International Journal of Research (IJR) Vol-1, Issue-10 November 2014 **ISSN 2348-6848**

Mobile Social TV Streaming in MANET

Chandu DelhiPolice & T. Rajendra Prasad

Email: chandudelhi@gmail.com

Abstract:

Mobile ad hoc networks are prone by nature to path breaks and reconnections. Routing protocols such as OLSR that provide topology information with a minimum delay to the quick reconfiguration of path breaks are desired. Applications and protocols should be able to adapt to the dynamics of these networks. However, this is not true for most applications and protocols. For example, a video streaming server does not become aware of the path break because it does not interact with lower layers that inform about this event. As a result, important resources such as battery and bandwidth are not used efficiently, and TCP connection failures happen frequently. In this paper, we present a proxy based solution to detect path breaks and reconnections using the proactive OLSR protocol for ongoing streaming sessions and we do corrective actions at the application layer to lead a more efficient usage of the bandwidth during disconnections.

Keywords: WiFi ad hoc networks, streaming the video, save the bandwidth and battery, Frames lost, OLSR.

I. I NTRODUCTION

A D HOC NETWORKS are multi hop wireless networks without a preinstalled infrastructure. They can be de-played instantly in situations where infrastructure is unavailable (e.g., disaster recovery), or where infrastructure is difficult to install (e.g., battlefields). It is maturing as a means to provide ubiquitous unmetered communication. With the increase both in the bandwidth of wireless channels and in the computing power of mobile devices, it is expected that video service will be offered over ad hoc networks in the near future.

Ad hoc networks pose a great challenge to video transport. There is no fixed infrastructure and the topology is frequently changing due to node mobility. Therefore, links are continuously established and broken. The availability and quality of a link further fluctuates due to channel fading and interference from other transmitting users. In

addition, an end-to-end path consists of a number of wireless links. Thus, transmission loss in ad hoc networks is more frequent than that in wireless networks with single-hop wireless paths connecting nodes to the wire line infrastructure. The most popular media access control (MAC) scheme, the carrier sensing multiple access/collision avoidance (CSMA/CA) schemes [1], is designed for best-effort data. It provides no hard guarantees for a session's bandwidth and delay. Although bandwidth reservation is possible with MAC schemes based on time-division multiple-access (TDMA) or code-division multiple-access (CDMA), practical implementations of these schemes are nontrivial because of the synchronization or code assignment problems when node mobility is allowed [2].

Video transport typically requires stringent bandwidth and delay guarantees. However, it is very hard to maintain an end-to-end route, which is both stable and has enough bandwidth in an ad hoc network. Furthermore, compressed video is susceptible to transmission errors. For example, a single bit error often causes a loss of synchronization when variable length coding (VLC) is used. Moreover, the motion compensated prediction (MCP) technique is widely used in modern video coding standards. In MCP, a frame is first predicted from a previous coded frame (called reference picture) and then the prediction error is encoded and transmitted. Although MCP achieves high coding efficiency by exploiting the temporal correlation between adjacent frames, it makes the reconstruction of a frame depend on the successful reconstruction of its reference picture. Without effective error protection and concealment, a lost packet in a frame can cause not only error within this frame, but also errors in many following frames, even when all the following frames are correctly received [3].

Given the error-prone nature of ad hoc network paths and the susceptibility of compressed video to transmission errors, effective error control is needed. Traditional techniques, including forward error correction (FEC) and automatic repeat request (ARQ), must be adapted to take into consideration the delay constraint and the error propagation problem [4]. In ad hoc networks, wireless links break down the traditional concept of topology, which is not constrained by physical cable connections anymore. Although user mobility makes links volatile, it provides variability of

topology. On the one hand, a link may break when nodes move away from each other. On the other hand, it is possible to quickly find new routes formed in a new topology. Furthermore, the mesh topology of ad hoc networks implies the existence of multiple routes between two nodes. Given multiple paths, a video stream can be divided into multiple sub streams and each sub stream is sent on one of the paths. If these paths are disjoint, the losses experienced by the sub streams would be relatively independent. Therefore, better error resilience can be achieved when traffic dispersion is performed appropriately and with effective error control for the sub streams. In a manner similar to multiantenna diversity that improves the capacity of wireless networks, path diversity can also be exploited to improve the capacity of ad hoc networks. Indeed, multipath transport (MPT) provides an extra degree of freedom in designing video coding and transport schemes.

