The DSS source code refers to this particular area as *danger for underflow*. We investigate two amendments to the DSS traffic control decision process that affect the stream thinning behavior. When the RTCP feedback is received within the said area the DSS halves the video quality by default. We propose the following alterations:

1) Increase the quality level one step, i.e., the said area is assimilated with the surrounding area;
2) Decrease the quality level one step, in contrast to the halving of the quality level.

The patches for these two amendments to the DSS 6.0.3 source code are given in the appendix *DSS patches*. The patches are



(a) HTC – Android 1.6 (b) Samsung – Android 2.1

Fig. 3: The plots show the probability of having no freeze before a given time instant in the movie. The lines are plotted for Samsung Android 2.1 in Figure (b) and for HTC 1.6 in Figure (a). The probabilities for the default DSS is shown together with the two proposed patches.

short and only affect a few lines in the source file.

B. Performance of the DSS source code amendment

The performance of the proposed amendments are measured with the same devices and settings as described in section III. To compare the performance of the default DSS and the patches we define the Area Under the Curve (AUC) metric. The AUC metric is the area under Complementary Cumulative Distribution Function (CCDF) curve for the event of having no freeze during the play-back before a given time. The curve is the complement of the probability that there is at least one freeze before a given time instant in a movie.

Figure 3 shows the probability of having no freezes before a given time instant with the default DSS and the two patches applied to the source code, for both the Samsung and the HTC device. When freezes would occur randomly in the movie the curves in Figure 3 would be a straight line from the top left to the bottom right corner. However, Figure 3 shows that the curves jumps at specific points in time. This indicates that at those points a freeze is more probably, e.g., Figure 3a shows that for the default behavior of the DSS the video has a probability to freeze of about 30% around 2:00. Ideally, the curves should be a horizontal line at 1, indicating a high probability for an interrupt-free playback. We see that for both devices the curves for the patches are closer to the ideal than for the default DSS behavior. This means that the patches reduced the probability for a jitter buffer exhaustions. Also, the curves for the patches are less jagged, this implies that the patches behave less content dependent.

Table II shows the AUC values for the experiments. The AUC metric shows that the patches perform better than the default DSS behavior. Also, patch 1 performs better than patch 2. A difference in behavior between the HTC and Samsung device is notable. Perhaps this can be attributed to the different Android and PVPlayer versions. Though, it should be noted that the Samsung device is more powerful than the HTC. But the HTC shows smoother playback capabilities, which may seem contradictory.

TABLE II: Values of the AUC which is a likelihood measure of an interrupt-free playback of the movie (AUC $\in [0, 1]$).

	HTC			SAMSUNG		
	default	patch 1	patch 2	default	patch 1	patch 2
FREEZE PR.	0.68	1	0.97	0.35	0.94	0.53

V. CONCLUSION

In this paper we analyzed the behavior of the default 3GPP-adaptation traffic and congestion control of the Darwin Streaming Server 6.0.3. The traffic and congestion control is based on the parameters: buffer usage, play-out delay, RTT, and packet loss. We observed that the buffer usage and play-out delay parameters vary dependent on the media content. Furthermore, we identified inconvenient behavior in the DSS traffic control decision algorithm. Sometimes the media transmission is suspended or the video quality is reduced based on video content dependent factors. We proposed two amendments to the DSS source code to increase the delivery reliability of data to the streaming client. We showed that the jitter buffer exhaustion probability is notably decreased by applying the amendments. By increasing the probability of an interrupt-free play-back the user experience may be increased.

We showed that it is possible to increase the reliability of the data supply to a video client. The side-effects of these amendments should be studied more closely, e.g., how the traffic control affects the recourse utilization fairness on a network. The patches yield better performance under optimal network conditions, but is this also the case for unreliable network conditions? The distinct behavior revealed in this paper may be specific for videos of the same type and GOP structure. A deeper investigation how this behavior generalizes to a wider array of video contents, more elaborate GOP structures and codec types must be conducted.

Appendix: DSS patches

Following are the patches used in this paper to investigate performance improvements to the DSS. The patches are to be applied to the file `RTPStream3gpp.cpp`.

1) Patch 1 – to remove the area of concern resulting in the increase of the quality level with one step:

```
365c365
<     {  fAdjustSize = kAdjustDown;
---
>     {  fAdjustSize = kAdjustUp;
```

2) Patch 2 – to reduce to thinning extend of the said area resulting in the decrease of the quality level with one step:

```
365c365
<     {  fAdjustSize = kAdjustDown;
---
>     {  fAdjustSize = kAdjustDownDown;
445a446,447
>     else if (fAdjustSize == kAdjustDownDown)
>         fRTPStream.SetQualityLevel(
>             fRTPStream.GetQualityLevel() + 1);
```

References

[1] Z. He and H. Xiong, "Transmission distortion analysis for real-time video encoding and streaming over wireless networks," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 16, no. 9, pp. 1051–1062, 2006.

[2] A. Ziviani, B. E. Wolfinger, J. F. Rezende, O. C. Duarte, and S. Fdida, "Joint adoption of QoS schemes for MPEG streams," *Multimedia Tools Appl.*, vol. 26, pp. 59–80, May 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1057905.1057908

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550 (Standard), Internet Engineering Task Force, Jul. 2003, updated by RFCs 5506, 5761, 6051. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt

[4] 3GPP TSG-SSA, "Transparent end-to-end packet-switched streaming service (PSS); protocols and codecs (release 9)," TS 26.234 V9.4.0, 2010.

[5] Apple Computer, Inc., "Darwin Streaming Server," 2011. [Online]. Available: http://dss.macosforge.org/

[6] E. Catalán, *Implementación de un servidor de streaming de vídeo adaptativo*, ser. Master Theses. Universitat Politècnica de Catalunya, 2009.

[7] Blender Foundation, "http://www.blender.org/," 2011. [Online]. Available: http://www.blender.org/

[8] Squared 5 srt, "MPEG Streamclip," 2011. [Online]. Available: http://www.squared5.com/

[9] GPAC multimedia framework, "MP4BOX," Institute Telecom, 2011. [Online]. Available: http://gpac.wp.institut-telecom.fr/