Overlay Solution for Multimedia Data over Sparse MANETs

Sergio Cabrero[1], Xabiel G. Pañeda[1], Thomas Plagemann[2], Vera Goebel[2], Matti Siekkinen[3]
[1]Department of Informatics, University of Oviedo, Gijón/Xixón, Spain
[2]Department of Informatics, University of Oslo, Oslo, Norway
[3]Department of Computer Science and Engineering, Helsinki University of Technology, Finland
{cabrerosergio, xabiel}@uniovi.es, {plagemann, goebel}@ifi.uio.no, matti.siekkinen@tkk.fi

ABSTRACT
Using Mobile Ad-hoc Networks (MANETs) for audio and video transmission is very promising for application domains such as emergency and rescue. However, audio/video streaming services are not designed for such dynamic and unstable networks. The problems are even more important in so-called *sparse MANETs* where the node density is relatively low so that disconnections and network partitions are common. We have designed an architecture that combines MANET routing with caching and delay tolerant store-carry-forward operations in an overlay network to improve the quality of audio/video transmission over sparse MANETs. We have implemented a prototype to evaluate the architecture. The results from the experiments demonstrate that our system clearly outperforms simple client-server solutions when the network has temporal disconnections.

Categories and Subject Descriptors
C.2.2 [**Network protocols**]: Applications.

General Terms
Algorithms, Design, Experimentation.

Keywords
Audio/video, sparse MANET, overlay, streaming.

1. INTRODUCTION
Mobile ad-hoc networks (MANETs) are a promising communication technology for environments lacking communication infrastructure, for example, in an area hit by a disaster. During the first hours after the disaster, the immediate goal of the response units is to save human lives. The efficiency of emergency and rescue operations can be improved by proper communication services. One such service could be video streaming from head mounted cameras of the response unit members to the command and control center. However, it is not clear whether such a service is feasible, because audio/video (A/V) streaming services are not designed to address problems that typically occur in MANETs, such as node mobility causing route changes, shared medium transmissions, or restricted bandwidth. In sparse MANETs where the node density is relatively low, disconnections and network partitions cause additional problems. For instance, connection loss between client(s) and server may terminate an existing session, if the disruption is long enough to produce a timeout in the TCP connection. Existing A/V streaming services assume connectivity between source and destination, which causes continuous packet losses when client(s) and server are in different network partitions. Moreover, the transport protocols that are used for streaming services, i.e., UDP and TCP, are both unsuitable. On the one hand, UDP is unreliable, which is a problem due to packet losses in MANETs. On the other hand, TCP has been shown to be problematic in this type of networks [1]. Specifically, TCP confuses network partitions with path congestion. When sender and receiver are disconnected, the protocol executes the exponential back-off algorithm, leading to a long idle connection period, even if the connectivity is restored.

In this paper, we present a middleware architecture for delay tolerant streaming over sparse MANETs to support multimedia streaming. The primary goal of our approach is to deliver as much video as possible, with the best quality achievable in such an environment. We describe a prototype implementation of the architecture and an extensive performance evaluation that demonstrates that our approach is in most node mobility and density combinations superior to classical client-server approaches. When network disconnections between server and client cannot be immediately overcome, it is unavoidable that users perceive a pause in the reception. However, as soon as the client can be reached again the live stream is transmitted to the client, and also those parts of the stream that could not be delivered due to the disconnection. At the application level, the user can decide to watch the live stream and/or the missed part of the stream in one or more windows, to store it, etc. Thus, certain audio/visual data might be delayed, but much more data is available for the user than in the classical client-server approach. The proposed solution to increase reliability of A/V streaming is the introduction of session nodes to, for example, cache data between server and client. In this way, the efficiency of error control is improved, because a long path in the MANET is composed out of a set of shorter paths. Furthermore, A/V data can be delivered to the client in case of network partitions, because session nodes serve as "message ferries". To make this idea work, specialized transport and signaling protocols and proper resource management is required.

