Improving Stability in QoS Routing for Ad-Hoc Networks

Tiago Coelho(✉), António Costa, Joaquim Macedo, and Maria João Nicolau

Centro ALGORITMI, Universidade do Minho, Braga, Portugal
a44048@alunos.uminho.pt, {costa,macedo}@di.uminho.pt, joao@dsi.uminho.pt

Abstract. Routing in mobile Ad-Hoc networks is normally a difficult task, but even more challenging if the network is expected to provide support for audio and video streaming between human-carried devices. Besides normal movement and signal impairment difficulties, those multimedia applications may impose extra end-to-end requirements. The computation of optimized and/or constrained network paths using one or more metric restrictions is called QoS (Quality-of-Service) routing. This paper presents QMRS, a QoS routing protocol for Ad-Hoc networks which aims to support applications with QoS requirements including requirements for the end-to-end delay. This protocol proposes a on-demand multiple routes discovery mechanism that is able to find up to three node-disjoint paths that meet the QoS requirement. Additionally, and for the purpose of guarantee the stability of the routing process, it uses the signal strength of the links between neighbouring nodes to elect the most stable route. Such route will be used as the primary choice to forward traffic. The other discovered routes will be maintained as backup in order to eventually replace the primary route, in case of disruption or degradation of required QoS.

QMRS was implemented in NS-3 and evaluated in comparison with AODV and AMR (an AOMDV-like protocol). Simulation results show significant improvements, with respect to the average end-to-end delay, packet delivery ratio and throughput.

1 Introduction

A mobile Ad-Hoc network is a self-configuring network composed by mobile nodes that communicate directly, while within the appropriate signal range. The nodes have the ability to autonomously create a communications network between them, without assistance from a network infrastructure. All network nodes behave simultaneously like end-systems and routers. They participate in the packets routing process and also host several applications.

The increased diversity and capacity of wireless mobile devices and the simultaneous evolution of multimedia applications has created the need to propose and evaluate strategies to provide QoS guarantees for end-to-end traffic, in order to comply with requirements requested by such applications (bandwidth, end-to-end delay, jitter, etc). Due to the nodes mobility, the network topology is highly

N. Mitton et al. (Eds.): ADHOCNETS 2014, LNICST 140, pp. 121–133, 2014.
DOI: 10.1007/978-3-319-13329-4_11

dynamic and unpredictable failures may occur in established paths. Moreover, the wireless communications medium is shared between the neighbouring nodes. These factors mean that routing traffic with QoS requirements constitutes an even greater challenge in Ad-Hoc networks than doing it in fixed networks.

In this work, a proposal is made for an on-demand routing protocol that is able to find multiple disjoint paths in Ad-Hoc networks, that can meet end-to-end QoS requirements. The end-to-end delay was chosen as an example. The protocol, called QMRS (Ad hoc QoS On-Demand Multipath Routing with Route Stability) determines route stability based on signal strengths observed along the path and uses it as the main criteria to choose the best route. By keeping more than one alternative feasible path, stability is also preserved in case of failures. QMRS was implemented and tested using the NS-3 simulator.

The rest of the paper is structured in five more sections. Section 2 presents related work, namely some routing protocols for mobile adhoc networks. Section 3 describes the proposed protocol, detailing its most important mechanisms. Section 4 describes the implementation of the proposed protocol in the NS-3 simulator. Section 5 shows the obtained results and Section 6 the conclusions and further work.

2 Related Work

Routing protocols are usually classified into two main categories: proactive (table-driven) or reactive (on-demand). Proactive routing protocols (eg, Optimized Link State Routing Protocol - OLSR [1]) aim to keep all routing tables always updated. They provide low latency for packet forwarding since routes are always available, but they need to constantly exchange control messages even without any traffic. With expected high mobility and energy limitations of nodes, such control information overload is seen as a possible limitation.

Reactive routing protocols (eg, Ad hoc on demand Distance Vector - AODV [2]) only compute routes at the request of a node. The route discovery process is initiated only when a source node wants to transmit to a given destination, and there is no valid route in its routing table. In this way, there are less control messages on the network, thus allowing the devices to save processing resources and, above all, energy. Among reactive routing protocols, AODV is usually one of the most cited. However, during the route discovery phase, AODV only finds a single route to the destination. This means that if the existing route breaks, the route discovery process must be restarted.

To overcome this problem, and reduce the need for frequent discovery of routes from the same source to the same destination, some protocols were proposed that are able to find multiple paths within the same request (eg, Ad hoc On-demand Multipath Distance Vector - AOMDV [3]). AODMV is a variant of the protocol AODV that discovers various alternative routes but without any specific quality of service concerns. However, many other routing protocols exists that aim to provide such QoS guarantees in mobile Ad-Hoc networks.

