An Energy-Delay Routing Protocol for Video Games over Multihops Ad Hoc Networks

Arnaud Kaiser, Khaled Boussetta, and Nadjib Achir

L2TI – Institut Galilée – University of Paris 13
93430 Villetaneuse, France
firstname.lastname@univ-paris13.fr

Abstract. Nowadays, with the development of networked technologies and the great expansion of video games industry, almost all video games propose a multi-player mode. Development of portables consoles which integrate 802.11 technology offers to people the opportunity to play anywhere, at any time and with anybody. However, wireless networked conditions are not always optimal and directly impact the quality of game perceived by players. In this paper, we propose a multimetric ad hoc routing protocol, based on OLSR, which improves game experience in terms of fairness among players and gaming sessions lifetime. We consider the delay and the energy as metrics to route informations through the network. We compare our routing protocol with an energy-efficient OLSR version and a delay-efficient OLSR version. We show that our routing protocol provides better performances in terms of fairness while keeping a good network lifetime.

Keywords: multihops ad hoc networks, multimetric routing, networked games, objective evaluation.

1 Introduction

Over the past fifteen years, the video games market has grown at an exceptional rate. This impressive progression was sustained by the innovation and the creativity of this industry. Among the basic ingredients in recent successful games, interactivity supported by network communications is a key element. As a matter of fact, most of nowadays popular games integrate a multiplayer mode.

Taking into consideration the popularity of LAN multiplayer games and the tremendous success of wireless networks, we believe that future game consoles will benefit from supporting MANET (Mobile Ad-hoc NETworks) mode. Indeed, MANET allows spontaneous creation of data networks by exploiting the multihops communication capacity of the mobile terminals. Moreover, all the networking functionalities (multihop routing, auto-configuration, self healing, security, etc.) are carried out by the terminals themselves in an entirely distributed way. Therefore, besides the social and cultural interest of playing with close located persons, MANET technology can allow players, to easily improvise a LAN multiplayer party without any need for an existing fixed wireless network

D. Simplot-Ryl et al. (Eds.): ADHOCNETS 2011, LNICST 89, pp. 193–208, 2012.

infrastructure. A scenario, like a group of kids equipped with MANET terminals wishing to play anywhere (e.g. in a playground, in a school bus etc.) a WLAN party will be possible.

In order to make such scenario a common based reality in the near future, several issues still have to be addressed by the research community. Several difficulties come from the inherent constraints that characterize MANET technology. In particular, the mobility of the core network, the energy limitation and the variability of shared resources. All these problems lead to a variable connectivity and potentially, to a high fluctuation in the provided Quality of Service (QoS). On the other hand, multiparty games are very sensitive to resource availability. They are very demanding on CPU and thus, high consuming on energy. In addition, the game play could experience a significant degradation caused by delays variability, which is typically the case in MANET, due to the mobility of users and to the random nature of IEEE 802.11 MAC layer.

In this work, we address the routing issue for multiplayer networked games in a multihop ad hoc network environment. We consider specifically the most popular ad hoc routing protocol. Namely, OLSR. Our aim is to increase the network life duration and improve game fairness considering OLSR and supposing a multiplayer FPS (First Person Shooter) game. We propose and analyze, through realistic experimentations and using objective metrics, a multimetric routing protocol based on energy consumption and end-to-end delay.

The remainder of this paper is organized as follows: in the next section, we briefly introduce OLSR and present the state of the art on ad hoc routing under energy and delay criteria. Section III describes our routing protocol. Section IV gives a detailed description of our experimentation scenarii. In section V, we analyze and discuss the obtained results. Finally, section VI summarizes the main contributions of this work and provides some future prospects.

2 Related Work

2.1 Ad-Hoc Routing Protocols

Routing protocols for ad hoc multihop networks can be classified into two main categories: *reactive* protocols and *proactive* protocols.

Routing protocols belonging to the first category compute a route to reach a destination only when needed. As long as a node in the network doesn't want to send data, no signaling messages are sent, leading to save energy and reduce congestion and collisions. However, when a node wants to send data, it first has to start a route discovery mechanism to find a route to reach its destination. This mechanism is time consuming and hence delays the data sending until the node has found a route.

Routing protocols from the second category periodically send signaling messages in order to discover and update the network topology. A node using this kind of routing protocol always has a route available to reach every nodes present in the network. Thus, the node can instantly send data when needed. However,

signaling message used are energy consuming and increase risks of congestion and collisions.

A *hybrid* category also exists. Routing protocols belonging to this category combine both reactive and proactive algorithms.

As we are considering a time sensitive application where delay is a crucial metric, we decide to base our work on the well-known proactive routing protocol OLSR.

2.2 OLSR Overview

OLSR [1] is a proactive routing protocol, which means that it memorizes permanently paths to all discovered nodes in the network. Thus, all nodes can be reached at any time. In order to discover and update the network topology, OLSR sends periodical signaling messages known as HELLO and TC messages.