There exist other proposals on many aspects of A/V streaming over MANETs. In [2], the use of Multiple Description Coding combined with multiple sources streaming to the same client is proposed. Thus, A/V content must be pre-distributed to several nodes that are subsequently used to stream parts of the media

IWCMC'09, June 21–24, 2009, Leipzig, Germany.

requested by the client node. In addition, Vista-XL [3] searches multiple paths between server and client using cross-layer information. Then, video is divided and sent using these paths, according to the QoS required. Seo et al. [4] propose an adaptive video encoding protocol. In this case, an optimal path is selected, examined and video is encoded taking into account the link capacity or the number of hops. In [5], a similar approach is proposed, focusing mainly on link layer parameters and looking for an optimal streaming rate. A multi-path routing scheme is used in [7] to improve streaming by finding several paths, better if disjoint, from the source to the receiver. Finally, MRTP [6] is also interesting as it proposes a protocol to support multi-path streaming by extending RTP. Our solution is different from related work. For example, the main stream solves problems of dense MANETs, such as looking for optimal paths, while we aim at sparse MANETs, where it is difficult to find paths. Furthermore, we try to tackle other problems, such as signaling or which applications to use, that are normally omitted from the scope of the existing research.

The remainder of the paper is organized as follows: In Section II, we describe the design of our solution. We have implemented a prototype, called MOMENTUM, which is described in Section III. Section IV presents the evaluation and the results. Section V concludes the paper.

2. DESIGN

The design of our streaming solution is strongly influenced by our goal that it has to run in real environments and not only in simulators. Thus, the system architecture should enable the reuse of existing tools and protocols, like standard MANET routing protocols and TCP/UDP, and off-the-shelf video streaming client and server.

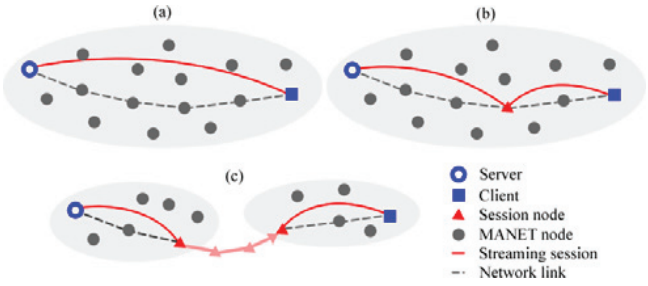

Figure 1. Client-server (a) vs. session node as a relay (b) or as a ferry (c)

Based on these primary requirements, we have designed our streaming solution as an overlay network on top of standard MANETs. The main idea is that the set of nodes that form the overlay provide delay tolerant transmission of data, but communicate with each other using standard routing and transport protocols. Delay tolerant data transmission is achieved by treating each message in a store-carry-forward [8] manner. The overlay consists of three types of nodes: client nodes, server nodes, and session nodes. Server nodes stream multimedia data and client nodes receive and play multimedia data. Session nodes collaborate in the data delivery. For example, when server and client are connected, session nodes could be used as relays, increasing reliability in the path and improving recovery after disruptions. However, if server and client are on different network partitions, a session node, connected to the server, could carry video to the client partition and forward it. In this way, our solution benefits from node movement and uses alternative "time" paths of communication between server and client. Therefore, multimedia sessions can be established even without direct connection between server and client. If we compare our overlay streaming solution to a classical client-server setup (Figure 1(a)), we observe that the session can be partitioned into multiple overlay hops and video data can be stored on multiple nodes (Figure 1(b)) or data can be stored-carried-forwarded by a node moving from the server partition to the client partition (Figure 1(c)).

The resulting architecture, shown in Figure 2, applies the main concepts of our proposal and divides the system into components. It can be defined as a middleware, with multimedia applications on top and standard network protocols below.

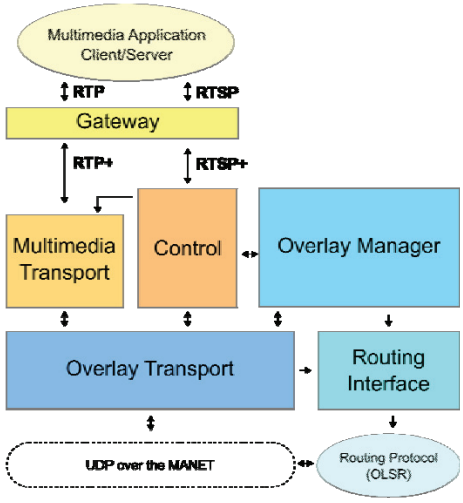

Figure 2. Overlay architecture

2.1 Routing Interface

The Routing Interface component gathers information from the routing protocol, such as neighbors, topology or hops to other nodes. This knowledge is often used for route planning and session node selection. The Routing Interface makes the rest of the system routing protocol independent.

