Video Streaming Over MANETs: Reality or Fiction?

Magnus Engh Halvorsen*
Tandberg ASA
P.O. Box 92
N-1325, Lysaker, Norway
magnus.halvorsen@tandberg.com

Thomas Plagemann, Matti Siekkinen
Department of Informatics
University of Oslo, Norway
P.O. Box 1080, Blindern
N-0316, Oslo, Norway
plageman, siekkine@ifi.uio.no

ABSTRACT

We focus in this paper on the performance evaluation of video streaming over mobile ad-hoc networks (MANETs) using resource constrained handheld real world devices, namely Nokia 770/N800 Internet Tablets. The main results from our experiments reveal that in principle video streaming in a multi-hop wireless ad-hoc network using such devices is possible. However, playing and forwarding the video stream at the same time could become infeasible, as receiving and playing a high-quality video stream at 1Mbps consumes almost 100% of the CPU time. In addition, an intermediate forwarding node is a potential bottleneck, being able to handle only three simultaneous high-quality (1Mbps) video streams before the CPU becomes overloaded. Our results emphasize the importance of resource monitoring and optimization for any future video streaming solutions for resource constrained devices in such networking environments.

Categories and Subject Descriptors

C.5.3 [**Microcomputers**]: Portable devices; H.5.1 [**Multimedia Information Systems**]: Video; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Experimentation, Measurement, Performance

Keywords

mobile ad-hoc networks, video streaming

1. INTRODUCTION

Mobile Ad-Hoc Networks (MANETs) are the existing solution to provide communication services in areas without networking infrastructure. Emergency and rescue operations is a promising application domain of MANETs, because these operations must often be performed in areas were the networking infrastructure has been damaged by the incident or has never even existed (e.g. mountains/tunnels). Multimedia streaming services could help to conduct such operations: The rescue crew can communicate via Voice-over-IP and head mounted cameras could provide video streams from the rescue scene to the command and control center.

However, live streaming has considerable resource requirements due to strict timing constraints for data delivery. Therefore, it is very important to understand the feasibility of implementing such services over MANET and to know the amount of resources required in order to provide acceptable quality of service. It is challenging to answer these questions because of the many factors that may have an impact: mobility and node density, node heterogeneity (CPU, memory, storage, bandwidth, transmission range, etc.), physical location (obstacles), etc. Except for some few works (e.g. [6]) , there are no well established testbeds available that allow realistic studies of multimedia streaming over MANETs. Furthermore, simulation environments are very useful for the development of new protocols and services, but generally provide very limited support to model the resource constraints of mobile wireless devices. Hence, there is an obvious need for empirical studies.

There has so far been little or no work published on the feasibility of multimedia streaming over MANETs focusing on the resource consumption and the achievable quality levels. The main contribution of this paper is to provide the first step towards a systematic evaluation of these properties. Due to the complexity of the overall problem, we focus on the most fundamental practical questions: Can we stream live video in a wireless multi-hop network formed by small PDA like devices? How much CPU time and bandwidth is consumed? What is the corresponding quality of the stream with respect to the resource consumption? By all means, in this paper, we are only able to take the initial stab at the problem and much work is left for the future.

In our experiments, we used Nokia 770 and Nokia N800 Internet Tablets (hereinafter referred to as just 770 and N800) that form a wireless multi-hop ad-hoc network using the OLSR[2] routing protocol. We set up monitoring services at all system layers (i.e., hardware, physical link, MAC, network, transport, application). We also used a laptop in some of the experiments together with the Nokia devices, which proved essential for discovering that some of the link layer

*Work performed during Master of Science studies at the Department of Informatics, University of Oslo

Mobimedia 2008 July 7-9, 2008, Oulu, Finland.
 ISBN
978-963-9799-25-7/08/07
DOI 10.4108/ICST.MOBIMEDIA2008. 3912

metrics extracted from the Nokia devices were misleading. The results from our experiments show that it is possible to stream video over multiple hops with these devices. However, we observe that streaming a high quality video at 1 Mbps rate can consume up to 50% of the CPU time of the forwarding node and 100% of the client node when using the 770 devices. In addition, the intermediate node is able to forward only three of such streams after which its CPU becomes overloaded.