A node uses the HELLO messages to discover its neighborhood. The HELLO messages are broadcasted with a TTL (Time To Live) of 1 such that only direct neighbors receive the message. A HELLO message contains the neighbors table of the node that sent it. The neighborhood table is a list of all the neighbors known by the node. An information about the link type to these neighbors is also noticed. A link with a neighbor can be unidirectional or bidirectional. If a node receives a HELLO message from a neighbor and sees its own IP address inside, it knows that the communication with this neighbor works in both sides, so the link is bidirectional. Finally, thanks to these HELLO messages, all nodes know their direct neighbor nodes (1-hop neighbors) and their 2-hops neighbors.

The second step of the protocol is the MPR (Multi-Point Relay) selection. The MPR selection is a technique that improves broadcasting messages by allowing only a subset of nodes to forward broadcasted messages. Each node chooses a subset of nodes among its 1-hop neighbors, such that all of its 2-hops neighbors are reachable via this subset. The chosen subset is called MPR set. When a node broadcasts a message, only its MPR nodes are allowed to forward the message. Thus, the number of retransmissions of the broadcasted message is reduced. The nodes inform their neighbors that they have chosen them as MPR via the HELLO messages. The nodes that were elected as MPR, create a MPR selector table which lists all neighbors that have chosen them as MPR. As an example, let us consider the network topology presented in figure 1. C_8 has three 1-hop neighbors (C_1, C_2 and C_3) and three 2-hops neighbors (C_4, C_5 and C_6). C_8 has to choose its MPR nodes among its 1-hops neighbors in order to be able to reach nodes C_4, C_5 and C_6. C_4 and C_5 are reachable via C_1, C_6 is reachable via C_3, so let us assume that the MPR nodes of C_8 are C_1 and C_3. When C_8 broadcasts a message, C_1 and C_3 will forward it but not C_2. This reduces flooding of the network.

The next step of the protocol is to transmit the MPR selector table to all nodes in the network. To this end, the TC (Topology Control) messages are used. All MPR nodes broadcast periodic TC messages that contain their MPR selector table. The TTL value is set to 255 (maximum value) such that all nodes

in the network receive it. Of course, the flooding of the TC messages uses the MPR technique.

Finally, by combining informations contained in HELLO and TC messages, the nodes have a global view of the network topology. Then, they compute the Dijkstra's shortest path algorithm to find the bests routes to reach all destinations in the network. The metric used to compute the routes is the number of hops.

2.3 Integrating Energy Consumption in OLSR

In order to reduce the energy consumption in ad hoc networks, many works were done in the literature to combine the OLSR protocol with the energy metric.

In [2], instead of the number of hops, the authors consider the residual energy of nodes to compute routes. Thus, they avoid nodes with low remaining energy, leading to increase the network lifetime.

In [3], the authors go further and also consider the energy consumed at the 1-hop and 2-hop neighbors of the node that sends a message. Indeed, as we are in a wireless environment, when a node sends a message, all its neighbors receive it, even if they are not concerned by it, leading to consume energy.

In [4], the authors base their route computing on a min-max energy algorithm. In each available route to reach a destination, they look at the node that has the lowest remaining energy. They choose the route whose lowest remaining energy node has the greatest value.

In [5], the authors exploit the Willingness field of the HELLO messages to compute their link cost. The Willingness is a value that informs if a node will forward data or not. When a node has a low remaining energy, it sets its Willingness to 0 in order to inform the other nodes that it won't forward traffic anymore. The authors also use the RTS/CTS messages in order to turn off (sleep mode) the nodes that are not concerned by the actual communication.

2.4 Integrating Delay in OLSR

As shown in our previous work [12], in the context of real time video games the delay metric is the most important one. Some works in the literature have studied the combination of the delay metric and the OLSR protocol.

In [10], the authors compute the delay between two neighbor nodes. They assume that nodes are synchronized. They timestamp the HELLO signaling messages, such that a node receiving a HELLO message can calculate the delay value of the link to its neighbor. The delay metric is then used in the Dijkstra's shortest path algorithm to compute routes.

In [11], the authors add three new signaling messages in the OLSR protocol. These messages enable to compute the end-to-end delay value between two nodes in the network, without the need of a synchronized network.

2.5 Multimetric Routing Protocol

Some works have been done to combine energy and delay metrics in a routing protocol.

In [6], the authors propose EDC-AODV (Energy Delay Constrained AODV), a modified version of the Ad-hoc On-demand Distant Vector (AODV) [13] routing protocol. Each node takes into consideration the current size of its buffer (which informs about the delay and congestion at this node, noted Q) as a first metric, and its residual energy as a second metric (noted ER). The cost of a link between two neighboring nodes is processed using the following equation:

$$cost = \alpha \sum_i \frac{1}{1 + ER_i^t} + (1 - \alpha) \sum_i (1 - \frac{1}{1 + Q_i^t}) \quad (1)$$

They use a parameter α in order to strike a balance between the two metrics. They compare their EDC-AODV routing protocol with the classical AODV routing protocol for different values of the α parameter.