2.2 Overlay Transport

The Overlay Transport protocol aims to provide efficient communication between overlay nodes. It tackles MANET issues, such as disconnections or dynamic topology, and isolates the rest of the components from them. The protocol should consider communication requirements of other components, such as those coming from multimedia services or overlay management. Then it should apply known network patterns and new mechanisms to overcome the challenges of MANETs. First, the heterogeneous nature of traffic requires prioritization and scheduling of packets. Thus, if a good traffic classification policy is applied, resources are spent on important content for the user experience. At this level, priority can change over time and affect packet sending and storage. In addition, MANETs are an unreliable environment, the well-known positive ACK and retransmission (PAR) paradigm can be useful to overcome losses of important packets. Furthermore, resources of the network can be saved checking the

connectivity of a node before sending packets to it. Finally, a control flow mechanism, that limits the sending bitrate of the node, can be useful to avoid network congestions in some situations. For example, when using the store-carry-forward paradigm, a node A can store many packets destined to a node B that is disconnected. When A and B reconnect, A might try to send all the packets as fast as possible, causing network congestion and UDP buffer overflows.

2.2.1 Overlay Routing

The Overlay Transport protocol must include the distributed nature of our solution. We have designed an Overlay Routing protocol that benefits from network topology provided by the Routing Interface. For example, if a node A is on the way to a node B, maybe not on the optimum path but close to it, the source can send a message to A, which will forward it to B. Moreover, if the message is lost on the way from A to B and it has to be retransmitted, this can be done by A. Extending this idea to video transmission from the server to possibly many clients and session nodes, the savings in network resources could be significant. Overlay routes increase reliability and save resources over common network paths. We consider that overlay routes can be planned by the source and progressively optimized by the rest of the nodes in the route, as shown in Figure 3. The Overlay routing protocol should notify nodes about new routes, removed routes and updates due to topology changes.

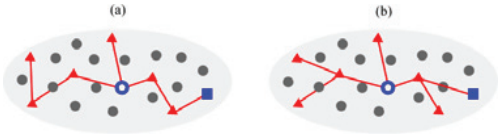

Figure 3. Overlay routes are planned (a) and optimized (b) on the fly.

2.3 Multimedia Transport

In contrast with the Overlay Transport protocol, the Multimedia Transport component contains a specific multimedia transport protocol (RTP+) for A/V streams. Furthermore, one of its important tasks is the assignment of Overlay Transport parameters (priority, reliability, etc.) to multimedia contents. Streams should be delivered from server to session nodes, between session nodes and from session nodes to clients. Thus, RTP+ should adapt to circumstances and available resources in each streaming hop. RTP+ uses the Overlay Transport protocol to send and receive messages. Thus, the selection of the adequate Overlay Transport protocol parameters is also important for RTP+ streaming success. Finally, there are many interesting approaches that can be used for stream adaptation. For example, tagging packets according to their payload, e.g. I, B, or P-frames, and streaming them according to the available resources. Indeed, some related works try different approaches on this, see [5].

2.4 Control

The Control component implements the streaming control protocol (RTSP+). We could define a multimedia session as the global process to deliver a specific A/V content, e.g. a live/stored video, from a source to a user or users. RTSP+ coordinates nodes in the session and establishes the policies that RTP+ must follow to stream contents. For example, when two session nodes arrive at the same time at the client partition, the Control protocol must determine how they will stream to the client with the best quality and no replication. Furthermore, the protocol must support sudden streaming stops and restarts due to disconnections and not to users' requests. The information from the Overlay Manager can be used to determine the available resources for streaming or select the session nodes.

2.5 Overlay Manager

Management of the overlay network should include features related to local and distributed resource monitoring. This information can be used with diverse tasks related to the streaming of A/V, such as session node selection or resource allocation. Several factors could participate in session node selection, such as available resources, topology, position, movement patterns, existing sessions, their priority or "a priori" knowledge. Finally, session node selection is carried out on demand of the Control component and Overlay Transport is used to communicate with other managers.

2.6 Gateway

The Gateway component shields the client and server application from all events that are related to disconnections and network partitions and, thus, emulates for the application a standard streaming environment. In order to support off-the-shelf streaming servers and video players, the RTP/RTSP packets from the standard streaming server are intercepted and translated into the Multimedia Transport and Control protocols at the server nodes and back to standard RTP/RTSP messages at the client nodes.