We want to emphasize here that we focus on live streaming of video with minimal delay due to the nature of the potential application domain (emergency and rescue operations). Hence, we do not consider solutions such as Coolstreaming[9] that can suffer from considerable lag when data is streamed over many hops. Also, we do not consider optimizations such as multicasting or multi-path routing. However, our evaluation results strongly suggest that such approaches must be considered in order to cope with the resource requirements.

2. SETUP OF THE EXPERIMENTS

This paper takes a practical approach to mobile ad-hoc network research, performing experiments with real devices in a live test bed in a multi-hop configuration. In this section we describe the setup of our testbed. The strategy was to first perform an experiment to investigate the capabilities of and the statistics reported by the 770 wireless interface, and then evaluate the video streaming performance of the 770 in a multi-hop ad-hoc network.

2.1 Equipment

The equipment used in these experiments was primarily 770 and N800 (successor to the 770 with faster CPU and more memory) Internet Tablets. The devices communicate via 802.11abg WLAN and Bluetooth interfaces. The 770/N800 CPU has a DSP for decoding audio and video. The OS on these devices is a modified version of Debian Linux, and the complete open source platform is known as Maemo[1]. The Linux-based OS was the most attractive feature of the device, as it enabled us to take advantage of the large amount of open source software for the Linux platform.

Except for some of the measurements in Section 2.4, all units where running the latest OS version[2]. We also used a Dell XPS 1210m laptop in order to perform wireless sniffing.

2.2 Software

Three main pieces of software were required: Streaming server, streaming client, and monitoring software. We decided to base our solution on open source software, as it gives us the possibility of extending and implementing changes to the software components at a later stage.

We chose *FFserver* as the streaming server, which is part of the *FFmpeg*[3] project.

The standard 770 *Video Player*, is able to play both locally stored video files and streaming video. Unfortunately, we were unable to make Video Player play HTTP and RTSP streams served by FFserver. Furthermore, it does not play videos without audio stream. We therefore use a version of the well-known and versatile open source video player, *MPlayer*[4] that is optimized for the 770. The multimedia architecture of the Internet Tablets is based on GStreamer[5], and support for DSP decoding of various media formats is handled through GStreamer plug-ins. However, MPlayer does not integrate with the GStreamer stack, and only has support for DSP decoding of MP3 audio. By measuring the CPU utilization when playing a video using MPlayer, and comparing it to the CPU utilization when playing it in the standard Video Player, which has a GStreamer back-end, we were thus able to see the difference between regular and DSP optimized performance.

As for the monitoring software, during the wireless radio range experiment, we monitored the output of `iwconfig` and `ifconfig`. In the video streaming experiment, as `iwspy` was unsupported, the output of `iwconfig` on all nodes was used to monitor the quality of the wireless channel[6]. All nodes also captured complete headers (i.e. 128 bytes) of all incoming and outgoing OLSR traffic, as well as all incoming and outgoing RTP/RTCP and RTSP traffic, using *tcpdump*. We also logged the output of *olsrd*. In addition, we set up the Dell laptop as a sniffer to capture all complete frames (incl. payload) of 802.11 traffic using *Kismet* with the wireless card running in monitoring mode. Finally, all nodes were running *sar* from version 8.0.3 of the *sysstat* package, to capture hardware statistics, such as CPU utilization, network transfer statistics, disk I/O, etc. An overview of the metrics we could capture with the above software is given in Table 1.

In addition to the streaming server, video player, and monitoring software, all nodes were running version 0.4.10 of the OLSR daemon *olsrd*[7] . We also used *ntpdate* to synchronize the system clock of the Nokias. In this way, we could compare the measurements from each device, as the impact of clock drift should be marginal considering the short duration of the experiments and the granularity of our analysis.

2.3 Video clips

As basis for our measurements, we used two different video clips: (1) an action clip from a movie containing a lot of movement and frequent cutting, and (2) a clip of an interview with very little movement and very few cuts. Both clips were encoded in 6 different combinations of resolution, bit rate and encoding (see Table 2). Due to the small screen size of the Internet Tablets, we used full-screen playback in all our experiments, and the videos were encoded with this in mind. The screen of the tablets has a resolution of 800×480 pixels, which corresponds to an aspect ratio of 15:9. In order to make the most use of the 770 and N800's hardware pixel doubling capabilities (see Maemo web site), the test video clips were encoded to this aspect ratio, in 400×240 and 240×144 resolution. Based on these resolutions and the bandwidth of a typical 802.11 wireless link, we encoded our videos with bit rates of 1000, 500 and 200 Kbps. The codecs used were MPEG-1 and MPEG-4 ASP, as they are both in widespread use, and supported by FFserver and MPlayer. All video clips were encoded with a frame rate of 25 fps. As