In [7], the authors propose OEDR (Optimized Energy-Delay Routing), an ad-hoc routing protocol based on energy and delay metrics. They use OLSR as a base protocol and modify it. They consider the delay between two neighbors, measured thanks to timestamped HELLO messages, as the delay metric. As for the energy metric, they consider the residual energy of the nodes. They use the HELLO messages to transmit the energy and the delay values. Then, the cost of a link is the multiplication of the delay and the energy values. The Dijkstra's shortest path algorithm is finally processed using this cost to find the bests routes.

In this paper, our work consist of proposing a routing protocol, specific to real time client-server video games, which consider both the energy and delay metrics to improve the gameplay and also to increase the duration of gaming sessions. Indeed, as we have previously shown in [14], a player who has a great end-to-end delay with the game server has a bad game quality. On the contrary, a player who is close to the game server in terms of end-to-end delay has a better game quality, but his energy decreases dramatically faster because he has to forward the game traffic coming from the other players. In the next section, we describe our proposal, which is close to the one proposed in [6]. However, we decide to dynamically modify the value of the balancing parameter depending on the distance, in terms of end-to-end delay, that separates a player and the game server. If a player has a great end-to-end delay with the game server, he gives priority to the delay metric. On the contrary, a player who has a smaller end-to-end delay with the game server favors the energy metric.

3 Energy-Delay Routing Protocol

In this section, we detail our routing protocol algorithm that is based on two metrics: energy and delay. Our protocol uses a parameter, like the one presented in [6], to introduce a balance between the two considered metrics. We propose to dynamically modify the value of the parameter depending on the position of the nodes compared with the game server node.

3.1 Considered Energy and Delay Metrics

We consider the residual energy of the nodes as the energy metric. The *energy_cost* of a link is computed as follow:

$$energy\_cost_{n_s \to n_r} = 1 - RE_{n_r} \tag{2}$$

where n_s is the node which send a message and n_r is the neighbor of n_s which receives the message, and RE_{n_r} is the residual energy of the node n_r. The residual energy of the nodes is normalized such that $0 \leq RE \leq 1$.

As delay metric, we consider the delay measured on a link between two neighbor nodes. We assume that all nodes in the network are synchronized. Thus, by timestamping the OLSR HELLO signaling messages, the nodes can easily compute the delay value of the links by subtracting the timestamped value of the HELLO message from their current clock. Finally, the *delay_cost* of a link between two neighbor nodes is the delay measured on this link.

In order to inform all the nodes of the *delay_cost* and the *energy_cost* of the links, each node adds its computed values in its HELLO and TC messages.

3.2 Link Cost Function

The cost of a link between two neighbor nodes directly depends on the position of the nodes compared with the game server in terms of end-to-end delay. Hence, a node first computes its shortest path to reach the game server by considering only the delay metric. We call the total cost of this path the *delay_path_cost*.

Our goal is to find a compromise between the delay and the energy metric. However, players who have a great *delay_path_cost* should not consider the energy metric, but only focus on the delay metric. To this aim, we define a *delay_threshold*, above which we consider that the players are too far from the game server, thus very hampered. All the nodes whose *delay_path_cost* is greater than the *delay_threshold* only consider the delay metric to compute their paths. All the other nodes consider both energy and delay metrics to compute their paths. For these latter, the link cost function is defined as follow:

$$link\_cost = f.delay\_cost + (1 - f).energy\_cost \tag{3}$$

where f is a balancing factor which varies between 0 and 1. This balancing factor determines the weight of each metric. Each node computes its own balancing factor f following this equation:

$$f = \frac{delay\_path\_cost}{delay\_threshold} \tag{4}$$

with $0 \leq f \leq 1$. If f is greater than 1, we set it to 1 and thus the node will only consider the delay as link cost. This parameter varies from one node to another

in function of their position compared with the game server in term of end-to-end delay. The far away from the game server a node is, the less it takes into account the *energy_cost*. On the contrary, the closest from the game server a node is, the less it takes into account the *delay_cost*.

4 Experimental Settings

In order to evaluate the performances of our multimetric routing protocol, we set up a multihop ad hoc network emulator.

Our testbed is composed of eight computers: seven clients (each one representing a player in the game) and one dedicated game server. The computers are connected in a LAN via a switch. The clients hardware configuration is as follow: a 2 GHz Core 2 Duo processor, two gigabytes of main memory, and consumer-level graphic and network cards integrated into the motherboard. The server has a 2.4 GHz Core 2 Duo processor with two gigabytes of main memory. All the eight computers run the same software configuration: Linux Ubuntu 10.04 distribution with the 2.6.32.21 kernel version.

