Corrective Actions at the Application Level for Streaming Video in WiFi Ad Hoc Networks

E. M. Macías, A. Suárez, and J. Martín
Grupo de Arquitectura y Concurrencia, Departamento de Ingeniería Telemática
Universidad de Las Palmas de G. C.
Campus Universitario de Tafira, 35017 Las Palmas de G. C. (Spain)

***Abstract*—Efficient video streaming in a mobile ad hoc network (MANET) is a challenging problem due to the dynamic nature of the network that leads to high bit error rates, unpredictable delay, jitter, throughput and packet delivery ratios and frequent short, intermittent and long-term link failures. Despite the MANET research community's efforts, there are still open problems. For example, protocols and mechanisms that hide these issues to the video streaming applications users are still in early stages. However, these applications must tolerate transparently the dynamic behavior of the network and be able to progress in presence of disconnections. In practice, this is the exception rather the rule. In this paper, we present a multilayer cooperative solution to detect disconnections and reconnections between a video streaming server and a client and we propose corrective actions at the application level. With our transparent approach to the user, the video streaming sessions can tolerate frequent long and short disconnections and use more efficiently the shared wireless bandwidth.**

I. Introduction

Achieving multimedia communications over MANET pose many challenges [1] and profitable business [2]. Video streaming is a very useful technique for devices with low storage capacity such as mobile phones and *Personal Digital Assistants* (*PDA*) that use cellular [3], WiFi [4] or WiMax [5] wireless communication technologies. Among others, the following situations degrade the video streaming performance on MANET: i) interruption in packet delivery when a link breaks (e.g. sender, receiver or intermediate node goes out of coverage or their batteries go down), ii) efficient alternative path discovering without degrading jitter [6], iii) high error rates due to the multipath fading, iv) limited bandwidth combined with variable network latency [7]. The efficient election of the transport or application level protocol for video streaming and the efficient control of intermittent wireless channel disruption are also very important issues.

Usually *User Datagram Protocol* (*UDP*) is used to transmit live streaming video [8]. Over UDP, the couple *Real-time Transport Protocol* (*RTP*) and *Real-time Transport Control Protocol* (*RTCP*) is used for real time streaming video [9]. The server continuously sends frames and the client usually does not pause the streaming. A path break implies the server continues with the frame transmission but the client will not receive any frame (data is lost and the bandwidth and battery are not used efficiently).

The persistent version of *HiperText Transfer Protocol* (*HTTP*) can support streaming for *Video on Demand* (*VoD*) so a client sends a request and gets a response, and then sends additional requests and gets additional responses without *Transmission Control Protocol* (*TCP*) connection release. *Real Time Streaming Protocol* (*RTSP*) can use any of the above protocols for transmitting video data and TCP client commands to control the user session on the server.

Although TCP reliability mechanism will retransmit the missing data, the TCP socket will become invalid to the server or the client if an abort occurs and with high probability the user session will end abruptly. Aborts primarily occur when data goes unacknowledged for a period of time that exceeds the limits on retransmission defined by TCP. Other causes for an abort include a request by the application, too many unacknowledged TCP keepalive probes, receipt of a TCP reset packet and some types of network failures reported by the IP layer. We do not consider the improved versions of TCP explained in [10] because of they require: modifications to existing TCP (e.g *TCP-F* and *split-TCP*), more bandwidth and power consumption during a path failure (*TCP-ELFN*), dependency on a particular routing protocol to improve its performance (*TCP-Bus*), addition of layers to the TCP/IP protocol stack (*ATCP*). We do not also consider the protocols overviewed in [11] due to their performance is not well enough [12].

Cross-layer techniques have been applied to solve the above challenges, for example in [13] it is proposed the adaptation of the retry limit parameter at the 802.11 *Medium Access Control* (*MAC*) level to avoid triggering the TCP congestion control mechanism during short-term link failures (it is not appropriated for long-term disruptions). We consider a multilayer cooperative solution: a particular efficient network routing algorithm, any of the above transport protocols, a mechanism to support short and long-term disruptions for TCP based connections, and an application level software that implements the corrective actions when appropriate to robustly tolerate short and long-term disruptions. In order to consider any kind of video streaming client and server we implement a client agent and a proxy server. In this way we

Research partially supported by the Spanish CICYT (MEC) and European Regional Development Fund (FEDER) under Contract TSI2005-07764-C02-01 and The Canaries Regional Education, Cultural and Sports Ministry and FEDER under Contract PI042004/164.

