IJCSN International Journal of Computer Science and Network, Volume 5, Issue 3, June 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Wireless Multi-Hop Dynamic Live Video Sharing using Android Gadgets

[1] Suvarna L. Kattimani, [2] Deepa Deshpande, [3] Nazeera Madabhavi, [4] Rohini Naganur, [5] Savitri Seeparamatti

[1]Asst. professor, Dept. of computer science and Engineering,
BLDEA'S CET, Vijayapur,Karnataka, India.

[2,3,4,5] UG Scholar, Dept. of Computer science and engineering,
BLDEA'S CET, Vijayapur, Karnataka, India.

Abstract - With the rising penetration of smart phones in the consumer market, mobile multimedia content is becoming the dominant form of information that people produce and consume on a daily basis. In this paper we present a wireless multi-hop video streaming application for mobile phones with the Android operating system. This application allows to share live information captured by mobile phone sensors (e.g., camera, microphone) with persons that might be multiple wireless hops away. The video streaming is based on peer-to-peer communication between mobile phones, i.e. without the use of video processing servers or network infrastructure. We show the feasibility of such peer-to-peer video streaming application for Android phones in a variety of experiments that evaluate various video streaming scenarios, including various video codecs and various generations of Android phones.

Keywords - *Android phones, Ad-hoc Network, Multi-Hoping, Video Streaming.*

1. Introduction

The International Telecommunication Union's (ITU) statistics on mobile subscriptions indicates five billion mobile subscriptions for 2010 [1], with a 17% penetration of smart phones in 2009 [2]. The rapid adoption of smart phones has created a unique opportunity for mobile multimedia services for mobile users. Currently a majority of smart phones are equipped with both hardware that supports real-time video processing and ad-hoc wireless communication between peers and this allows real-time video streaming over multiple wireless hops between peer devices. Phones within communication range of each other automatically establish a wireless link creating a client mesh network (ad-hoc network of devices). Each phone in the client mesh network is able to produce/consume video and also acts as a relay to forward video to its next hop neighbors. Peer-to-peer video streaming from the cameras on smart phones to people nearby allows users to share what they see. Such streaming can be used in a variety of applications, in particular in various social network applications including sharing unforgettable moments with friends that can be multiple wireless hops away, cooperative fieldwork (providing video sharing for teams distributed in a small area, e.g. teams of repairmen, and search and rescue teams in disaster areas), and support for health impaired persons including the elderly.

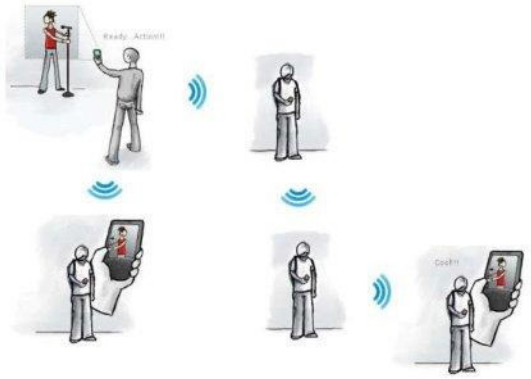


Fig1.Example Application scenario

There exist solutions that provide video services for mobile devices. In [3], Cycon et. al. present a peer-to-peer video-conferencing application with modified H.264 video codec for mobile phones. However, the use of a customized codec and development library poses an issue on portability to other mobile devices for wider deployment. In addition, the paper does not present results to demonstrate the feasibility in an actual network deployment scenario. Qik [4] is one of the best known

real-time video streaming services, which has support for a number of mobile devices. It adopts the client-server architecture; that is, mobile phones (with Qik as the client) stream live video to centralized processing servers using the available network infrastructure (such as cellular or Wi-Fi infrastructure networks). Video sharing is then done over the Internet, placing great dependency on the network infrastructure. Peer-to-peer live video streaming is currently not supported by Qik.

In this paper we present a wireless multi-hop video stream-ing application for the Android based mobile devices (including Android based tablet PCs). This application allows users to capture live video using camera on mobile devices, and to share this video feed with people nearby using a free-of-charge wireless mesh network. Routing protocols can be installed to facilitate the multi-hop communication to go beyond a single hop. Fig. 1 shows an example application scenario in which a person streams live video of a concert to friends nearby over the wireless mesh network of phones, without being caught by the expensive mobile phone bill. We evaluate the presented peer-to-peer video streaming application in a variety of experiments. In these experiments we show feasibility of peer-to-peer video streaming for various generations of Android phones and also evaluate performance of various video encoding and decoding schemes.