Y. Hwang and P. Varshney proposed a protocol (An Adaptive QoS Routing Protocol with Dispersity for Adhoc Networks - ADQR [4]) that can discover

multiple disjoint paths per route request. Based on the bandwidth information collected during the discover phase, resource reservations must then be requested on all routes found. Data is transmitted on all reserved paths. The protocol monitors the network to detect topology changes and update routes before they become unavailable. With the discovery and maintenance procedures, ADQR aims to significantly improve the network performance and to provide end-to-end QoS in mobile Ad-Hoc networks. Nevertheless, ADQR doesn't solve the packet reordering problem inherent to load balancing traffic between multiple paths. Furthermore, to process route requests it needs to store topology state information in each node. Finally, the proactive monitoring that is required for normal operation originates an overhead of control packets.

Qi Xue and Aura Ganz proposed a QoS routing protocol (Ad hoc QoS on-demand routing - AQOR [5]) based on AODV with resource reservation. The protocol is able to estimate the available bandwidth and measure the end-to-end delay, in order to provide QoS. It includes also bandwidth reservation and route recovery mechanisms. To avoid having to deal with the release of reserved resources in each node, in case of connection failure, the resource reservation is performed only temporarily. The route recovery procedure includes the detection of broken links, the verification of the required QoS and a destination-initiated recovery process.

Nityananda Sarma and Sukumar Nandi proposed the SMQR (Route Stability based Multipath QoS Routing) [6] protocol for ad-hoc networks. This protocol is able to discover multiple disjoint routes with higher stability. The discovered routes meet the requirements of a maximum end-to-end delay and a minimum effective transfer rate, in order to support real-time applications. Based in simulation results, the authors claimed improvements when compared with AODV protocol in terms of packet delivery ratio, average end-to-end delay, maximum delay variation and effective throughput.

Shun Liu and Jian Liu proposed a QoS routing protocol (Delay-aware multipath source routing protocol)- DMSR [7]) to support real time applications in Ad-Hoc networks. These applications are throughput and delay sensitive. DMSR is a multipath routing protocol that uses end-to-end delay as a metric. At each node, local information is collected and the delay experienced in the node is computed. This metric is used to select the best path. It combines the number of neighbour nodes, the contention time, and the number of packets in queue.

3 A Proposal for QoS Routing in Ad-Hoc Networks

This section presents the proposed protocol called QoS Multipath Routing with Route Stability (QMRS), a reactive protocol based on AODV. The new features introduced enable the discovery of multiple paths, reach QoS requirements and optimize the stability of the routing process. The QMRS protocol enables the discovery of up to three node-disjoint paths that meet an end-to-end delay requirement. Among the paths found, the most stable one is selected as the active path, while the others are maintained as alternatives. If the paths found have the same stability, the one with the lowest end-to-end delay is selected.

The discovery of multiple node-disjoint alternative paths allows a faster reaction when a fault occurs on the active path. In this case, the source node will react to the fault notification by changing the active path to one of the alternatives. The probability for existence of a valid alternative is high due to the algorithm used for finding paths. This algorithm only considers as alternative paths those without any nodes in common between the source and the destination. If the paths found contained disjoint links instead of disjoint nodes, the movement of only one node could break all alternative routes between source and destination. In that case the source node would need to restart the route discovery process again. Thus, the discovery of multiple node-disjoint alternative paths decreases the amount of control traffic on the network and consequently the end-to-end delay for data flow transmission. Figure 1 illustrates the concept of node-disjoint paths. With such topology gathered during the route discovery process, it would be possible to discover three node-disjoint paths: *source-A-D-destination*, *source-B-destination*, *source-C-E-destination*. As can be verified, the same intermediate node can only belong to one of the paths found between the source and destination, and can never belong to other alternative paths found and stored in the routing table.

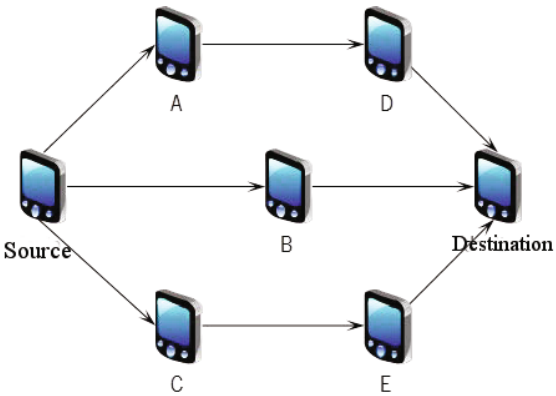

Fig. 1. Multiple node-disjoint alternative paths

As described, QMRS begins by discovering multiple node-disjoint paths between source and destination that meet the QoS requirement. Then, and to improve stability, QMRS uses the received signal strength to select the best route for transmission. If two or more discovered paths present similar values for received signal strength metric, end-to-end delay is used to select the best route.

A route maintenance mechanism is applied on all paths found between the source and destination nodes, in order to periodically check if the QoS guarantees are maintained. The values of the used metrics (end-to-end delay and

received signal strength) are updated. Then, a route change may occur, to one of the valid alternative routes stored in the routing table. With these mechanisms, the proposed protocol enables an efficient data transmission providing QoS guarantees.