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 525–530.

tested the good performance of our multi-protocol and multi-client and server solution for *WiFi* [14] ad hoc networks.

The rest of the paper is organized as follows: section 2 is devoted to discuss the related work. Section 3 reviews the software architecture. Section 4 presents the corrective actions. In section 5 we describe some experimental results. Finally, concluding remarks are summarized in section 6.

II. Related Work

The distributed and self-organizing nature of a MANET stems from having a routing protocol installed in each wireless node. The major routing protocols for MANET are classified into *on-demand* or *reactive* and *proactive* routing algorithms. The former initiate route discovery only after a path breaks incurring a high cost to establish a new route whereas the latter initiate route discovery early and before the path breaks at the cost of higher routing load.

Proactive protocols show more benefits to send video over ad hoc wireless networks than reactive protocols [15]. Due to the reactive behavior, the delay, jitter, throughput and packet delivery ratio for the communication flow may vary a lot in quantity. We use a proactive protocol named *Optimized Link State Routing Protocol* (*OLSR*) [16] that consists of: i) a neighbor sensing mechanism that detects changes in its neighborhood injecting and receiving HELLO messages periodically, ii) an efficient flooding of control traffic, i.e. OLSR packets injected into the network for the quick reconfiguration of path breaks. All nodes receive the messages and there are not duplicated messages thanks to the use of multipoint relays. This is an important property that favors its use in a wireless network which is by nature prone to mobility of nodes and collisions due to the hidden terminal problem, iii) diffusion of topological information necessary to obtain optimal routes in terms of the number of hops. This information is valid for a period of time so expired information is removed. All the traffic in OLSR is UDP and it is transmitted by broadcast or multicast on port 698.

Ref. [17] proposes a hybrid mechanism that consists of an early warning to a reactive protocol in order to initiate route discovery only when a path is likely to break. With this approach, the authors try to reduce the time to detect the disconnection and find a new path, and also reduce the routing load. The signal strength is used as the preemptive trigger. However, as the authors recognize, this physical parameter is not optimal because the value reported differs among 802.11 cards vendors [18]. Therefore, the signal strength values read from a 802.11 card should not be assumed to be particularly accurate [19].

Ref. [20] presents an architecture for detecting and diagnosing faults in IEEE 802.11 infrastructure wireless networks. One of its contributions is enabling bootstrapping and fault diagnosis of disconnected clients to report information to network administrators and support personnel. This work does not provide any support for disconnected clients during on-going video streaming sessions. On the contrary, we provide corrective actions at the application level as well as detection of disconnected clients.

References [15], [17] and [20] are concerned about providing a route between the client and the server in terms of the quick reconfiguration of paths but they are not concerned in providing solutions in the scope of video streaming sessions when a path can not be established.

Multimedia data replication at several servers in a MANET is proposed in [21]. The client establishes a connection to the nearest server once a data block has been received or it is renewed to the same server if it is still the nearest one. To our knowledge, this approach will not perform well with standard streaming protocols because a new connection for VoD implies starting the streaming from its beginning.

The factors causing the low communication quality of current WiFi ad hoc networks and its derived implications for application development is the main concern of [18]. For example, the authors state that applications must tolerate frequent disconnections and the programmers must define when a link is considered to fail but they do not provide a practical solution. We have programmed a proactive mechanism that detects when a link between the client and the server is not available (there is not an alternative route according to the routing protocol) and we do the corrective actions at the application level described in section 4.