The remainder of the paper is organized as follows. In Section 2, we provide a brief overview of the fundamental building blocks that have been used in the development of our solution. This is followed by a description of the design and implementation of the peer-to-peer video streaming application in Section 3. In Section 4 we present the evaluation of the video streaming application in a number of scenarios with the emphasis on codec performance, multi-hop forwarding performance and the overall performance of the application on various generations of Android phones. The conclusion and future work are presented in Section 5.

2. The Building Blocks: Overview

In this section we provide a brief overview of the building blocks that we have used for prototyping the peer-to-peer video streaming.

2.1. Overview of Android System

Android is an operating system for mobile devices that are developed by the Open Handset Alliance. User applications are mostly written in Java and run on Android's own Java virtual machine (named Dalvik). Fig. 2 shows the Android system architecture, which consists of the Linux kernel with device drivers and the Android runtime environment (along with a number of libraries) that support interactions between the Linux kernel and the high-level application framework. The application framework released in a bundle as the Android SDK [5] provides high-level Java interfaces for accessing the underlying resources, such as camera and Wi-Fi. For example, our video streaming application makes use of the activity manager to detect and respond to events when triggered. The use of standard development toolkit encourages interoperability between components and maximizes portability of the application.

In addition to the SDK, there is also a native development toolkit (NDK), which supports the use of native C or C++ codes in the applications. The NDK is an extension of the SDK to allow the development of lower-level source codes for more efficient data processing in the system. In our application, parts of the video encoding and decoding are implemented using the NDK to allow more efficient processing of video streams.

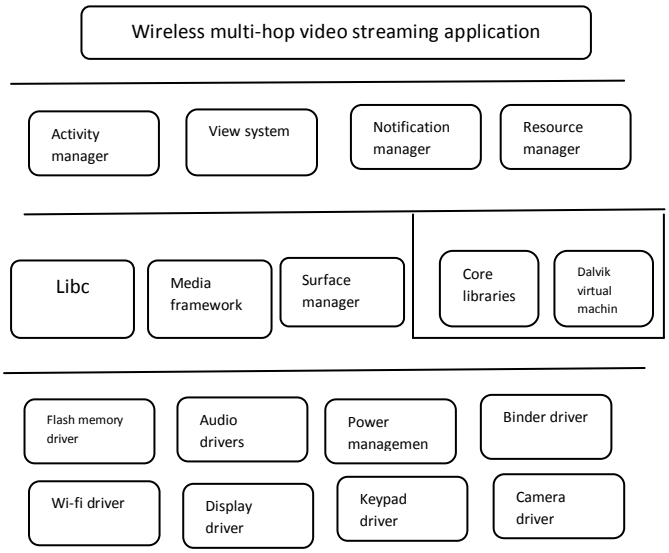


Fig2: Android System Architecture

Above figure shows the Android system architecture, which consists of the Linux kernel with device drivers and the Android runtime environment (along with a number of libraries) that support interactions between the Linux kernel and the high-level application framework. The application framework released in a bundle as the Android SDK [5] provides high-level Java interfaces for accessing the underlying resources, such as camera and Wi-Fi. For example, our video streaming application makes use of the activity manager to detect and respond to events when triggered. The use of standard development toolkit encourages interoperability between components and maximizes portability of the application.

2.2 Codecs and Method of Video Encoding

Video encoding and decoding is an important aspect of any video streaming application. There are many ways by which a video can be encoded or decoded. We briefly describe two widely used video coding techniques which are implemented in our application.

- *Intraframe encoding* is the simplest form of encoding. It treats every frame as an individual image to encode. This method is resilient against lost frames due to each frame having enough information to create an entire image.

- *Interframe encoding* uses two types of frames, i.e., the key frames and predicted frames, for better compression ratio. The key frame contains complete information to create an image, whereas the predicted frames only contain the differences between frames thus previous frames are required for their successful decoding.