3. PROTOTYPE IMPLEMENTATION

We have implemented a prototype in Java, called MOMENTUM. The purpose is to have a scalable and ready to deploy prototype, where our ideas can be added progressively and their validity evaluated in environments as close to reality as possible. We have chosen the Optimized Link State Routing (OLSR) [10] as the underlying routing protocol. For simplicity, we use UDP below the Overlay Transport. The chosen multimedia applications are OpenRTSP (client) and Live555MediaServer .

3.1 Overlay Transport

The Overlay Transport protocol is in charge of sending, receiving, storing and forwarding packets of all other components. Standard streaming protocols like RTSP and RTP communicate using TCP and UDP. In contrast, the Overlay Transport protocol uses only UDP below, because of the bad performance of TCP in MANETs [1]. In order to increase efficiency, we carry out selective packet forwarding. We distinguish the final destination of a packet, for example the client is the final destination of video packets, and packets are sent only to the next hop in the overlay route that goes to the final destination. Otherwise, if the final destination is not in the same network partition, packets are delivered to all the nodes in the overlay route in the same partition. It is very likely that one of these nodes will reach the final destination and forward the message. In other words, packets are either sent directly to their destination or replicated in session nodes until the delivery is possible for one of them. Prioritization of packets is implemented by a set of outgoing queues sorted by priority. These queues are polled from high to low and the packets sent in this order. In addition, a packet is only sent when the destination is connected, according to the network routing protocol. Output flow control is established with a maximum of 11Mbps. Thus, a delay between

packets is introduced to avoid the surpassing of this bitrate. Finally, we include acknowledgments and retransmissions for Control messages and Overlay Route Announcements. At this stage, a 3 second timer triggers a retransmission, if the ACK is not received from any of the next hops in the overlay route. This 3 seconds value has shown a good performance in the experimentation, however other techniques such a dynamic (TCP-like) adjustment of the retransmission timer should be studied.

We have implemented a first version of the protocol that supports route planning, announcement, optimization and updating. The intermediate nodes algorithm from the Routing Interface is used for initial planning and optimization of the route. Then a Route Announcement is spread from the source to all the nodes in the route. At each overlay hop, every node optimizes the route by planning again the topology for its successors. Once the route is established, updates are triggered by the source node of the route using a simple connectivity checking. If nodes in the route disconnect, the route is reset. All nodes are responsible for delivering packets to the disconnected nodes, but thanks to Receiver Reports it will only be done by one of them. Receiver Reports are messages containing the list of packets already received by a node, and they are sent whenever a duplicated packet is detected. Thus, memory can be freed and network resources are not wasted.

3.2 Multimedia Transport and Control

For this first prototype, these protocols are extended versions of RTSP and RTP. The first extension done to the control protocol is the inclusion of a new field "Session nodes" in the RTSP messages. When a node receives a control message it becomes part of the session and knows which other nodes are in this session. Packet prioritization and other desirable adaptation features are not yet included in this prototype.

In a traditional multimedia session, multimedia streams and control channels are differentiated by the connection at transport level, in other words the protocol and the ports. This approach is not valid for our solution, because data is delivered using a common transport channel. For that reason, packets are labeled with a stream and session identifier. Therefore, they can be processed in the session nodes and delivered correctly in the client or server gateways.

3.3 Overlay Management

```
SN: Session nodes, DC: Stored Destination connectivity

if (SN == null || DC != isConnected(destination))
   DC = isConnected(destination)
   if (DC == true)
       SN = Random selection among intermediate nodes
   else
       SN = Random selection among neighbors
else
        Use SN
```

Figure 4. Session node selection code

The Overlay Manager's current task is the selection of session nodes. As we have discussed, this can be a complex task. The results presented in this paper have been obtained using a simple algorithm that carries out the selection of session nodes depending on the connectivity between client and server. Thus, when the server/client node needs to send a message (source) to the client/server (destination), session nodes are checked (see Figure 4 for pseudocode).

3.4 Gateway

The Gateway must translate RTSP/RTP to our extended protocols. Thus, it must intercept and also send messages from/to the applications. In addition, it communicates with the applications using RTCP, so they do not start unilateral session shutdowns. We have checked that the server needs periodic RTCP Receiver Reports to keep the session open. The Gateway generates them, pretending to be the client because the real client application may be disconnected.