Ref. [22] presents a proactive adaptation to the UDP-based streaming video sent by a fixed server to a mobile client connected to one 802.11b infrastructure wireless network. The adaptation consists of increasing the buffer size on the client to store more frames just before entering the low quality area (termed *trouble spot*) in the hope that the mobile client will exit the trouble spot before the buffer runs out. In this paper we consider not only UDP-based streaming video but also TCP-based. On the other hand, we consider both the client and the server mobiles. In this scenario, path breaks take place frequently and quickly so the proactive adaptation proposed in [22] could not be viable.

III. The Software Architecture

Fig.1.a. shows our target MANET consisting of any number of hops. The server node (S) communicates with the client node (C) via zero or more intermediate nodes (I).

The shaded parts in Fig. 1.b to d are the new software elements we introduce (in the protocol stack) to avoid modifying the client application, the server application and the streaming protocols:

- *olsrd* (OLSR daemon) [23] is an implementation of OLSR. OLSR routes efficiently the packets into the network according to the number of hops between the sender and the client. Due to the time-varying characteristics of the wireless links, olsrd can be configured to calculate the optimal routes defined as the number of attempts by a node on average to

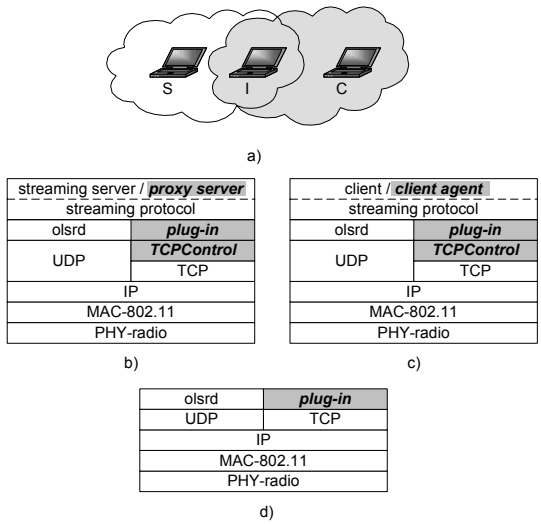

Fig. 1. A two hop MANET. Topology (a), Software architecture: server node (b), client node (c), intermediate node (d).

successfully transmit a packet to a destination, instead of the number of hops. This is a useful behavior since it is important to consider the quality of the link when choosing a path.

- The proxy running on the server and client machines are called *proxy server* and *client agent* respectively. Since both the client and the server are mobile, the proxy server is installed on the wireless node that serves the stream (S), and the client agent on the client node. The proxies are also in charge of detecting if there is, or is not, a path between the server and client to do corrective actions. These corrective actions depend on the type of video streaming being served (VoD or live video) and the type of streaming protocol used to transport the data (RTSP, HTTP or RTP) as we will show in section 4.
- OLSR lets use its optimized flooding mechanism to send information, routing related or not, from the application level using a plug-in. We just use this property to inject user defined packets (OLSR packets type 200) to control the client's and server's availability and to announce the UDP services. The plug-in on the server, client and intermediate nodes conveys to olsrd the information to be sent into OLSR packets type 200 to the MANET. The plug-in on the server and client nodes also communicates to the proxies the OLSR packets type 200 captured by olsrd from the network.

Whenever the communication between the client and the server is possible (via zero or more intermediate nodes), the proxy server receives periodically OLSR packets type 200 from the client agent and viceversa. If the proxy server does not receive at least one of this kind of packet for a while (1 second by default although configurable), 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 are detected. The optimal value for the timeout is difficult to choose: 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 detects a disconnection. The value of 1 second in our experiments gave good results.

- *TCPControl* [24] is the mechanism for transparently detecting TCP connection failures and to create a new TCP connection that avoids the streaming session release.

IV. 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. Irrespective of the streaming protocol, the client agent 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 our TCPControl mechanism. 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 forwarded from the proxy server to the client agent to save bandwidth.

V. Experimental Results

We tested the behavior and performance of our software architecture using the topology showed in Fig. 1.a. We are concerned in presenting results that show the benefits of using our corrective actions and the TCP connections management between the proxy server and the client agent. For doing that, we did several experiments that consisted of forcing client's disconnections and reconnections, and evaluating the behavior for RTSP, HTTP and RTP/UDP based streams using or not our proxies based solution. We measured the data volume and the TCP disconnections during a disconnection and show how this is solved with our approach.