In the proposed video streaming application we provide support for three video codecs and evaluate their performance. These codecs include (i) Motion JPEG (MJPEG), which is a form of intraframe encoding that encodes each frame as a JPEG image; (ii) MPEG-4, which supports both intraframe and interframe encoding and also allows profiling (resulting in different performance) for devices with different resource capabilities, and (iii) H.264, which is a popular coding technique that has been used in a number of online multimedia services (such as YouTube). Similar to the MPEG-4 encoding, H.264 allows profiling according to the needs/capabilities of various devices.

3. Design and Implementation

In this section, we describe the design of our video streaming application and its implementation on the Android system.

3.1 Routing

To stream a video across multiple hops, a multi-hop path between the sender and receiver has to be found. A routing protocol is therefore required to establish and manage connectivity within the mobile ad hoc network. A routing protocol may form various logical network topologies depending on the selected metrics (such as link quality) and QoS requirements.

In our prototype implementation, the peer-to-peer video streaming application relies on a network layer routing protocol (such as OLSR [6]) to carry out the routing tasks.

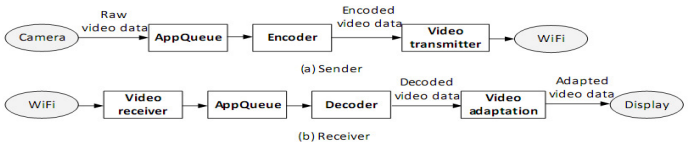


Fig. 3. Procedures of encoding and decoding

The information that the routing protocol can provide to the application layer (e.g. link quality) can be used by our video streaming application to adapt the video to the current context of the network, similar to our work presented in [7]. However such adaptation is out of scope for this paper as we concentrate here on the feasibility of using various generations of Android phones for multi-hop infrastructure less video streaming

3.2 Video content providers discovery

Routing is an important aspect of peer-to-peer video stream-ing as it discovers the path from the sender to the receiver. Another crucial aspect of the video streaming is the discovery of video content providers in the ad-hoc networks. In the current version of the proposed application, the video con-tent providers use periodic broadcast advertisements over the multi-hop network to advertise to potential receivers whenever they are ready to stream a video. Therefore, video content consumers within the network will receive information about who is offering video contents and the content description.

This approach to content provider discovery can be extended in the future to support a publish/subscribe mechanism. Clients could subscribe for video content based on their interest and needs and receive a notification when a video meeting their interest is available.

3.3 Video en/decoding

Video frames need to be encoded (using one of the codecs discussed in Section II) before sending on to network. Encoding can significantly reduce the size of individual video frames, therefore minimizing the bandwidth required for the streaming application. Fig. 3 shows the procedures of encoding on the sender and decoding on the receiver.

As shown in Fig. 3, when starting the streaming application raw video data is retrieved from the camera and stored in the application queue (for buffering purpose). This raw data is then passed to the encoder, which encodes the data using the selected codec and encoding technique. The encoded video frame is then

transmitted over-the-air by the Wi-Fi module. At the receiver, when an encoded video frame arrives it is buffered and sent to the decoder. Before being displayed on the screen, the video frames may require adaptation to the hardware specifications (e.g., screen resolution).

In the current version of the application, we implement the encoder and decoder in native C++ (supported by the NDK) for efficiency reason. Other application components are implemented in Java, which is supported by the Android SDK.

4. Evaluation

In this section, we present an evaluation of the video streaming application in a number of experimental scenarios and discuss the results.

4.1. Experiment Devices and Setup

To demonstrate the feasibility of the video streaming application on Android phones, we put together an experimental testbed using six off-the-shelf Android mobile phones. The selected mobile phones represent three generations of mobile devices on the consumer market. As shown in Fig. 4, these include the early version of HTC Dream, Nexus One, and the latest Samsung Galaxy S2. Table I shows their corresponding hardware specifications.

(a) HTC Dream (b) Nexus One (c) Samsung Galaxy S2

Fig. 4. Experiment devices

Table 1: Device Specifications

	HTC Dream	Nexus One	Samsung Galaxy S2
Processor	528 MHz	1 GHz	1.2 GHz dual-core
Memory	192 MB	512 MB	1 GB
Connectivity	802.11b/g	802.11b/g/n	802.11a/b/g/n
OS Version	1.5	2.1	3.0

As shown in Table I, each generation of mobile phone is equipped with significantly different hardware resources; that is, the newer phones have almost double resource capacity compared to older generations. Therefore, the newer generation mobile phones (such as the Samsung Galaxy S2) should achieve better performance in video/data processing. Based on this understanding of the differences in the hardware specifications we decided on the way in which the experimental testbed should be set up.