3.5 Routing Interface

```
S: Source node, T: Target node, IN: Intermediate nodes

h = hops from S to T
IN[h] = T + neighbors of T at distance h of S
while (h > 0)
{
   for each n in N[h]
       IN[h-1] += neighbors of n at distance h-1 of S
   h--
}
```

Figure 5. Intermediate nodes algorithm

The Routing Interface component gathers information from the routing protocol, which is often used for route planning and session node selection. The overlay currently uses a proactive underlying routing protocol (OLSR), because it maintains full state of the network topology and in this way provides crucial information for the overlay management and routing. The algorithm of Figure 5 is implemented to calculate the set of nodes in the path from one node to another using the information declared by TC messages from OLSR, see [10]. It is used for session node selection and overlay route planning. Its result is a set of nodes that stays topologically in the middle of two others, or, in other words, the nodes surrounding the shortest path.

4. EVALUATION

4.1 Experiment setup

In order to understand and quantify the tradeoffs of performing classical client-server streaming versus delay tolerant overlay solutions of streaming protocols for sparse MANETs, we have performed several experiments with our current prototype. We assume that a traditional client-server solution will work more efficiently than our overlay approach in fully connected networks, but with increasing amount of route changes and network partitionings and mergings the advantages of the overlay approach should overcome the additional overhead introduced by it. Furthermore, it is obvious that there are certain scenarios with very low density and mobility in which none of these streaming solutions are feasible. In order to identify the regions in the mobility/density space in which the different approaches are preferable, we aim to cover in our experiments the entire (realistic) mobility density space.

For our studies, we use the MANET emulator NEMAN [9], because it allows us to run on top of virtual network interfaces (TAP interfaces) our prototype code without any modifications. NEMAN accepts standard ns-2 scenario files as input and is able to simulate MANETs on a single PC and run for each of these

nodes our protocols (including the application at client and server) in independent processes.

We generate two types of Random Waypoint mobility scenarios with a duration of 1000 seconds. Firstly, we vary the node density from 2 - 20 nodes with speeds ranging from 1 - 20 m/s in an area of 1000x600 meters. Secondly, we consider larger scenarios, 3000x1000 meters, with 30, 40 and 50 nodes and the same speed variation, 1-20 m/s with a duration of 1000 seconds. The communication range of the nodes is 250 meters. As workload we use a 1200 seconds long action video encoded in MPEG-2. The video is composed of a single stream, with the same priority for all the packets in it. Since the video is longer than the experiment, our results are similar to a live video situation. Finally, cross traffic, packet losses, collisions, etc. are not considered in these preliminary experiments. For each mobility scenario, we evaluate the client/server approach using directly the MANET, and our solution, labeled as MOMENTUM. Each experiment is repeated five times and the values we report are the average values.

4.2 Results

In Figure 6, we show a comparison of the amount of video received at the client. In order to illustrate our results for the entire mobility/density space in a concise way, we plot for each combination of number of nodes and node speed the relative improvement of MOMENTUM to the client-server solution. The surface of the bubble expresses the difference in bytes from one solution to the other, colored green when MOMENTUM receives more video and gray when the client-server solution is better.

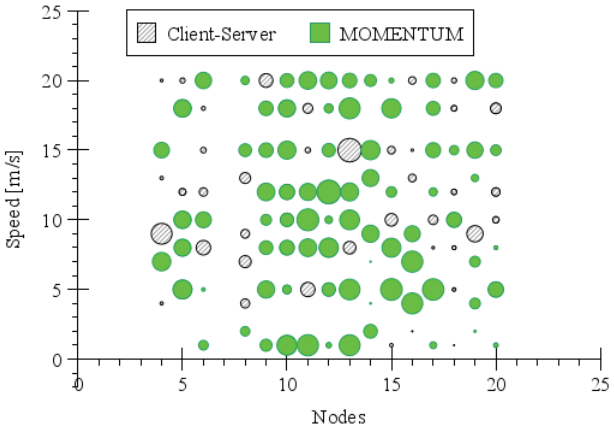

Figure 6. Relative increase of received video

We can see that in most of the scenarios MOMENTUM is superior. In bigger scenarios with 30, 40 and 50 nodes results are similar. We have found that the scenarios that present better results of the client-server approach are the ones in which client and server nodes are in the same partition during a high percentage of the experiment duration. We have studied and will design solutions to improve the behavior in these cases, although our solution is designed for scenarios that lack connectivity. The delay introduced by MOMENTUM and other communication issues are the main factors behind the better performance of client-server in these scenarios. For example, if an overlay route breaks, there is an idle time where packets are stored before the session goes on. In addition, the Overlay Transport protocol introduces an artificial output bitrate limitation (11 Mbps). This makes the delay of the packets in our solution bigger than the packet of the client-server approach, which is close to zero in the emulation environment. However, our solution will pay off in a real environment with limited network bandwidth.