TABLE I
CORRECTIVE ACTIONS DURING DISCONNECTIONS AND RECONNECTIONS

Action vs. Protocol	**RTSP**	**HTTP**	**RTP**
Disconnection	Pause the server (PS), warning the	Freeze frames forwarding from	Freeze frames forwarding from

	user (CA), close TCP connection (PS,CA)	PS to CA, close TCP connection (PS,CA), warning the user (CA)	PS to CA, warning the user (CA)
Reconnection	Create TCP connection (PS,CA), resume the server (PS), warning the user (CA)	Create TCP connection (PS,CA), resume frames forwarding (PS), warning the user (CA)	Resume frames forwarding (PS), warning the user (CA)
Total disconnection	End session (PS)	End session (PS)	End session (PS)
Lost frames	Yes (live video), No (VoD)	Yes	Yes
Abrupt ending	No	No	--
Batt. saving	Yes	No*	No*
BW saving	Yes	Yes	Yes

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

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 2.8 GHz with 512 MB and 802.11b compliant. The intermediate node was a Centrino at 1.6 GHz, 512 MB and 802.11b/g. The client node was a Celeron 1.4 GHz, 1024 MB and with a 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 we also 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 [25], 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 unicast or multicast in IPv4 or IPv6. We used VLC for serving the video in unicast in IPv4.

A. RTSP

Fig. 2.a presents the number of packets per second injected by the server during a VoD RTSP streaming session at a rate of 2 Mbps (green curve) and the OLSR traffic transmitted by the client node including our packets type 200 (red curve). The green curve only shows the multimedia traffic using RTP protocol, i.e. RTSP commands using the TCP connection are not shown in this curve but in Fig. 2.b. We forced a disconnection period of 8 s (about the 8th second until the 16th second). During this time interval, Ethereal tool did not capture OLSR traffic (the red curve falls to 0) since the client is out of coverage. However, Ethereal captured RTP traffic transmitted by the server since the server is not aware of the disconnection period (no proxies were used for this test). As a result, about 2 MB are transmitted and lost using RTP protocol since we do not use a proxy server to pause the server.

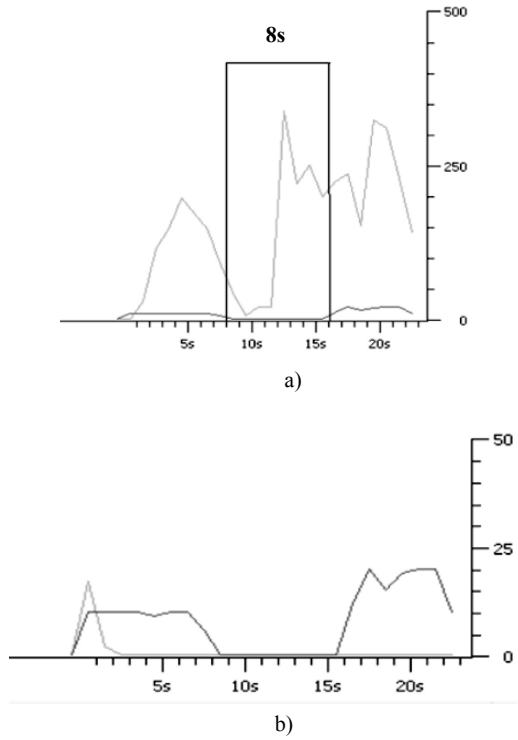

Fig. 2. Behavior during a disconnection and after the reconnection for a RTSP session without corrective actions: RTP traffic from the server (green curve) and OLSR traffic (red curve) from the client (a). TCP traffic between the client and the server (green curve) and OLSR traffic (red curve) from the client (b).

Fig. 2.b presents the number of packets per second injected by the server and the client using the TCP connection (green curve). As you can see, the TCP connection is lost during the disconnection period (again red curve shows the OLSR traffic injected by the client that falls to 0 during the disconnection period) and it is not recovered after the client's reconnection. As a result, any attempt of the client to use this TCP connection to control the streaming will fail or even the streaming session will end abruptly.