We deploy the application on each of these phones. Fig. 5 shows a screen shot of the live view of the application and its menu bar. In the release version shown in Fig. 5, it has a simple user interface with two functional buttons:

- *Start Streaming* triggers the actions to broadcast information to the multi-hop network that the node is a video content provider, gather raw video frames from the camera, encode these video frames using a particular codec, and finally stream out these encoded video frames.

- *Request Video* starts the discovery process to search for video content providers within the network; following that, users can select from a list of providers the node from which they will receive video streams.

Due to the use of the standard Android development kits, the application can be easily deployed on all of these mobile phones without modifications to the source code.

4.2 Performance of video codecs and coding

In a live video streaming application, the performance of video encoding and decoding is an indication of the feasibility of the application. The speed of en/decoding has impact on perceived quality of the video and in turn affects the usability of the application. We conduct experiments to investigate the performance of different codecs and coding methods in terms of the average en/decoding time of single video frame and the average size of a compressed frame under different video

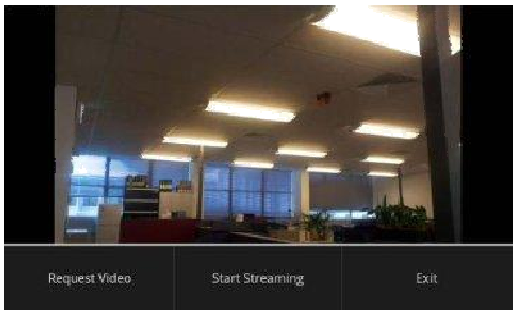


Fig. 5. Screenshot of the Application

quality (as represented by PSNR - peak signal-to-noise ratio). To vary the video quality, we adjust the associated parameters in the encoder.

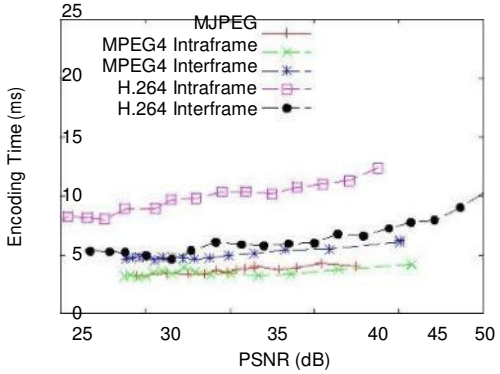
Evaluation of live video streaming application is difficult, because live video feeds do not provide the level of consistency required for performance evaluation of different codecs. Therefore, in these experiments we use standard test sequences offered by the video processing community. Two of the widely used test sequences are selected for the tests; they are Akiyo[1] and Coastguard[2]. Both have 300 video frames and are at QCIF resolution, but they are representing two different types of video datasets. Akiyo is darker and has only limited movement whereas Coastguard is brighter with much more movement.

Fig. 6 shows the encoding time of different codecs (i.e., MJPEG, MPEG4 and H.264) and encoding methods (i.e, intraframe and interframe encoding) for the two test video sequences performed on the Samsung Galaxy S2. It needs to be noted that MJPEG only uses intraframe encoding technique.

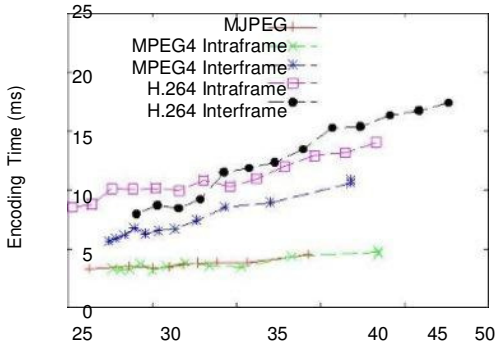
Fig. 7 shows the compressed size of the video frames at the corresponding quality using different codecs and encoding techniques. As shown in Fig. 7(a), in the case of the Akiyo test sequences (with limited motion) using interframe encoding achieves higher compression ratio than the intraframe encoding, without a large increase in the encoding time (as shown in Fig. 6(a)). When much more motion is introduced as in the Coastguard test sequences (as shown in Figs.6(b) and 7(b)), there is a significant increase in encoding time when using interframe encoding to achieve the similar degree of compression ratio. In addition, we observe that for Samsung Galaxy S2 the encoding time in both test sequences is below 20ms. This encoding time is well under the maximum encoding time of approximately 66 ms to produce video stream at 15 fps. The authors in [8] suggest that the perceived quality of a video stream is based on both the quality of the individual frames and the frame rate of the sequence. They further discover that video stream with a low frame rate of 5 fps is considered as fair quality; 10 fps (which require no more than 100 ms encoding time) is considered as good quality video streams and perceived quality tends not to get better at 15 fps or above. Based on these figures, we are certain that our video streaming application can produce good quality video stream (even with45 dB PSNR) if using the Samsung Galaxy S2.