In this paper, we propose three MCP-based video transport techniques for mobile ad hoc networks. These schemes take advantage of path diversity to achieve better performance. Compared with the coding methods considered in our previous work [5], these techniques can achieve significantly higher coding efficiency and are compliant with the H.26X and MPEG series standards (possibly with simple modifications). The techniques that we have examined include:

1) A feedback based reference picture selection scheme (RPS) [6];

2) A layered coding (LC) with selective ARQ scheme (LC with ARQ) [7];

3) A multiple description motion compensation coding scheme (MDMC) [8].

We studied the performance of these three schemes via a top-down approach. First, we used a popular Markov link model [6], [9], where lower layer detail is embodied in the bur sty errors generated. This simple model enables us to examine the system performance over a wide range of pack loss rates and loss patterns. Next, lower layer details, including user mobility, multipath routing, and the MAC layer are taken into account in the OPNET simulations [10], which provide a more realistic view of the impact of these factors on the system performance. Furthermore, we implemented an ad hoc video streaming test bed using notebook computers with IEEE 802.11b cards. This further validates the viability and performance advantages of these schemes.

The results of our experiments show that video transport is viable in ad hoc networks given careful cross-layer design. Combining multi stream coding with MPT improves video quality, as compared with traditional schemes where a single path is used. Each of these three techniques is best suited for a particular environment, depending on the availability of feedback channels, the end-to-end delay constraint, and the error characteristics of the paths.

The remainder of the paper is organized as follows. In Section II, we present the general architecture of multi stream video coding with MPT.

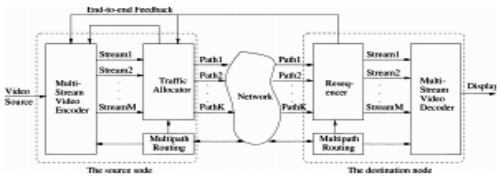

Figure 1: General architecture of the proposed system using multi stream coding and multi path transport.

2. DETECTING CLIENT'S AVAILABILITY AND UDP SERVICES ANNOUNCEMENT

Both tasks are done injecting and capturing OLSR packets type 200. Whenever the communication between the client and the server is possible (directly or via one or more intermediate nodes), the proxy server receives periodically OLSR packets type 200 from the client agent and vice versa. If the proxy server does not receive at least one of this kind of packet for a while (1 second by default although this timeout can be modified by the user when the proxy is started), this is indicative that the client is disconnected. Similarly, if the client agent does not receive a packet OLSR type 200 from the proxy server (after 1 second by default, also configurable), it becomes aware of the disconnection. The reconnection is detected by the proxy server and the client agent when they receive at least one packet OLSR type 200 from the other one during an interval of 1 second. Appropriate actions are done on both peers when the disconnection or the reconnection is detected and they depend on the type of streaming protocol used. The chosen timeout (1 second in our experiments) is a key parameter because a high value could lead a high delay to detect the disconnection whereas a low value could trigger false alarms, i.e., no packet is received because of network congestion but the proxy server or the client agent wrongly detect a disconnection.

The packet size injected by the proxy server and the client agent is 206 and 66 bytes respectively. The difference in size is due to the message content differs. For example, the proxy server is concerned about broadcasting the type of server (e.g. Video Lane [11]), its IP address, the proxy server's name to the client agent can distinguish among different proxies, a command used by the TCP Control mechanism, the play listing using RSTP, and HTTP and UDP ports avail-able to serve streams. As you can see, the broad-casting of the own OLSR packets type 200 is also the mechanism used to announce the UDP services. The message content is useful since the client agent shows it to the user via a Graphical Unit Interface (GUI) and then it launches the user's player with the appropriate command according to the user's election. The content of the packet injected by the client agent includes its IP address, the type of player and requests of UDP streams.

3. CORRECTIVE ACTIONS

Table 1 summarizes the actions that the proxies do when they detect disconnections and reconnections (in brackets it is shown the process that does the action), and the benefits of these actions. Fig.3 and Fig.4 show the proxies behavior during disconnections and reconnections for RTSP, and HTTP or RTP respectively. Irrespective of the streaming protocol, the client agent

Table 1 Corrective action during disconnections and reconnections.