To correct this inefficient usage of the server and the available wireless bandwidth, and to avoid the lost of the TCP connection, we repeated the experiment using the proxies and we forced a higher disconnection period of 35 s (Fig. 3.a). During this period, the server is paused by the proxy server and no frames are transmitted avoiding that the server injects a total of 8.75 MB. As it is shown if Fig. 3.a, the proxy server lasts about 1.5 s to detect and react properly to the disconnection and about 2 s to detect and react to the reconnection. Both values are a bit higher to the theoretical value of 1 second we fix to warn the proxy about a

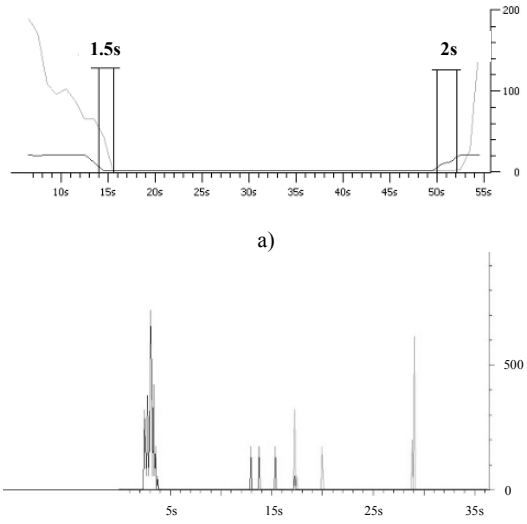

Fig. 3. Improved behavior using proxies during a disconnection and after the reconnection for a RTSP session: RTP traffic from the server (green curve) and OLSR traffic (red curve) from the client (a). TCP traffic between the client and the server (b).

disconnection or reconnection because it is included the time the proxy server needs to do a corrective action, e.g. sending a RTSP compliant pause or play command to the server. 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 as *Ad hoc On-demand Distance Vector Routing* (*AODV*) or *Dynamic Source Routing* (*DSR*) [10].

Fig. 3.b shows the behavior of the TCPControl mechanism for the RTSP session. Initially, the port used for the TCP connection between the client agent and the proxy server is 1273 (blue curve). About the second 17, the TCP connection is lost but using TCPControl a new TCP connection over port 1280 is created (violet curve) and the streaming session is resumed and not abruptly ended after the reconnection.

B. HTTP

Fig. 4.a presents the number of packets per second injected by the server during a streaming session over HTTP (green curve) and the OLSR traffic transmitted by the client (red curve). During the disconnection period, i.e. the red curve falls to 0, the HTTP based stream is not captured since the TCP connection is lost. After the reconnection, the TCP connection is not recovered so the streaming is stopped and any attempt of the reconnected client to receive the stream will fail. However, using proxies (Fig. 4.b), the TCP connection is reestablished once the client is reconnected (blue curve in Fig. 4.b) using our *TCPControl* software.

C. RTP

Fig. 5.a presents the number of packets per second injected

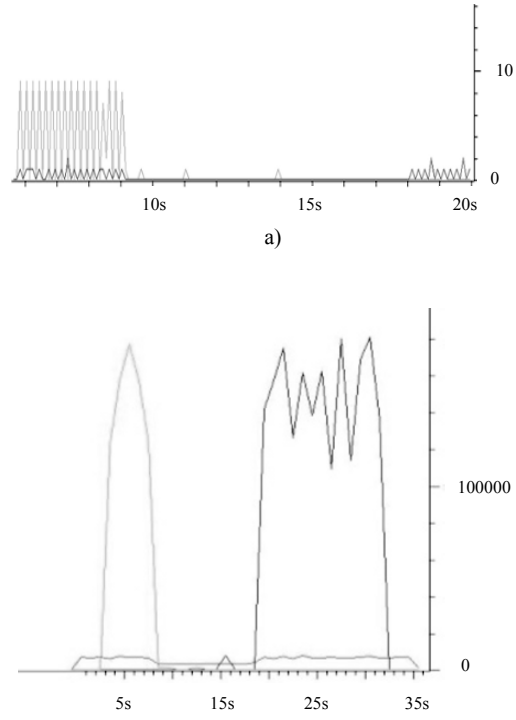

Fig. 4. HTTP based streaming: Behavior during a disconnection and after a reconnection without corrective actions (a). Improved behavior with proxies (b).