Having tested our application on newer generation phones, we repeat the same set of experiments for the other two models of mobile phones. Fig. 8 shows a comparison in encoding time for different models of mobile phones using the aforementioned codecs and encoding techniques. There is no surprise to see that the HTC Dream, which is the first generation of Android phones with a much lower resource profile, performs much worse than the Samsung Galaxy S2. In the worst case when using H.264 with intraframe encoding it takes up to around 100 ms to encode a video frame. However, according to [8] 100 ms encoding time should be able to produce video streams at around 10 fps, which is still a fairly good quality of a video stream. Fig. 8 also shows that we can reduce the encoding time by using other codecs or encoding techniques, even on the first generation of Android phones.

We also perform experiments to measure the decoding time on these phones using different codecs and decoding techniques. The results show consistency with the encoding time experiments. In addition, we notice that the decoding time (less than 15 ms for HTC Dream in the worst case and around 2.7 ms for Samsung Galaxy S2) is much less than the encoding time in general.



(a) Akiyo test sequences



(b) Coastguard test sequences

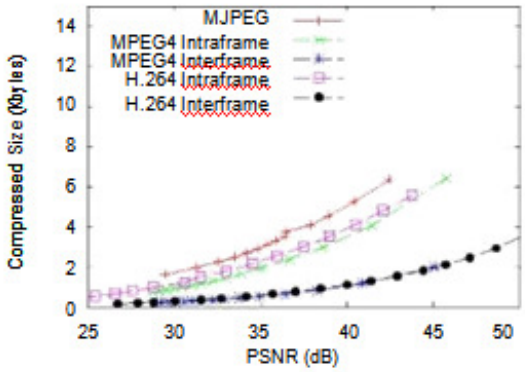
Fig. 6. Encoding time for different PSNR

4.3 Performance of Multi-Hops Forwarding

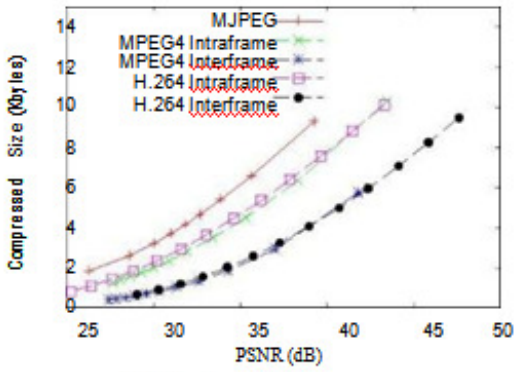
Video encoding/decoding is an important aspect that can have direct impact on the perceived quality of a video stream. However, to stream live video across multiple wireless hops, the multi-hops forwarding capacity is also

crucial to ensure that sufficient bandwidth is available for the application.

To study the multi-hop forwarding capacity of the selected phones, we assemble a testbed using Samsung Galaxy S2 as the sender and the receiver, and other phones as the forwarders in the middle of a chain topology, as shown in Fig. 9. Phones are incrementally added into the testbed to extend the chain topology; that is, we use two Samsung Galaxy S2 to form the single hop topology, and we add a Nexus One in the middle to form the two hops topology (as shown in Fig. 9(b)). During each experiment all nodes are placed in an RF shielding enclosure, which provides 85 dB of isolation from the external RF sources. Each node is connected to its next hop neighbors (via static routing) in a chain topology, which is enforced by the MAC address filtering. The experiment uses IEEE 802.11g, because of its wide availability. IEEE 802.11n is automatically selected by the phones that support this standard.