These results show that disconnections and disruptions can be overcome using store-carry-forward approaches. However, during live video streaming, a temporal disconnection of the client implies a disruption in the video delivery and might cause a frozen frame in the video player. In such a case, an adequate buffer dimensioning is a powerful tool to obtain a smooth user experience. Therefore, we compare in the next set of experiments the seconds of video that are received in two different scenarios.

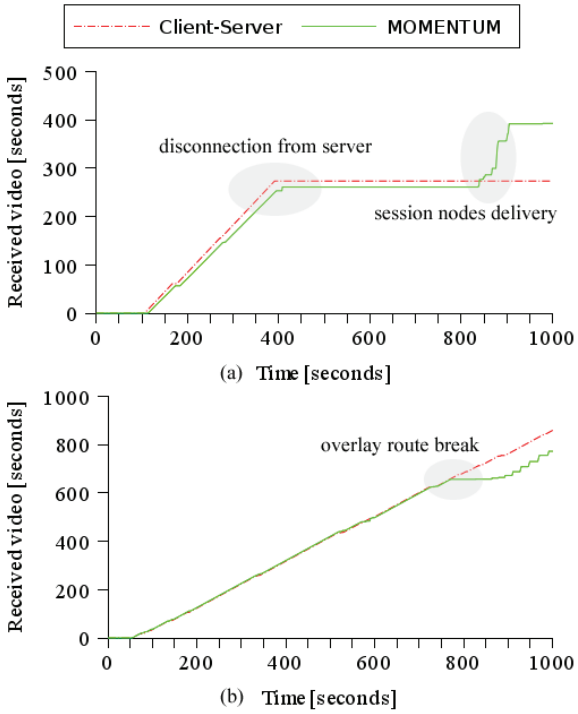

Figure 7. Video seconds with (a) 12 nodes at 10 m/s and (b) 18 nodes at 18 m/s

Figure 7(a) shows the amount of received video in a scenario with 12 nodes moving with an average speed of 10 m/s. Specifically, it represents the accumulated seconds of video received by the client at every second of the experiment duration. In the first 150 seconds, it is impossible for both solutions to establish the session. Then, both are streaming and we can observe a small delay suffered by MOMENTUM. After 400 seconds, a sudden disconnection occurs and none of the solutions are able to reach the client for a significant period of time. While for the client-server approach this is the end of the streaming session, our delay tolerant solution overcomes this situation by sending more video through ferry nodes that reach the client at some point, approximately at 850s. Hence, these results prove the validity of message ferrying applied to multimedia transmission in our system.

In contrast to Figure 7(a), Figure 7(b) shows an unfavorable scenario, where client-server receives more video. The scenario is composed of 18 nodes moving at an average speed of 18 m/s. The session is established from almost the beginning of the

experiment and the streaming is constant in both cases. However, there is a disconnection of one of the session nodes used as a relay by our proposal. Then, packets stop reaching the client, the session needs to be reconfigured, and at the end of the emulation the client-server approach shows a better performance. Although we are working to solve this type of issues, if the disruption has occurred at the beginning of the scenario and not at the end, MOMENTUM would have time to recover and the total amount of video delivered by both solutions would be the same. Furthermore, we think that relay session nodes, like the one producing the disruption here, will show their real potential in scenarios with packet losses.

In Figure 8 we extrapolate the results in a density against mobility domain. When density and mobility are low, it is very difficult to find communication paths and no solution will work properly. In sparse MANETs with different ranges of mobility, our solution is better than others. Mobility helps us to establish communication paths with session nodes acting as ferries, although they move freely. Moreover, if there was "a priori" knowledge about node movement, we could use that to obtain better results. In addition, we increase reliability when there are continuous topology changes. There are scenarios where networks are connected and are almost static. In such a situation, the client-server paradigm can work together with other ideas, for example multipath routing to avoid link congestions.