by the server during a streaming session over RTP/UDP (green curve) at a rate of 2 Mbps and the OLSR traffic transmitted by the client (red curve). We forced a disconnection period of 9s (about the 11th second to the 20th second). During this time interval, Ethereal did capture RTP/UDP traffic transmitted by the server since the server is not aware of the disconnection period. As a result, a data volume of 2.25 MB is transmitted using inefficiently the wireless bandwidth. We repeated the experiment using proxies. Fig. 5.b shows the RTP/UDP stream captured by Ethereal (green curve) and transmitted by the server, and the OLSR traffic sent by the client (red curve) both in terms of number of packets injected per second. This time, the proxy server lasts about 1.5s to detect and react properly to the disconnection and the reconnection. During the disconnection period, no RTP/UDP based stream is forwarded by the proxy server to the proxy client and as a result, bandwidth is saved.

D. Percentage of recovered TCP connections

For video streaming based on RTSP and HTTP we forced 25 disconnections between the client and the server and we studied the percentage of TCP connections recovered. This value was 92% (23 successful reconnections). For the two TCP connections lost and not recovered by our TCPControl mechanism, the server's resources allocated for these

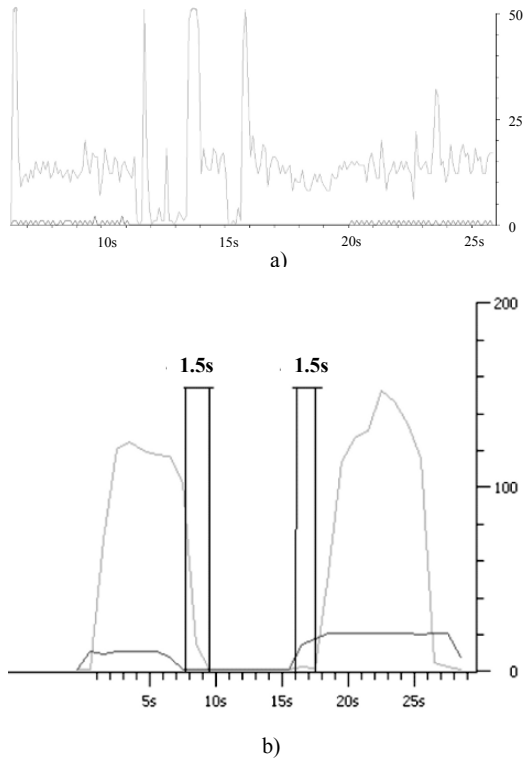

Fig. 5. RTP based streaming: Behavior during a disconnection and after a reconnection without corrective actions (a). Improved behavior with proxies (b).

streaming sessions were silently released.

VI. Conclusion

This paper discussed about the challenges that a video streaming session faced in a MANET. We proposed some corrective actions at the application layer for different streaming media protocols. This support is done by proxies that detect path breaks and reconnections thanks to the feedback provided by the proactive OLSR protocol. Experimental results showed the convenience of using our software architecture for a better use of the bandwidth and to avoid losing of frames under certain conditions. We are thinking to improve the corrective actions, e.g. store frames during disconnections for HTTP and RTP based streams and give early directions to the user to a better coverage area.

References

[1] M.D. Kwong, S. Cherkaoui, and R. Lefebvre, "Multiple description and multi-path routing for robust voice transmission over ad-hoc networks", in *2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 262-267.

[2] Y. Tashiro, Y. Yashima, and H. Fuju, "NTT's technologies for next-generation video services", *ACM Computers in Entertainment*, vol. 4 (1), January 2006.