(a) Akiyo test sequences



(b) Coastguard test sequences

Fig. 7. Compressed size for different PSNR

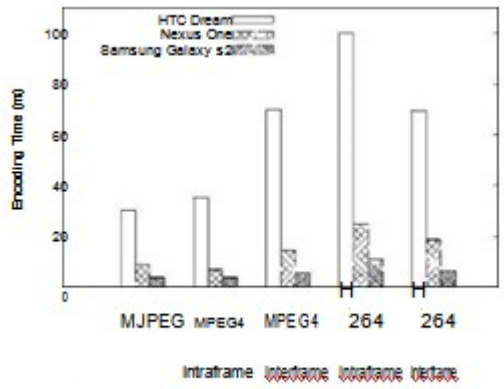
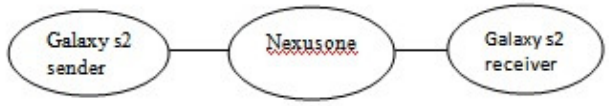


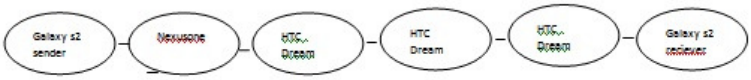
Fig. 8. Comparison of encoding time using Akiyo sequence

However, to stream live video across multiple wireless hops, the multi-hops forwarding capacity is also crucial to ensure that sufficient bandwidth is available for the application.

To study the multi-hop forwarding capacity of the selected phones, we assemble a testbed using Samsung Galaxy S2 as the sender and the receiver, and other phones as the forwarders in the middle of a chain topology, as shown in Fig. 9. Phones are incrementally added into the testbed to extend the chain topology; that is, we use two Samsung Galaxy S2 to form the single hop topology, and we add a Nexus One in the middle to form the two hops topology (as shown in Fig. 9(b)). During each experiment all nodes are placed in an RF shielding enclosure, which provides 85 dB of isolation from the external RF sources.



(a) two hops setup



(b) five hops setup

Fig. 9. Testbed for multi-hops experiments

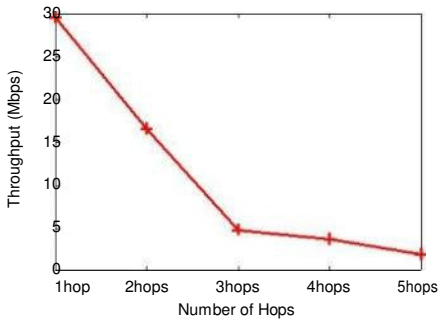


Fig. 10. Throughput achieved over multiple hops

Each node is connected to its next hop neighbors (via static routing) in a chain topology, which is enforced by the MAC address filtering. The experiment uses IEEE 802.11g, because of its wide availability. IEEE 802.11n is automatically selected by the phones that support this standard.

Fig. 10 shows the throughput achieved over different hops. As we can see the throughput drops almost half for every increase in the hop count (still achieving 1.9 Mbps over five-hops), except the case when the hop count increases from two-hops to three-hops (i.e., throughput drops from 16.6 to 4.8 Mbps). Our hypothesis is that this is due to the use of a different network standard. In the single hop and two-hops topologies every node supports IEEE 802.11n standard, therefore higher throughput is achieved. When we extend the topology to three-hops by adding a HTC Dream (only supports IEEE 802.11g network), this older generation phone becomes the bottleneck of the whole network. To verify this is happening, we construct a chain topology using only this model of phones. The results seem to confirm our assumption (achieving 4.5 Mbps over single hop, 2.15 Mbps over two-hops, and 1.11 Mbps over three-hops).

4.4 Overall Performance: Live streaming in Action

Having investigated the performance of video coding and the capacity of multi-hops forwarding on the selected Android phones, we conduct a number of multi-hop experiments by deploying our application on the phones and streaming live video (either using the test sequences or video feed captured by the camera) to evaluate the overall performance and to validate the feasibility of peer-to-peer video streaming on Android phones. For these experiments we use the multi-hop chain topology described earlier. We still use the Samsung

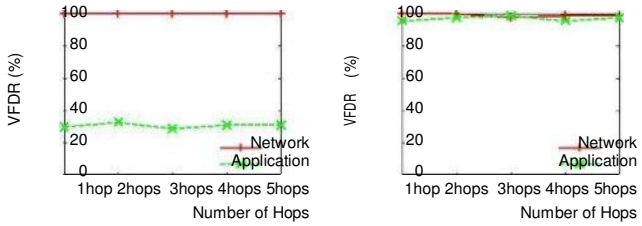


Fig. 11. Overall performance of multi-hops video streaming

Galaxy S2 as the sender and receiver to minimize the coding time, which in turn reduces the impact of bottleneck problem due to processing capability and allows more video frames to be sent over the network. All video data is encoded using MPEG-4 codec with Intraframe encoding.