[1]http://www.maemo.org
[2]At the time, this was version 3.2006.49-2 for the 770 and version 4.2007.38-2 for the N800.
[3]http://ffmpeg.mplayerhq.hu
[4]http://mplayer.garage.maemo.org
[5]http://gstreamer.freedesktop.org
[6]Although link quality value reported by `iwconfig` is supposed to be undefined when running in ad-hoc mode, preliminary testing showed that `iwconfig` on the 770 and the N800 produces sensible values also in ad-hoc mode.
[7]http://www.olsrd.org

Layer	Protocol/codec/device	Monitoring method	Useful information
Media layer	MPEG-1/MPEG-4 ASP	Simple subjective evaluation	Perceived quality
Session layer	RTSP	FFserver, tcpdump	Media control info, media location,transport protocol
Transport layer	RTP/RTCP	tcpdump	Payload type, sequence number, timestamps, interarrival jitter, fraction lost, cumulative no. packets lost, highest sequence no. received
MANET layer	OLSR/UDP/IP	olsrd, tcpdump	Message type, originator address, hop count, src./dest. IP
MAC layer	802.11 MAC	Kismet	Frame type and sub type, more fragments, retry, duration, source MAC address, receiver MAC address, transmitter MAC address, fragment number, sequence number
Physical layer	802.11 PHY	iwcon↓g	Link quality, signal level, noise level, bit rate, #Tx/Rx errors
Device hardware	Nokia 770 Internet Tablet	sar	CPU, memory/paging statistics, network/disk I/O activity

Table 1: Overview of monitoring architecture

Resolution	Bit rate (Kbps)	Codec
400 × 200	1000	MPEG-4
240 × 144	500	MPEG-4
240 × 144	200	MPEG-4
400 × 200	1000	MPEG-1
240 × 144	500	MPEG-1
240 × 144	200	MPEG-1

Table 2: Video clip encoding parameters.

live video streaming was the focus of our measurements, the clips were encoded without audio.

For experiments evaluating the impact of the DSP using the 770 Video Player (results in Section 3.2.4), we used the video clip *Discovery.avi*, which comes with the device, since Video Player does not seem to support playback of video files not having an audio stream. The clip is MPEG-4 (DivX) encoded in 352 × 208 resolution, with a frame rate of 15 fps and a bit rate of 389.5 Kbps. The audio track is 44100Hz, 16 bit stereo and MP3 encoded with a 64 Kbps bit rate. For comparison, we re-encoded the video stream to MPEG-1, as the DSP does not support decoding of MPEG-1 video. Since MPEG-1 does not support a frame rate of 15 fps, the frame rate had to be doubled to 30 fps, but we kept the bit rate as close as possible to the original.

2.4 Wireless radio range experiment

Preliminary testing revealed that some of the metrics reported by `iwconfig` on the 770, e.g. link quality and noise level, behaved differently than on other devices. As the values reported by `iwconfig` are driver dependent, the documentation for the Linux Wireless Tools was unable to help us. We were also unable to locate any documentation for the drivers. In addition, at the time these experiments were performed, the source code for the 770 wireless drivers had not yet been released.

In order to find out more about the statistics reported by the 770 wireless drivers, and get a better idea about capabilities of the wireless interface, we performed an experiment where we measured the statistics reported by the wireless driver on two 770s when communicating at different distances. The general idea was to start out with two devices at close range, and gradually increase the distance between them, while logging as much wireless measurement data as possible on the units. We performed measurements at the following distances: 0 meters, 1 meter, 2 meters, 4 meters, 8 meters, 16 meters, 32 meters and 64 meters. Care was taken to perform the measurements in an environment with as little interference as possible: The measurements were performed outside, on a dry, sunny summer's day, in level terrain on a large open field (a soccer field). We scanned the wireless medium using `iwlist` prior to each measurement to make sure that no other 802.11 networks were in range.

2.5 Video streaming experiment

We tested three different scenarios: Local playback, single-hop streaming and multi-hop streaming in a three-node ad-hoc network. In each scenario, all the videos were played in sequence while performing measurements as described in Section 2.