As our computers are directly connected to AC power, we emulate the battery consumption. In the beginning of each experimentation, all the nodes are full of battery. Each time a network frame is sent or received by a node, we decrease the remaining energy of this node according to the energy consumption model presented in [9]. This model takes into consideration the size of the frames and the nodes state (transmission, reception, idle) to compute the specific amount of energy spent when sending or receiving a frame. The consumed energy is computed as follow:

$$consumed\_energy = m * s + b \tag{5}$$

where s is the size of the frame (in bytes), m and b are constants which can be determined empirically. The constant values that we used in our experimentations are given in table 1. In order to obtain the size of the frames, we use the *pcap API* to capture the frames and recover their length. To simplify the analysis, we consider neither the energy consumed in idle mode, nor the energy consumption due to the game engine processing. We just focus on the transmission and the reception of frames. Finally, we stop our experimentations when the residual energy of a node reaches 0%.

To emulate the wireless conditions, we add some delay on each link. The delay values are chosen depending on the density of the nodes. Each node has a 10 milliseconds delay penalty per neighbor. Thus, the more neighbors a node has, the more penalized it will be. We use the *Traffic Control* (TC) and *Netem* tools available on Linux kernel to add the delay penalties.

The multihop network is emulated by using the *iptables* tool, also available on Linux kernel. Each node adds *iptables* filters, based on the MAC address, to drop all the frames that it should not have received in a real ad hoc environment. Figure 1 depicts the network topology we used in our experimentations. Values on links are the delays values in milliseconds.

Table 1. m & b values

	m (μW.s/byte)	b (μW.s)
point-to-point transmisson	0.48	431
broadcast transmisson	2.1	272
point-to-point reception	0.12	316
broadcast reception	0.26	50

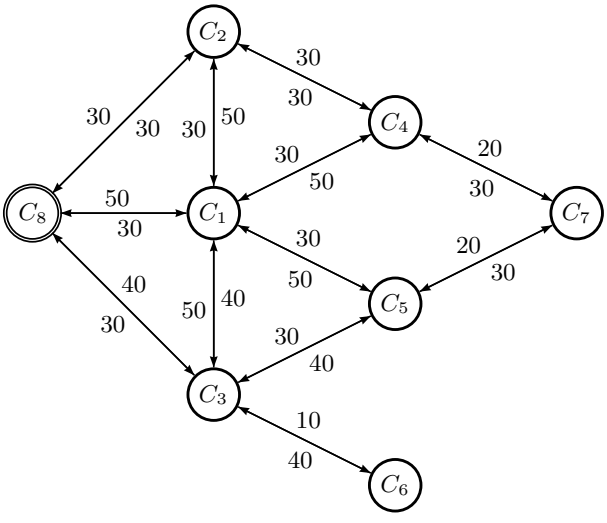

Fig. 1. The network topology used during the experimentations with delay cost of links in ms

As game application, we adopted Quake III Arena, a real-time FPS video game. Quake III Arena is a well-known multiplayer FPS which is, nowadays, still largely played over the Internet. Moreover, this game has been released to the open source, IOQuake III [8]. The game is set in *Free For All* (FFA) mode with a maximum score of 40. The goal of a FFA match is to eliminate as much enemies as possible. Each kill gives +1 point. The match ends when a player reaches the score of 40. As game quality not only depends on network quality but also on players skills, we use autonomous robots (or *bots*) on the clients side. Of course, the bots are configured in the same way, such that player's skills don't impact on game quality.

5 Results and Analysis

5.1 Energy Analysis

In figure 2, we present the energy consumption versus time while using the delay-efficient OLSR. According to the topology presented in figure 1, C_2 and C_3 are

the 1-hop neighbors of the game server which belong to the shortest paths, in terms of delay, of nodes C_4, C_5, C_6 and C_7. Thus, they forward their game traffic, that is why their residual energy decreases faster. Indeed, C_3 forwards the traffic of C_5 and C_6 whereas C_2 forwards the traffic of C_4 and C_7. Until 3000 sec., we can distinguish three groups of nodes: C_2 and C_3 which forward two traffics each, C_4 which forwards one traffic (the one of C_7) and leaf nodes (C_1, C_5, C_6 and C_7) which do not forward any traffic. After 3000 sec., C_2 and C_3 reach 0% of residual energy and shut down. Then, C_1 takes over and forward the traffics of C_4, C_5 and C_7. As a consequence, its residual energy decreases faster. The only path for C_6 to reach the game server is through C_3. However, as this node is out of energy, C_6 cannot reach the game server anymore and has to leave the game. Finally, after 4600 sec., C_1 is out of energy and no other node can reach the game server. These results show that finding routes for traffic according to only the delay metric seriously reduce the game session.