Fig. 11 shows the Video Frame Delivery Ratio (VFDR) achieved over different number of hops for the Akiyo test sequences and live camera feed. We measure the VFDR at two layers: (i) at the network layer - it is the number of video frames that have been successfully received by the receiver, and (ii) at the application layer - the VFDR is the number of video frames that are received by the application and displayed successfully on the screen. It should be noted that the application usually maintains a fixed size (configurable, but fairly small) queue to buffer the received frames. When the phones can not draw the video frames to the screen fast enough, this queue can become full and starts to discard incoming video frames.

As shown in Fig. 11 the network VFDR in both cases achieves around 100% even over five-hops. This is because the required bit rate for streaming such video feeds over five-hops is around 500-600 Kbps. The required bit rate is a lot smaller than the throughput (achieving 1.9 Mbps over five-hops) we measured in the previous experiments.

We observe that the application VFDR are around 30% over different hops for the Akiyo test sequences. Further investigation discovers that because the individual video frames in the Akiyo test sequence are very small, the encoding time is therefore short. This means the application is sending out video frames at very high rate (at approximately 40 fps, frames are arriving every 25 ms). It should be noted that the frame rate of 40 fps is faster than it is required by a reasonable quality video streaming. The loss in the application layer indicates the devices are receiving more video frames than they can decode and draw on the screen (drawing video frame on the screen alone takes around 30 ms), therefore a lot of video frames have been discarded. These results show how the

application performs under high load. After the above extreme scenario with the test sequences, we evaluate our application using live camera feeds. Fig. 11(b) shows that the application VFDR achieves around 98%.

5. Conclusion and Future Work

In this paper, we presented a wireless multi-hop video streaming application for Android mobile phones. This application allows users to capture live video feeds using the mobile phone camera and to share these feeds with people who might be multiple wireless hops away. The video feeds are shared using wireless client mesh network (ad hoc network) established between mobile phones. Thus the video streaming does not rely on a traditional network infrastructure (such as the cellular), therefore it is a free-of-charge communication. Such a multi-hop video streaming can be used in a variety of application domains including social networking.

We presented an evaluation of the prototype application and demonstrated the feasibility of the multi-hop video on three generations of Android phones (with different resource capabilities). We showed that even after five wireless hops, our application still can handle video streams with high quality.

For future work we are planning to extend the evaluation testbed to study the application performance within a larger network. We are also considering developing a richer user interface with additional features, such as implementing multicast over multiple hops and allowing users to record video contents on local SD cards while streaming or forwarding.

References

[1] ITU. Statistics on Global mobile subscriptions. http://www.itu.int/newsroom/press_release/2010/06.html.

[2] C. Quick.(2009) With smart phone adoption on the rise, opportunity for marketers is callinghttp://blog.nielsen.com/nielsenwire/online mobile/with-smart phone - adoption - on – the – rise – opportunity -for marketers-is-calling/.

[3] H. L. Cycon, T. C. Schmidt, G. Hege, M. Wahlisch, D. Marpe, and M. Palkow, "Peer-to-peer videoconferencing with h.264 software codec for mobiles," in *Proc. Int. Symp. a World of Wireless, Mobileand Multimedia Networks WoWMoM 2008*, 2008, pp. 1–6.

[4] Qik. www.qik.com.

[5] Android development sdk. http://developer.android.com.

[6] Optimized link state routing protocol (olsr). http://www.ietf.org/rfc/rfc3626.txt.

[7] P. Hu, W. L. Tan, R. Wishart, M. Portmann, and J. Indulska, "Meshvision: an adaptive wireless mesh network video surveillance system," *Multimedia Systems*, vol. 16, pp. 243–254, 2010, 10.1007/s00530-010-0191-z.

[8] Q. Huynh-Thu and M. Ghanbari, "Temporal aspect of perceived quality in mobile video broadcasting," *Broadcasting, IEEE Transactions onDOI - 10.1109/TBC.2008.2001246*, vol. 54, no. 3, pp. 641–651, 2008.