	Disconnection	Reconnection	Total disconnection	Lost frames	Abrupt ending	Batt. saving	BW saving
RTSP	Pause the server (PS), warning the user (AC) and close TCP connection (PS,AC)	Create TCP connection (PS,AC), resume the server (PS) and warning the user (AC)	End session (PS)	Yes (live video), No (VoD)	No	Yes	Yes
HTTP	Freeze frames forwarding from PS to AC, close TCP connection (PS,AC) and warning the user (AC)	Create TCP connection (PS,AC), resume frames forwarding (PS) and warning the user (AC)	End session (PS)	Yes	No	No*	Yes
RTP	Freeze frames forwarding from PS to AC and warning the user (AC)	Resume frames forwarding (PS) and warning the user (AC)	End session (PS)	Yes	–	No*	Yes

PS: Proxy Server **AC**: Agent Client **BW**: Bandwidth
*The server is still sending frames but PS does not inject them into the MANET

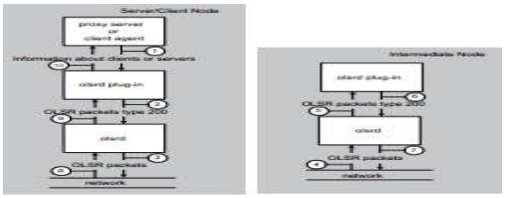

Figure 2: Information passing between proxies.

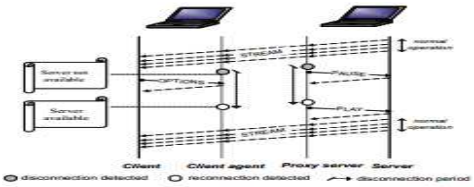

Figure 3: Proxies behavior during disconnections and reconnections for a RTSP session.

Starts a warning message box on the user's screen when a disconnection or a reconnection is detected and the proxy server ends the session when the disconnection exceeds a period of time. For the streaming protocols built on top of TCP

(RTSP and HTTP), the TCP connection between the proxy server and the client agent is closed when a disconnection happens and a new one is created after the reconnection. Using RTSP compliant commands such as pause and play, the proxy server pauses or resumes the server. For HTTP or RTP, the server is not paused during the disconnection period but the frames are not for warded from the proxy server to the agent client to save bandwidth.

4. THE TCP CONTROL MECHANISM

When establishing one TCP connection for a RTSP or HTTP session, three TCP connections are created: one between the client and the client agent, another to communicate the client agent and the proxy server, and the last one for the communication between the proxy server and the server. The former and the latter are local connections and they are not likely to fail during disconnections. The mechanism is as follows (Fig.5):

Identify the TCP connection. After the second of the three TCP connections is created, the client agent requests a unique identifier for this TCP connection by sending a TCP Control: 0 command to the proxy server using the newly TCP connection. The proxy server responds to the client agent with a TCP Control: id where ID is the identifier, a value different for any client agent and the proxy server.

- Session communication. The HTTP or RTSP session begins normal communication.
- Disconnection detected. If the client is out of coverage, then the client agent do the following actions:

  1. Close the TCP connection with the proxy server.
  2. Detect the server's availability.
  3. Create a new TCP connection with the proxy server.
  4. Send the TCP Control: id command to the proxy server to replace the old TCP connection with the last one. The proxy server sends TCP Control: OK as acknowledgement. These two commands are communicated using the new TCP connection.

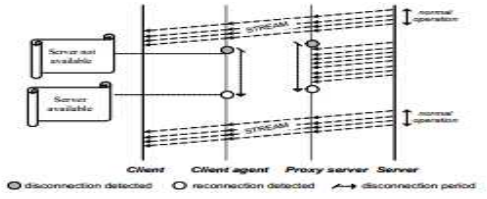

Figure 4: Proxies behavior during disconnections and reconnections for a HTTP or RTP session.

If the server is out of coverage, it must order all the client agents to close the TCP connections and create new TCP connections. For doing that, the proxy server broadcasts a message using the packet OLSR type 200 including in the message's content of the packet the command TCP Fall to convey all the client agents that they must do the steps 1 to 4 described in the phase Disconnection detected.

- End sessions with clients disconnected for a long time. After a timeout, the server frees the software resources reserved for serving the client and the session ends. Any later attempt to recover the TCP connection will fail (command TCP Control: ERROR) and in that case the client agent will warn the user that the session has ended.

5. EXPERIMENTAL TESTS

We tested the behavior and performance of our soft-ware architecture using the topology described in Fig.1.a.We are concerned in presenting results that show: a) the low overhead of the packets OLSR including our packets type 200 in comparison with the data volume of the streaming, b) the low usage of CPU and battery consumption due to the proxies, c) the de-lay and jitter because of the proxies, and d) the benefits of using our corrective actions and the TCP connections management between the proxy server and the client agent.

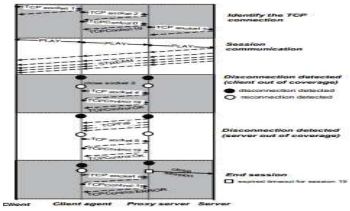

Figure 5: TCP Control mechanism.