The local playback scenario allowed us to isolate any performance issues related to the video playback on the 770. The purpose of this scenario was to see how much resources are consumed by the playback of the media itself, without involving data transmission, routing, and other complicating factors.

During single-hop streaming we were able to see any overhead introduced by streaming, and the impact of wireless network transmissions without the complicating factor of routing and route management at the MANET layer. The measurements were performed with two 770 devices placed next to each other, with the wireless sniffer in close proximity.

Finally, during multi-hop streaming, we could see the impact of MANET routing, multi-hop wireless transmission, and the effects on the intermediate node. We found that the bomb shelter in the basement of the department building was a good location to conduct our experiments.

To make sure that an intermediate node is used for forwarding, we placed two nodes at opposite sides of a thick concrete wall and reduced their power. The third node was placed in such a way that it was within communication range of both nodes and would work as forwarding node (see Figure 1). By placing the sniffer node next to the intermediate node in our static 3-node MANET it was able to hear all three nodes.

3. RESULTS

Due to space constraints, we only highlight the main findings here. More results can be found in [5].

3.1 Wireless radio range experiment

Using one laptop and one 770, we were able to complete our measurements at all distances up to and including 64 meters. Using two 770s, we discovered that the maximum communication range was at a distance of approximately 60 meters.

From the results of our measurements, it is apparent that the noise levels reported by the Nokia wireless driver cannot be trusted. The Nokias reported a very low and almost

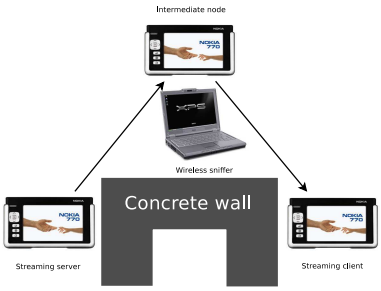


Figure 1: Three-node experiment setup

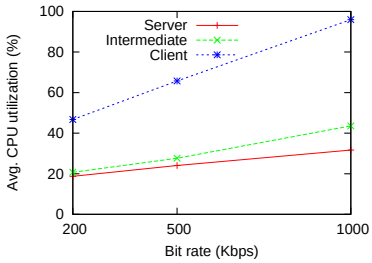


Figure 3: Average CPU utilization vs. bit rate in multi-hop streaming of MPEG-4 video.

constant noise level at all distances. Compared to the very fluctuating noise levels reported by the laptop, this leads us to conclude that noise detection on the wireless channel does not work on the 770. We also observed that the Nokia reported a constant bit rate despite the very varying channel conditions. The erroneous noise level reporting could be a possible explanation of this behavior.

We also observed that the link quality and signal level metrics reported by the devices fluctuate in an almost identical manner. The correlation coefficient for these two metrics during one of our experiments was 0.99 for the Nokia running OS 1.2006 and 1 for the Nokia running OS 3.2006. Similar correlation could be seen in the results from all the measurements. By examining the `iwconfig` output we discovered that the 770 wireless drivers apparently calculate the link quality as:

Link quality = *Signal level* − *Noise level*

Due to the already mentioned issues with reported noise level, this causes the relationship between link quality and signal level to be an almost constant difference. It also causes the reported link quality to drop sharply with distance, since signal level fades according to the inverse square law.

This experiment demonstrates that it is crucial to evaluate the robustness of a given metric before using it as input for e.g. a routing protocol.

3.2 Video streaming experiment

By streaming video clips encoded with different parameters, we were able to determine the impact of *encoding*, *resolution*, *bit rate* and *content* on the quality and performance of the video streaming solution. Subjectively, the quality of all the clips were good enough to watch on the 770, although the 240 × 144 200 Kbps video may be too grainy to see some details in the image. In this respect, 400 × 240 or 240 × 144 at 500 Kbps probably has a better bandwidth/quality ratio. We note that the perceived quality of all the video streams was the same in the single-hop and multi-hop scenarios. That is, the introduction of an extra hop did not introduce any noticeable effects on the client side.