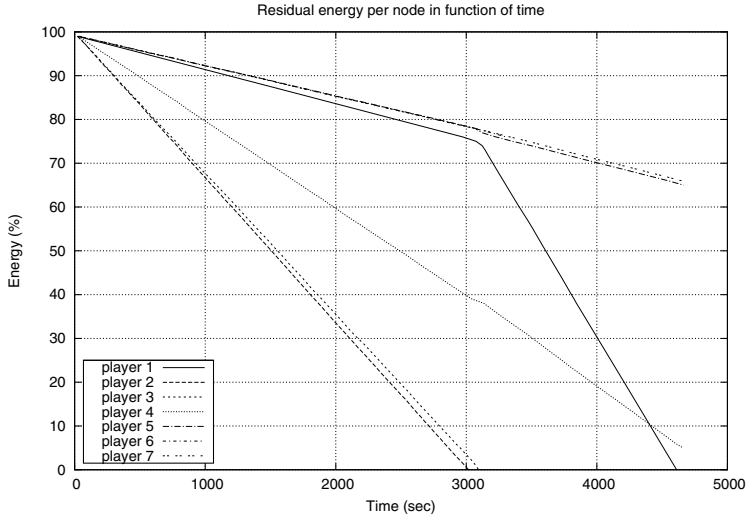

Fig. 2. Residual energy per node in function of time, considering delay-efficient OLSR

In figure 3, we present the energy consumption versus time while using the energy-efficient OLSR. Routes are now computed according to only the energy metric. We can clearly distinguish three groups of curves in this figure. Members of these groups are in fact determined by their position compared with the game server. Indeed, the first three curves are the ones of nodes C_1, C_2 and C_3, which all are 1-hop far from the game server. These three nodes forward game traffic from and to nodes C_4, C_5, C_6 and C_7, that is why their residual energy decrease faster than the other nodes. The second group of curves is composed of nodes C_4 and C_5, which both are 2-hops far from the game server. They fairly forward traffic from and to C_7. Finally, curves of C_6 and C_7 represent the third

group. These two nodes are leaf nodes and do not forward any other traffic than theirselves. Thus, their residual energy decreases slower than all other nodes. This figure shows well that the traffic load is fairly distributed among the nodes of a same group to avoid excessive energy consumption of only one node. Indeed, all the players can play together up to only 3000 sec. against 4000 sec. for energy-efficient routing. Using the energy metric rather than the delay metric increased the game session from 3000 sec. to 4000 sec.

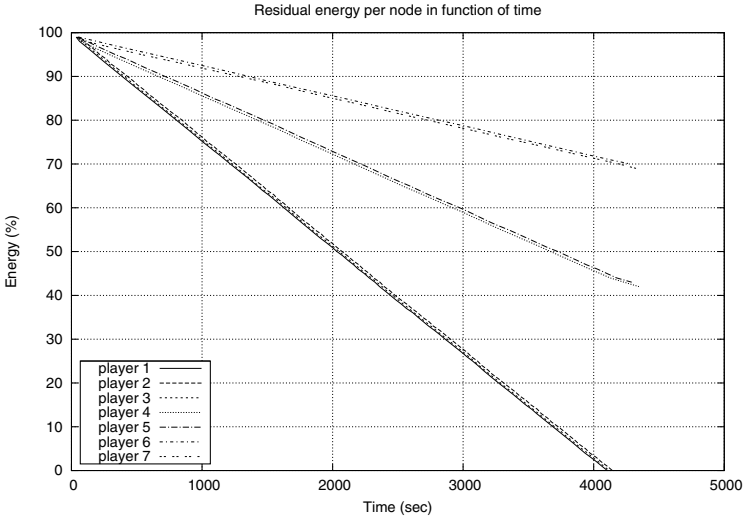

Fig. 3. Residual energy per node in function of time, considering energy-efficient OLSR

In figure 4, we present the energy consumption versus time while using our energy-delay routing protocol (with a threshold of 150 ms). In the beginning, all nodes have the same amount of energy (100%). Thus, it is the delay metric that differentiates the costs of the links. C_4 chooses C_2 and C_5 chooses C_3 to forward their game traffic to the game server because their delay cost is smaller than the one of C_1. The residual energy of C_1 thus decreases slower than the one of C_2 and C_3. If we consider the game traffic from the players to the game server, C_1 can be considered as a leaf node because it does not forward any traffic. However, its residual energy decreases faster than the ones of C_6 and C_7, which are leaf nodes too. Indeed, if we consider the game traffic from the game server to the players, C_8 has a *delay_path_cost* equals to zero (i.e. its factor $f = 0$) because it is the game server. Hence, it only considers the energy metric as link cost. As C_2 and C_3 already forward the game traffics of C_4 and C_5 respectively, and thus consume more energy than C_1, C_8 chooses C_1 to forward its game traffic to the different players. This explains why the residual energy of C_1 decreases faster than the ones of C_6 and C_7. However, after 650 sec., the delay advantage that C_2 and C_3 offer is counterbalanced by their lower remaining energy compared to

the one of C_1. C_4 and C_5 then balance their game traffic between C_1 and C_2 and between C_1 and C_3 respectively, that is why the curve of C_1 becomes parallel to the curves of C_2 and C_3. The same phenomenon appears after 1100 sec. between C_4 and C_5. Indeed, at this time, C_7 starts to balance its traffic between these two nodes. Once more, these two curves become parallel. Our routing protocol tends to reduce the energy consumption on nodes that are close to the game server and have a short delay cost. Moreover, we can see that all nodes are able to play until 4000 sec., which is the same amount of time as with energy-efficient routing protocol.