The plug-ins and the proxies were programmed using C and C++ languages respectively for Windows operating system. The server was installed on a Pentium IV at 2.8GHz with 512 MB and 802.11b compliant. The intermediate node was a Centurion at 1.6GHz, 512MB and 802.11b/g. The client node was a Celeron 1.4GHz, 1024MB and with an 802.11b/g wireless interface. All the nodes were located in the same room and we added mobility to the network by allowing the client node to be within radio range of the server node via the intermediate node and also we moved the client and the server to

make each other out of coverage to test the corrective actions made by proxies and the TCP connections management. We used VLC media player [11], a free cross-platform media player that supports a large number of multimedia formats and it is available for several operating systems, it needs little CPU power and it can be used as a streaming server to stream in unicast or multicast in IPv4 or IPv6. We used also VLC for serving the video in unicast in IPv4.

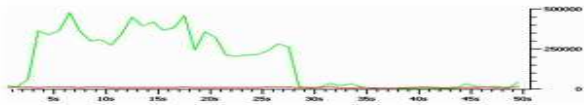

Figure 6: Traffic OLSR (red curve) and RTSP traffic (green curve).

5.1 OLSR Traffic

Fig.6 shows the bytes per second for a RTSP session (green curve) and the traffic due to OLSR protocol including our own packets type 200 (red curve) obtained using Ethereal tool [13]. As you can see, the overhead due to OLSR is negligible in comparison with the traffic of the ongoing streaming session.

5.2 CPU and Battery Consumption

In the experiment, the server sends streams at a rate of 2400Kbps. It was measured the CPU percentage consumed by the proxy server and the client agent for different streaming protocols varying the number of sessions from 1 to 10 (Fig.7.a to c). During the experiment, the processes running on the server were the proxy server, the server and olsrd, whereas on the client were running the client agent, the player and olsrd. As you can see, the proxies need little CPU power being something higher for RTP sessions on the client agent in comparison with the same session on the proxy server because for a new RTP session the client agent instantiates new objects and threads that increase the necessary CPU power. In general, the differences of CPU usage among the different streaming protocols are due to different software resources necessary on both proxies to manage the sessions.

Fig.7.d shows the battery consumption on the client and server nodes from 1 to 5 ongoing RTSP sessions with and without proxies. As yo can see, same values are obtained for all the experiments irrespective of the proxy's usage or not. The battery consumption on the client node is higher than on the server node due to the CPU usage of the player to uncompress and play the stream.

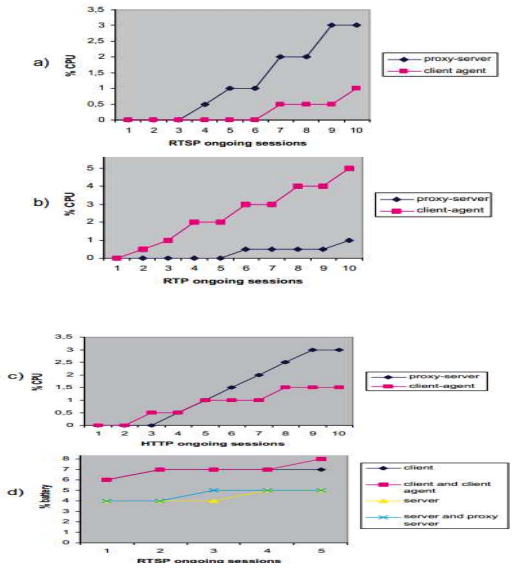

Figure 7: CPU consumption for: RTSP (a), RTP (b) and HTTP sessions (c). Battery consumption for RTSP sessions (d).

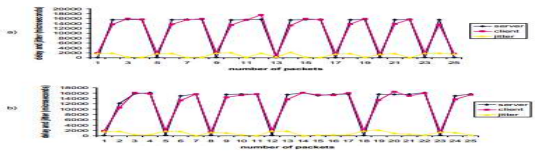

Figure 8: Delay and jitter for 25 samples. Without proxies (a), With proxies (b).

5.3 Delay and Jitter

Fig.8.a shows the delay in microseconds to transmit from 1 to 25 consecutive packets from the server to the client when only olsrd, the server and the player are running. It is also presented the jitter. The average jitter is 1.13608 ms, a viable value for the streaming transmission. Fig.8.b shows the same results when also the proxies are running. In this case, the average jitter is lower (0.94224 ms).