Figure 2 shows the CPU utilization for all nodes during multi-hop streaming of the action clip in 400 × 240 resolution at 1000 Kbps. The results clearly demonstrate that the client suffers from the highest load, with the resource requirements of video playback dominating the CPU. In contrast, the CPU on the server and intermediate node is dominated by system time due to the network transmissions, with the intermediate node having the second highest load of all three nodes, as it has to both receive and forward each packet of the stream. Differences between the nodes become more pronounced as the bit rate is increased.

3.2.1 Bit rate

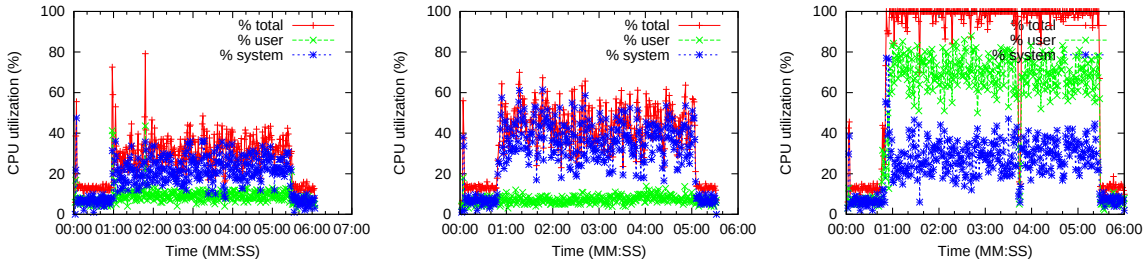
During local playback, the difference in CPU utilization on the client between 1000 Kbps and 500 Kbps for a clip in 400 × 240 resolution encoded with MPEG-4, was about 20%. This difference was about the same for MPEG-1 encoded video. For video in 240 × 144 resolution, the difference between 500 Kbps and 200 Kbps was only about 5% for both MPEG-4 and MPEG-1 video. This shows that the CPU utilization on the client associated with an increase in video bit rate is the same for both MPEG-1 and MPEG-4 encoded video.

Single stream: Figure 3 shows the relation between the increase in bit-rate and in CPU utilization in the multi-hop streaming scenario with MPEG-4 video (MPEG-1 results are similar). We see that as the video bit rate is increased, the CPU utilization on the intermediate node increases by roughly twice the amount as the server, which raises concerns about the capabilities of the forwarding node. We look at the scalability in the case of multiple simultaneous streams later in this section.

During the streaming of the highest quality clips, the playback was very choppy and generally unwatchable. From Figure 2(c) we can see why: The CPU utilization on the client was 100% most of the time. While the 770 is able to produce smooth playback of the video locally, the added resource costs associated with streaming the video over a network overloads the CPU.

When examining the network transfer statistics, we found that throughput varied considerably throughout the streaming sessions. For 1000 Kbps clips, there were variations in transmission speed as large as 1500 Kbps, while for 500 Kbps and 200 Kbps, there were variations of about 700 Kbps and 200 Kbps respectively. The results for MPEG-1 encoding were similar to the results for the MPEG-4 streams. Block device statistics show that FFserver reads the clips from disk at an almost constant data rate, but it seems it does not transmit it over the network at a constant rate. In fact, the fluctuations in transmission rate increase with the bit rate of the video stream. Plots from a rate analysis of the encoded video clips using *mpeg_stat*[8] (version 2.2b), show that the bandwidth utilization was almost equal to the variations in bit rate of the video files, meaning that the fluctuations

[8]http://bmrc.berkeley.edu/frame/research/mpeg/



(a) Server node (b) Intermediate node (c) Client node

Figure 2: CPU utilization during multi-hop streaming of 400 × 240 1000 Kbps MPEG-4 video

in bandwidth were due to the bit allocation of the video encoding. The results suggests that increasing the bit rate of the video stream increases the bandwidth requirements by more than the proportional increase in bit rate. We observed that the maximum bandwidth requirement seems to be about 150% of the video stream bit rate.

Multiple streams: With these results in mind, we performed another set of measurements to see the impact of multiple simultaneous streams on the server and intermediate node. First, we used the laptop as a client and connected directly to the server. Over a single-hop, we were able to play five simultaneous streams of the 400 × 240 1000 Kbps streams before the quality started to degrade. In order to see the impact of multiple streams on the intermediate node, we set up a multi-hop topology as previously described, but this time using the laptop as a client. With this configuration we were able to reliably stream either three simultaneous 1000 Kbps streams, six simultaneous 500 Kbps streams, or 12 simultaneous 200 Kbps streams. At this point, CPU utilization on the server was about 60%, and 90% on the intermediate node. If more streams were added after this point, the quality of the other streams became increasingly degraded, and the route from the server to the client became very unstable, breaking and reestablishing several times. We suspect that these routing instabilities were due to the overloaded intermediate node being unable to maintain the routes, causing them to time out. As we reduced the number of streams, the route became stable again.