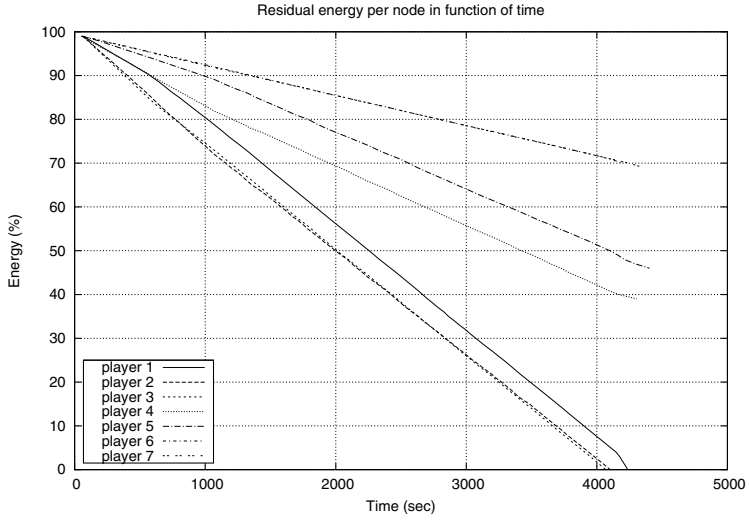

Fig. 4. Residual energy per node in function of time, considering energy-delay OLSR (threshold = 150 ms)

Finally, we plot in figure 5 the duration of a gaming session in function of the routing strategy chosen. Results clearly show that increasing the *delay_threshold* value leads to increase the gaming session. Indeed, the bigger is the *delay_threshold*, the more nodes will take into account the energy metric to compute their paths.

5.2 Score and Fairness Analysis

Figure 6 represents the CCDF (Complementary Cumulative Distribution Function) of score of players with the delay-efficient OLSR. We can distinguish in this figure two groups of curves. The first group is composed of nodes C_1, C_2, C_3 and C_6. Their curves are very close to each others. That means that there is a good fairness between them. Moreover, their probability of ending a match with a score at least equals to 30 is about 70%. The second group is composed of nodes

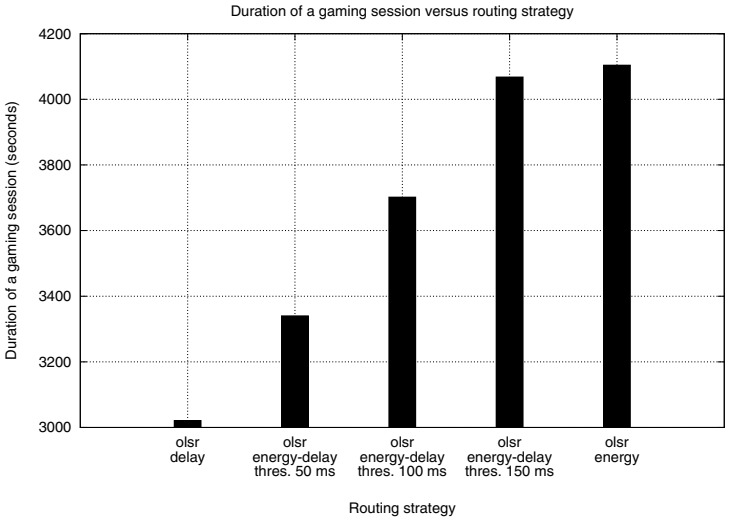

Fig. 5. Duration of a gaming session versus routing strategy

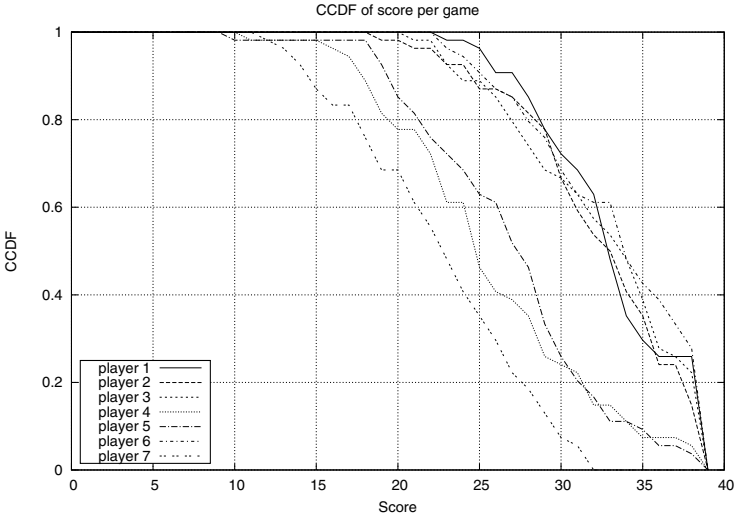

Fig. 6. Score CCDF of each player with delay-efficient OLSR

C_5, C_4 and C_7. Their probability of having a score of at least 30 is very low and their curves are more distant from those of the first group. Indeed, delay-efficient routing selects paths that have the smaller end-to-end delay. Therefore, nodes which are far from the game server or which have higher end-to-end delays are penalized. This kind of routing protocol does not provide fairness among players, so game experience of penalized players may be frustrating.