5.4 Corrective Actions

Fig.9.a presents the number of packets per second in-jected by the server during a VoD RTSP streaming session at a rate of 2000Kbps (green curve). The red curve is the OLSR traffic transmitted by the client node. We forced a disconnection period of 8s (about the 8th second to the 16th second). During this time interval, Ethereal tool did not capture OLSR traf-fic since the client is out of coverage but it captured RTSP traffic transmitted by the server since the server is not aware of the disconnection period (no proxies were used for this experiment). As a result, about 2MB are transmitted using RTP protocol and lost since we don't use a proxy server to pause the server. To correct this inefficient usage of the server and the available wireless bandwidth, we repeated the experi-ment using the proxies and we forced a higher discon-nection period of 35s (Fig.9.b). During this period, the server is paused and no frames are transmitted

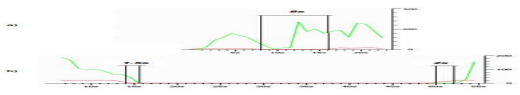

Figure 9: Bandwidth usage: Without proxies (a), With proxies (b).

avoiding that the server injects a total of 8.75 MB. As you can see, the proxy server lasts about 1.5s to detect the disconnection and about 2s to detect the reconnection. Both values are a bit higher to the the oretical value of 1 second we fix to warn the proxy about a disconnection or reconnection. Since we use a proactive protocol to detect them, the detection time is even lower that the one we would obtain using reactive protocols such asAd hoc On-demand Distance Vector Routing(AODV) or Dynamic Source Routing (DSR) [14]. Similar results were obtained for RTP and HTTP sessions.

6. CONCLUSIONS

Enabling video transport over ad hoc networks is more challenging than over other wireless networks, both because ad hoc paths are highly unstable and compressed video is susceptible to transmission errors. However, multiple paths in an ad hoc network can be exploited as an effective means to combat transmission errors. Motivated by this observation, we chose three representative video coding schemes, all based on MCP used in modern video coding standards, and show how to adapt these schemes with MPT to enable video transport over ad hoc networks.

7.REFERENCES

[1] Draft Standard for Wireless LAN: Medium Access Control (MAC) and Physical Layer (PHY) Specifications IEEE , IEEE Standard 802.11, July 1996.

[2] C. R. Lin and M. Gerla, “Asynchronous multimedia multihop wireless networks,” in Proc. IEEE/ACM INFOCOM, Kobe, Japan, Apr. 1997, pp.118–125.

[3] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, “Error resilient video coding techniques,” IEEE Signal Processing Mag., vol. 17, pp.61–82, July 2000.

[4] Y. Wang and Q.-F. Zhu, “Error control and concealment for video communication: A review,”Proc. IEEE , vol. 86, no. 5, pp. 974–997, May 1998.

[5] N. Gogate, D. Chung, S. S. Panwar, and Y. Wang, “Supporting image/video applications in a multihop radio environment using route diversity and multiple description coding,” IEEE Trans. Circuit Syst. Video Technol. , vol. 12, no. 9, pp. 777–792, Sept. 2002.

[6] S. Lin, S. Mao, Y. Wang, and S. S. Panwar, "A reference picture selection scheme for video transmission over ad hoc networks using multiple paths," inProc. IEEE ICME , Tokyo, Japan, Aug. 2001,pp. 96–99.

[7] S. Mao, S. Lin, S. S. Panwar, and Y. Wang, "Reliable transmission of video over ad hoc networks using automatic repeat request and multipath transport," in Proc. IEEE Fall VTC , Atlantic City, NJ, Oct. 2001, pp.615–619.

[8] Y. Wang and S. Lin, "Error resilient video coding using multiple description motion compensation," IEEE Trans. Circuits Syst. Video Technol ,vol. 12, no. 6, pp. 438–452, 2002.

[9] E. N. Gilbert, "Capacity of a bursty-noise channel," Bell Syst. Tech. J.,vol. 39, no. 9, pp. 1253–1265, Sept. 1960.

[10] OPNET Modeler. OPNET Tech., Inc. [Online]. Available: http://www.mil3.com

[11] VideoLAN - Free Software and Open Source Video Streaming Solution for Every OS!,http://www.videolan.org/.

[12] Clausen, T.H., Jacquet, P., Optimized Link State Routing Protocol, RFC3626,Internet Engineering Task Force (IETF),http://ietf.org/rfc/rfc3626.txt, October 2003.

[13] Ethereal: A Network Protocol Analyzer, www.ethereal.com.

[14] Siva Ram Murthy, C., Manoj, B.S.Ad Hoc Wireless Networks. Architectures and Protocols, Prentice Hall PTR, 2004.