The results also show that as more streams are added, the connection setup is the most resource intensive part for the server. As the load on the server increased, the quality of already established streams was degraded during setup of new streams, although all the streams played smoothly once set up.

To see the effect of transmitting several low-bit rate streams compared to fewer streams of higher bit rate, we compared the results of streaming one single 1000 Kbps stream to five 200 Kbps streams. The results show that on average, the CPU utilization was slightly higher for the five lower-quality streams, but that the CPU utilization for the high-quality stream was more fluctuating. This was reflected in the network transfer statistics, where the throughput was more stable in the case of the five lower-quality streams. The increased variations for higher bit rate videos described earlier explains this behavior.

3.2.2 Resolution

In order to see the impact of resolution on video playback performance, we compared the 500 Kbps clip in 240 × 144 resolution to the same clip with the same bit rate in 400×240 resolution. The results for MPEG-4 show that the CPU utilization was 20% higher for the clips in 400 × 240 resolution, while the difference for the MPEG-1 encoded clips was closer to 10%. Subjectively, the perceived quality of the video was higher for the 400 × 240 resolution clips.

3.2.3 Clip type

The content and amount of motion in the clips had little impact on the results, with the exception of completely still images. The interview clip is divided into four segments, each of which starts with a still image showing the topic of the upcoming segment. During these seconds, both the CPU and network bandwidth utilization dropped drastically. For the rest of the clip, the results were similar to the action clip.

3.2.4 Encoding and DSP

The results showed that the playback of MPEG-4 encoded video caused slightly higher CPU utilization on the client node compared to MPEG-1. As MPEG-4 is a more advanced encoding scheme than MPEG-1, this is not surprising. The choice of encoding scheme had no apparent effect on neither the server node nor the intermediate node. RTCP reported a total of 50% higher packet loss for the 1000 Kbps MPEG-4 streams compared to the MPEG-1 streams, while analysis of the captured traces shows 0% packet loss, and very few link layer retransmissions. Thus, the packet loss reported by RTCP are packets dropped by the application, due to the high CPU utilization for these streams.

For MPlayer, the results for both videos were similar, at about 60% CPU. For VideoPlayer, however, playback of MPEG-4 video consumed about 40% CPU time. That is 20% less CPU resources than compared with MPlayer, and 60% less than for MPEG-1 video. By examining the interrupt statistics from the two experiments, we learned that the number of DSP interrupts/second was twice as high during playback of the MPEG-4 video in Video Player compared to MPlayer. This suggests an increase in DSP utilization, which in turn explains the much lower CPU utilization.

3.2.5 Comparison with N800

We only had access to a single N800 to perform our comparison. Since the results from our measurements with the 770 showed that device performance during video playback was by far the most resource consuming task, we chose to look at the performance of N800 during local playback of video. It is worth noting that the N800 does not support

DSP decoding of any video format, but decoding of most common audio formats is still supported. In contrast with the 770 Video Player, the N800 Media Player is also able to play videos without audio streams. We could therefore conduct a performance test between MPlayer and the N800 Media Player using our regular experiment videos.

Interestingly, MPlayer outperforms Media Player on the N800 for both MPEG-1 and MPEG-4 video, with the difference being the most pronounced for MPEG-1. During playback of 400 × 240 1000 Kbps MPEG-1 encoded video, the average CPU utilization was about 40% for MPlayer and about 80% for Media Player. For MPEG-4, the results were about 40% CPU in MPlayer and about 60% in Media Player.

3.2.6 Summary

Of the video parameters, bit rate is the only one where changes have a noticeable impact on all nodes. Clip content only has an effect in the case of still images, and both resolution and encoding only impact the client node.

From the results, we see that playing and forwarding at the same time could become infeasible. We also see that the intermediate node is a potential bottleneck. Thus, resource monitoring is crucial for a video streaming solution for resource constrained devices.