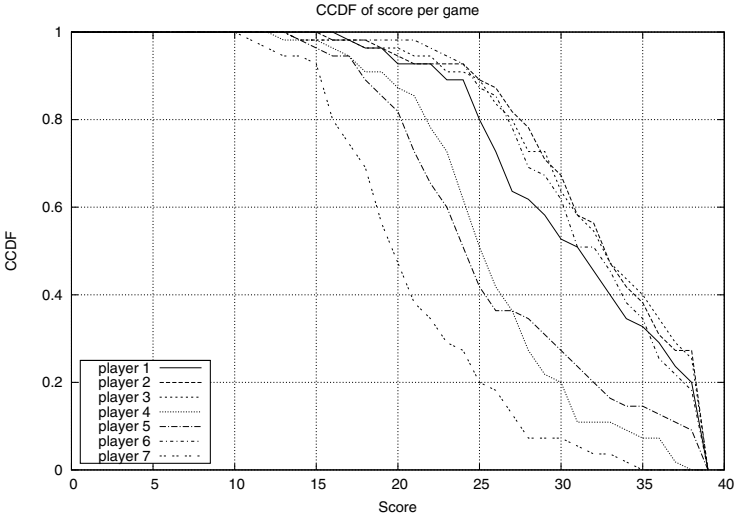

Fig. 7. Score CCDF of each player with energy-efficient OLSR

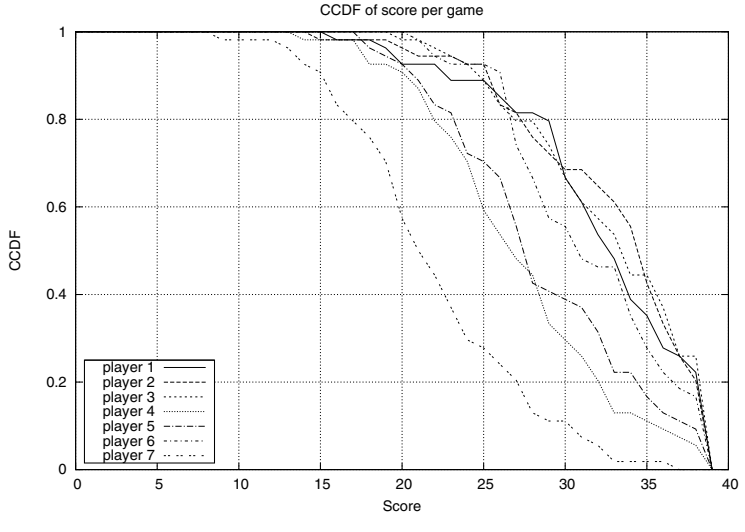

Fig. 8. Score CCDF of each player with energy-delay OLSR (threshold = 150 ms)

Figure 7 represents the CCDF of score of players with the energy-efficient OLSR. Here again, the nodes C_1, C_2, C_3, and C_6 have better probabilities of finishing a match with a good score because they are closer to the game server in term of end-to-end delay. C_7 is still the last one because it has the greater end-to-end delay. However, curves of C_4 and C_5 are closer to each other, which means that fairness between these two nodes is better than with the delay-efficient OLSR.

Indeed, C_4 and C_5 do not only choose C_2 and C_3 respectively to reach the game server but they balance their traffic between C_1, C_2 and C_3 in function of the residual energy. Thus, when they use C_1 as relay, their end-to-end delay increases. Finally, their average end-to-end delay tends to be closer and that is why their score curves are closer. We also see on that figure that the global score of each node is a little worse than with delay-efficient OLSR. This is due to the fact that only the energy metric is taken into account, so sometimes the nodes choose a path with a higher end-to-end delay. However, unfairness between C_1, C_2, C_3, C_6; C_4, C_5 and C_7 is not that much increased.

Figure 8 represents the CCDF of score of players with our energy-delay OLSR (with a threshold of 150 ms). Compared with the two others figures, we can see here that the interval between curves of C_1, C_2, C_3, C_6 and C_4, C_5 is reduced. Indeed, the probabilities to reach a score at least equals to 30 at the end of a match increased for C_4 and C_5 and slightly decreased for C_1, C_2, C_3 and C_6, making the game more fair. Except C_7, which is very far from the game server, our modified OLSR routing protocol enable to increase fairness among players.

Finally, figure 9 depicts the average score of the players in function of the routing strategy chosen. Results show that increasing the *delay_threshold* value leads to slightly decrease the average score. These results confirm the ones presented in figure 5. Increasing the *delay_threshold* value improves lifetime of game session and fairness among players but at the cost of some gaming quality (in particular for players which are close to the game server). On the contrary, reducing the *delay_threshold* value, decreases the lifetime of game session and the fairness among the players but improves the game quality.