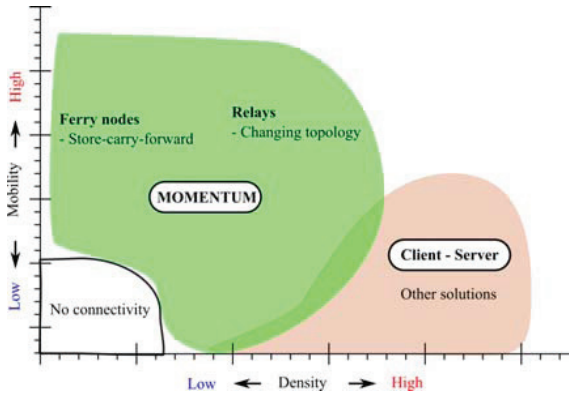

Figure 8. MOMENTUM application area

5. CONCLUSIONS

In this paper, we propose a solution to support A/V streaming over sparse MANETs. We have presented an architecture and described the necessary protocols to solve the main issues in this environment. In addition, we have implemented and emulated a prototype. The results show how our design outperforms client-server solutions in this environment.

The prototype development and evaluation processes confirm our architectural choices. Although we know that there is still room for improvements, the division in independent components has been very adequate when it is necessary to fine tune parts of the system. A good initial architecture, such as this one, will definitely speed up the evolution of the components and protocols. In addition, it will ease the testing of new ideas.

For future work, our first step will be the complete implementation and evaluation of our ideas. In addition, we consider the improvement of some aspects. For example, other session node selection algorithms could be designed and evaluated to see which one is better depending on the situation. For this purpose, resources of the nodes, node movement patterns and other useful information could be added as parameters of the algorithm. We also aim for a full development of the multimedia protocols, both for Control and Multimedia Transport. Specifically, the Control protocol should be capable of making decisions in a broad variety of scenarios. The first step for the Multimedia Transport protocol could be the implementation of an adaptive streaming mechanism. The Overlay Transport protocol could also be revised, improving retransmissions and testing different prioritization policies to see which one best suits streaming applications. Finally, we aim to achieve an exhaustive evaluation with other simulators, real test-beds, other routing protocols, more scenarios, realistic physical layer, and background traffic.

6. ACKNOWLEDGMENTS

This work has been funded by the Spanish National Research Program (FUTURMEDIA Project TSI2007-60474), the Norwegian Research Council (DT-STREAM Project 183312/S10), and it received support from the EU Network-of-Excellence CONTENT.

7. REFERENCES

[1] G. Holland, N. Vaidya. "Analysis of TCP Performance over Mobile Ad Hoc Networks," Wireless Networks, Springer, 2002.

[2] C.O. Chow, H. Ishii. "Enhancing real-time video streaming over mobile ad hoc networks using multipoint-to-point communication," Computer Communications, Elsevier, 2007.

[3] G.D. Delgado, V.C. Frias, M.A. Igartu. "ViStA-XL: A Cross-Layer Design for Video-Streaming over Ad hoc Networks," 3rd Int. Symp. Wireless Communication Systems, 2006.

[4] J. Seo, E Cho, S. Yoo. "Protocol Design for Adaptive Video Transmission over MANET," Lecture Notes in Computer Science, Springer, 2006.

[5] E. Setton, T. Yoo, X Zhu, A. Goldsmith, B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," IEEE Wireless Communications, 2005.

[6] S. Mao, D. Bushmitch, S. Narayanan, S. S. Panwar. "MRTP: A Multi-Flow Realtime Transport Protocol for Ad Hoc Networks," IEEE Transactions on Multimedia, 2006.

[7] S. Kompella, S. Mao, Y. Thomas Hou, H.D. Sherali, "Cross-Layer Optimized Multipath Routing for Video Communications in Wireless Networks," IEEE Journal on Selected Areas in Communications, 2007.

[8] W. Zhao, M.H. Ammar, "Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks," 9th IEEE Workshop on Future Trends of Distributed Computing Systems, 2003.

[9] M. Pužar, T. Plagemann, "NEMAN: A Network Emulator for Mobile Ad-Hoc Networks," 8th Int. Conf. on Telecommunications, 2005.

[10] T. Clausen, P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626. 2003.