3.3 Advantages of cross-layer monitoring

So far the primary focus of our analysis has been the CPU utilization. However, an incident during the multi-hop streaming of the 240 × 144 200 Kbps MPEG-4 video clip, gives an opportunity to demonstrate the advantages of monitoring on several layers.

The visible effect at the client side was that the playback froze for 30 seconds, before resuming. Figure 4(a) shows how the client's CPU utilization dropped. The other nodes experienced similar behavior. By examining graphs of the network transmission statistics reported by *sar* on each of the nodes, we found that the client stopped receiving data at the time when playback froze. The intermediate node stopped both receiving and forwarding data, and the server stopped transmitting data. Figure 4(b) shows the network transmission statistics as reported by the server. Graphs for the intermediate and client node are similar. By examining the logs from the OLSR routing daemon, we found that the route broke between the intermediate and the client node at the time the transmission stopped. Finally, by examining the wireless link statistics reported by the intermediate node (Figure 4(c)), we determined that a drop in link quality at the intermediate node caused the route to break. We were thus able to conclude that playback froze at the client side as a consequence of the route between the intermediate and client nodes breaking, due to low quality of the wireless link.

Many papers have been published on cross-layer optimization for wireless networking (e.g., [3]). Indeed, we also observed how combining information from several layers can provide valuable information to a video streaming solution on what actions to take when reacting to degrading QoS. We demonstrated here that it, at least to some extent, could be possible to implement such type of schemes with real devices. However, while we are able to monitor the relevant metrics in real time, it remains as a challenge for future work to also analyze this data efficiently and accurately enough in real time, in order to enable a video streaming solution to adapt to the changing environment and maximize QoS.

4. RELATED WORK

Providing delay and bandwidth sensitive services such as multimedia streaming in MANETs are a major challenge. There exists little prior research combining the areas of multimedia streaming and MANETs. Also, most MANET research is theoretical, and experiments are mostly based on simulations, such as in [10]. Concerning multimedia streaming on mobile devices, most research only considers mobile clients, and not mobile servers.

[12] performed real-world experiments with video streaming in mobile ad-hoc networks. Their analysis focuses on hardware resource management. In contrast to the work presented in this paper, the nodes in their experiments were quite powerful laptops running primarily closed-source software. Still, their findings are consistent with ours, in that the CPU load on the intermediate node is a cause for concern.

The authors in [1] performed subjective monitoring of video streaming to mobile phones. While there are several differences (UMTS instead of 802.11, lower video bit rate, client-server model, etc.), the results from their work show that CPU and battery is a limitation for these devices.

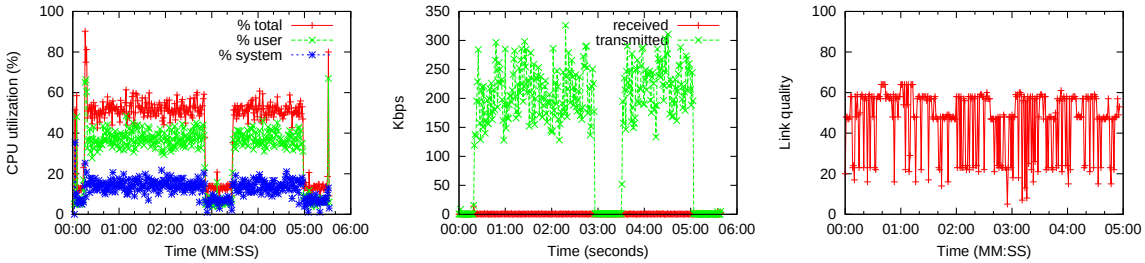
[8] was among the first to perform a measurement study of video streaming performance on a wireless network, using a combination of packet capturing with tcpdump at each node and wireless sniffing. [7] evaluated the performance of RealVideo streaming in a wireless network using end-to-end measurements, performing experiments with different signal-to-noise ratios and competing cross-traffic. However, both works performed their measurements on a regular 802.11 WLAN in infrastructure mode.

[11] performed a performance evaluation of both video and voice traffic in a real-world mesh network. Still, these experiments were not performed using actual video and voice streaming software, but by replaying an RTP stream using `RTPtools`. It thus only looks at the networking aspect, and does not take into account the resource requirements of video playback on the client node, for instance.