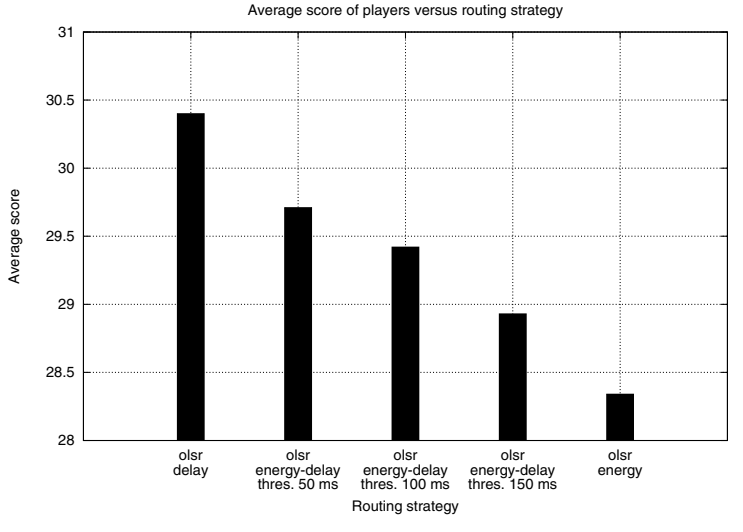

Fig. 9. Average score of player versus routing strategy

6 Conclusions and Future Work

In this paper, we focus on the energy consumption and the game quality of a real-time multiplayer video game session over multihop ad hoc networks. We propose a multimetric routing protocol based on the delay and the energy metrics. It is specialized in client-server based applications because it takes into consideration the position of the nodes compared with the game server.

We study the impact of a delay-efficient protocol and an energy-efficient protocol on the duration and the quality of gaming sessions. We showed that one improves game quality but seriously decreases the network lifetime whereas the other do the contrary. Moreover, no one of them improves the fairness among the players.

We then compare our routing protocol with the two mentioned above. We showed that it enables to find a compromise between game quality and energy consumption and also increases the fairness among the players, depending on its *delay_threshold* parameter. Setting the latter to a value close to the maximum existing end-to-end delay between the game server and the players seems to be a good compromise between lifetime and quality of gaming sessions.

Our future work consist of using a dynamic *delay_threshold* value: the game server can compute it depending on the different end-to-end delay it has with the players and share the computed value to all the players by using the routing protocol signaling messages. Also, we plan to evaluate our proposition in a mobile environment (MANET).

References

1. Clansen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR) (October 2003), `http://www.ietf.org/rfc/rfc3626.txt`
2. Kunz, T.: Energy-Efficient Variations of OLSR. In: International Wireless Communications and Mobile Computing Conference, IWCMC (2008)
3. Mahfoudh, S., Minet, P.: An energy efficient routing based on OLSR in wireless ad hoc and sensor networks. In: Advanced Information Networking and Applications Workshops, AINAW (2008)
4. Benslimane, A., El Khoury, R., El Azouzi, R., Pierre, S.: Energy Power-Aware Routing in OLSR Protocol. In: International Conference on Mobile Computing and Wireless Communication, MCWC (2006)
5. De Rango, F., Fotino, M., Marano, S.: EE-OLSR: Energy Efficient OLSR Routing Protocol for Mobile Ad-hoc Networks. In: Military Communications Conference, MILCOM (2008)
6. Sanchez-Miquel, L., Vesselinova-Vassileva, N., Barcelo, F., Carbajo-Flores, P.: Energy and Delay-Constrained Routing in Mobile Ad Hoc Networks: an Initial Approach. In: 2nd International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN (2005)
7. Regatte, N., Sarangapani, J.: Optimized energy-delay routing in ad hoc wireless networks. In: Proceedings of World Wireless Congress (2005)
8. `http://ioquake3.org/`

9. Feeney, L.M.: An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. In: Mobile Networks and Applications (2001)
10. Badis, H., Al Agha, K.: QOLSR Multi-path Routing for Mobile Ad Hoc Networks Based on Multiple Metrics: Bandwidth and Delay. In: IEEE Vehicular Technology Conference, VTC (2004)
11. Meraihi, A.N., Jacquet, P.: Le Controle du Delai dans le Protocole OLSR. Research Repport (2003)
12. Kaiser, A., Maggiorini, D., Achir, N., Boussetta, K.: On the Objective Evaluation of Real-Time Networked Games. In: Global Telecommunications Conference, GlobeCom (2009)
13. Perkins, C.E., Royer, E.M.: Ad hoc On-demand Distant Vector Routing. In: IEEE Workshop on Mobile Computing Systems and Applications, WMCSA (1999)
14. Kaiser, A., Achir, N., Boussetta, K.: Multiplayer Games over Wireless Ad hoc Networks: Energy and Delay Analysis. In: International Workshop on Ubiquitous Multimedia Systems and Applications, UMSA (2009)