[3] P. Fröjdh, U. Horn, M. Kampmann, A. Nohlgren, and M. Westerlund, "Adaptive streaming within the 3GPPP packet-switched streaming service", *IEEE Network*, vol. 20 (2), pp. 34–40, March-April 2006.

[4] N. Cranley, and M. Davis, "Study of behaviour of video streaming over IEEE 802.11b WLAN networks", in *2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 349–355.

[5] O.I. Hillestad, A. Perkins, V. Genc, S. Murphy, and J. Murphy, "Delivery of on-demand video services in rural areas via IEEE 802.16 broadband wireless access networks", in *2006 Proceedings of the Second ACM International Workshop on Wireless Multimedia Networking and Performance Modeling*, pp. 43-51.

[6] H. Fujisawa, H. Minami, M. Yamamoto, Y. Izumi, and Y. Fujita, "Route selection using retransmission packets for video streaming on ad hoc networks", in *2006 IEEE Radio and Wireless Symposium*, pp. 607–610.

[7] J. Ding, C. Lin, and K. Huang, "ARS: an adaptive reception scheme for handheld devices supporting mobile video streaming services", in *2006 IEEE International Conference on Consumer Electronics, Digest of Technical Papers*, pp. 141–142.

[8] G. Cunningham, S. Murphy, L. Murphy, and P. Perry, "Seamless handover of streamed video over UDP between wireless LANs", in *2005 Consumer Communications and Networking Conference,* pp. 284–289.

[9] J. Li, and L. Li, "Research of transmission and control of real-time MPEG-4 video streaming for multi-channel over wireless QoS mechanism", in *2006 1st International Multi-Symposiums on Computer and Computational Sciences*, vol. 2, pp. 257-261.

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

[11] G. Yang, L. Chen, T. Sun, M. Gerla, and M.Y. Sanadidi, "Real-time streaming over wireless links: a comparative study", in *2005 Proceedings of 10th IEEE Symposium on Computers and Communications*, pp. 249–254.

[12] M. Li, C. Lee, E. Agu, M. Claypool, and R. Kinicki, "Performance enhancement of TFRC in wireless ad hoc networks", in *2004 Proceedings of the 10th International Conference on Distributed Multimedia Systems*.

[13] S. Lohier, Y. Ghamri-Doudane, and G. Pujolle, "MAC-layer adaptation to improve TCP flow performance in 802.11 wireless networks", in *2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 427–433.

[14] IEEE 802.11, The Working Group Setting the Standards for Wireless LANs [Online]. Available: http://grouper.ieee.org/groups/802/11/

[15] J. Karlsson, H. Li, and J. Eriksson, "Real-time video over wireless ad-hoc networks", in *2005 14th International Conference on Computer Communications and Networks*, San Diego, California.

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

[17] T. Goff, N.B. Abu-Ghazaleh, D.S. Phatak, and R. Kahvecioglu, "Preemptive routing in ad hoc networks", in *2001 ACM SIGMOBILE*, pp. 43-52.

[18] G. Gaertner, and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks", *IEEE Internet Computing*, pp. 55-60, January-February 2004.

[19] J. Bardwell. (2004). You Believe You Understand What You Think I Said. The Truth About 802.11 Signal and Noise Metrics [Online]. Available: http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf

[20] A. Adya, P. Bahl, R. Chandra, and L. Qiu, "Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks", in *2004 10th Annual International Conference on Mobile Computing and Networking*, pp. 30-44.

[21] S. Ghandeharizadeh, S. Kapadia, and B. Krishnamachari, "Server replication techniques for display of continuous media in mobile ad hoc networks", Computer Science Department, University of Southern California, Los Angeles, CA, Tech. Rep. 2005.

[22] V. Sunderam, J. Pascoe, and G. Tonev, "Reconciling the characteristics of wired and wireless networks: the Janus approach", in *2002 4th Annual International Workshop on Active Middleware Services*.

[23] O L S R . O R G [Online]. Available: http://www.olsr.org/

[24] E. Macías, A. Suárez, J. Martín, and V. Sunderam, "Using OLSR for streaming video in 802.11 ad hoc networks to save bandwidth", *Special Issue of IAENG*, in press.

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