5. CONCLUSIONS

In this paper, we described how we created a platform for open source, mobile ad-hoc network video streaming using Nokia 770 Internet Tablets. We explained how we implemented cross-layer monitoring of the video streaming solution and evaluated the performance of such a video streaming solution.

We observed that the information reported by the 770 WLAN drivers is limited and unreliable, which implies that it is crucial to evaluate the robustness of a given metric before using it as input for QoS adaptation algorithms, as such discrepancies can have unexpected impact on the behavior. We then showed that, in principle, video streaming in MANETs using such resource constrained devices is possible. However, we have not tested with mobility and the size of the tested network is limited. Monitoring results revealed that due to lack of CPU resources, playing and forwarding at the same time could become infeasible. In addition, the intermediate node is a potential bottleneck for the same reason, while it still has spare bandwidth. Thus, resource monitoring and optimization (e.g. multi-path routing[4]) are crucial for any feasible video streaming solutions for such resource constrained devices.



(a) Client node's CPU utilization (b) Server node's network transfer statistics (c) Intermediate node's reported link quality

Figure 4: Interrupted multi-hop streaming

The most limiting factor of video streaming performance in MANETs appears to be the CPU of the client, so conservation of CPU time must be a priority when designing a streaming solution for MANETs. We also found that the bit rate of the video stream is the main parameter affecting the performance of all nodes in the path, and see that the MANET must be able to handle peak bandwidth requirements that are much higher than the video bit rate. On the client node, both resolution and encoding also have a large effect on resource consumption.

As for future work, we plan to examine the video streaming performance of the N800 and possibly implement streaming of live video from the N800 web camera. We would also like to design and build a more scalable wireless, multi-hop test bed and introduce mobility in the experiments. Development of new, and evaluation and improvement of existing, services that rely on monitoring services based on what we have learned about the capabilities of real world devices is another challenging future thread of work.

Acknowledgments

This work was supported by the CONTENT Network-of-Excellence (EU FP6) and the VERDIKT Programme of the Norwegian Research Council through the DT-Stream project (project number 183312/S10).

6. REFERENCES

[1] P. Backe, A. Andersen, T. Plagemann, and A. Spilling. Mobile services for horse-race betting: a qos study of streaming horse-race videos to 3g phones. In *MobiMedia '06: Proceedings of the 2nd international conference on Mobile multimedia communications*, pages 1–5, New York, NY, USA, 2006. ACM.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), Oct. 2003.

[3] Q. Du and X. Zhang. Cross-layer resource-consumption optimization for mobile multicast in wireless networks. In *Proceedings of WOWMOM '06*, pages 368–376. IEEE Computer Society, 2006.

[4] E. Gabrielyan and R. Hersch. Reliable multi-path routing schemes for real-time streaming. *Proceedings of ICDT '06*, pages 65–65, 2006.

[5] M. E. Halvorsen. Quality of service monitoring for video streaming in mobile ad-hoc networks. Master's thesis, University of Oslo, February 2008.

[6] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. *Proceedings of INFOCOM 2006.*, pages 1–12, April 2006.

[7] Y. Koucheryavy, D. Moltchanov, and J. Harju. Performance evaluation of live video streaming service in 802.11b wlan environment under different load conditions. Technical report, Institute of Communication Engineering, Tampere University of Technology, September 2004.

[8] T. Kuang and C. Williamson. Realmedia streaming performance on an ieee 802.11b wireless lan. Technical report, Department of Computer Science, University of Calgary, 2002.

[9] B. Li, S. Xie, Y. Qu, G. Y. Keung, C. Lin, J. Liu, and X. Zhang. Inside the new coolstreaming: Principles, measurements and performance implications. *Proceedings of INFOCOM 2008*, pages 1031–1039, April 2008.

[10] K. Rojviboonchai, F. Yang, Q. Zhang, H. Aida, and W. Zhu. Amtp: a multipath multimedia streaming protocol for mobile ad hoc networks. *Proceedings of ICC*, 2:1246–1250 Vol. 2, May 2005.

[11] Y. Sun, I. Sheriff, E. M. Belding-Royer, and K. C. Almeroth. An experimental study of multimedia traffic performance in mesh networks. In *Proceedings of WiTMeMo '05*, pages 25–30, 2005.

[12] P. Xue and S. Chandra. Revisiting multimedia streaming in mobile ad hoc networks. In *Proceedings of NOSSDAV '06*. ACM, 2006.