3.1 Maintaining State Information Between Neighbours

Periodic exchanges of *Hello* messages occur between the nodes within the transmission range. The reception of these *Hello* messages indicates that the neighbours are available and accessible, information that is required for the route discovery and maintenance procedures. Through the *Hello* messages, the received signal strength and the delay for each neighbour are also updated. The signal strength on reception is directly taken from the physical layer via a cross-layer interaction. The delay for each neighbour is the time since the packet is queued until it reaches the neighbour. This time includes the queue time, contention and media access time, transmission time and propagation time. The periodic exchange of *Hello* messages is necessary to maintain the values of these metrics updated and enable intermediate nodes to answer route requests for its neighbours. When a neighbour misses *Hello* messages during a given time interval, a route recovery process must be initiated.

3.2 Route Discovery

The route discovery process is based on AODV protocol. Significant changes were done in order to introduce additional mechanisms to enable the discovery of multiple node-disjoint paths that meet an end-to-end delay requirement. *End-to-end delay* is an additive metric that results from the sum of all the delays obtained in the nodes that constitute the path. For each path found, QMRS algorithm also computes the *signal strength*, a concave metric that is computed as the minimum value of the received signal strength in all links that make up the path.

When a node wishes to transmit data to some destination node, and doesn't have a valid route for that destination in its routing table, it has to start the process of route discovery. This process consists on a broadcast of a *Route Request (RREQ)* packet to all neighbouring nodes. The *RREQ* packet contains information about the source node (*Originator IP Address* and *Originator Sequence Number*) and the destination node (*Destination IP Address* and *Destination Sequence Number*). The sequence numbers are increased by the source node, every time a *RREQ* packet is sent. Upon receiving a *RREQ* packet, each intermediate node can verify if the information it contains on its routing table is still updated or already obsoleted, in relation to the information hold by the request originator. The mentioned fields, along with the *ID* field and advertised *hop count* are essential to distinguish between successive requests and avoid routing cycles. Table 1 shows the structure of *Route Request (RREQ)* and *Route Reply (RREP)* packets.

Table 1. RREQ and RREP Packet Fields

RREQ ID
Destination IP Address
Destination Sequence Number
Originator IP Address
Originator Sequence Number
hopCount
firstHop
RxSignalStPath
delayAcc
delayPath
delayReq

(a) RREQ Packet

Destination IP Address
Destination Sequence Number
Originator IP Address
hopCount
firstHop
RxSignalStPath
RxSignalStPathToDst
delayPath
delayReq

(b) RREP Packet

When receiving the *RREQ* packet, neighbouring nodes increment the *hopCount* field and introduce their own address in the $firstHop$ field. This field is used to verify if the path found is a node-disjoint path. When a node receives the *RREQ* packet for the first time (verifiable by the $RREQ\ ID$ and $Originator\ IPAddress$ fields, that uniquely identify the request) and doesn't have a valid route to the requested destination, it broadcasts the *RREQ* again through the network in order to find the destination node or an intermediate node that has a valid route to the destination node. The route is considered valid only if the intermediate node contains in its routing table a route to destination node with a sequence number (*Destination Sequence Number*) greater than or equal to the one contained in the *RREQ* packet received. When an intermediate node responds to a request, made by a source node, it must also send a packet called *Gratuitous Route Reply* to inform the destination node of the route to the source node. After retransmitting the *RREQ* packet, the intermediate node waits a period of time for a valid response. The reception of a *Route Reply (RREP)* packet validates the route to destination.

When receiving the *RREQ* packet, the intermediate nodes must also update the routing entries relative to the source node. The intermediate node begins to check if the sequence number of the source node contained in the packet is greater than or equal to the one in the routing table. In case of being equal, and if it is a node-disjoint path, then it is also checked whether the number of routes contained in the routing table is less than three (the protocol stored up to three node-disjoint paths). After these checks performed, the routing table entry to the source node is updated. If the sequence number is greater than the one in the routing table, the routes to the source node are removed and replaced by the new one found.

During the route discovery, a mechanism for admission control is used. Whenever there is non compliance with maximum end-to-end delay required by the

application (*delayReq*), the *RREQ* packet is discarded. To verify that the QoS requirement is being met during the route discovery, a node calculates the delay that occurs in the node and adds it to the one contained in the *RREQ* packet (*delayAcc*). A comparison is then performed between the calculated value and the maximum end-to-end delay required. If the accumulated end-to-end delay isn't less than the requested one ($delayAcc < delayReq$) the packet is discarded.

When the path already traversed meets the QoS requirement contained in *RREQ* packet (*delayReq*), the accumulated end-to-end delay (*delayAcc*) and the minimum received signal strength (*RxSignalStPath*) from the path so far is updated on the *RREQ* packet. The signal strength is obtained directly from the physical layer, through a cross-layer interaction. The value obtained on receiving node (*RSSRead*) is compared with the value stored in the *RREQ* packet. If the value read is less than the value stored in packet, it is a new minimum, and must replace the value stored in the packet. Otherwise the value remains unchanged (equation 1). With the metrics updated, the *RREQ* packet is retransmitted by broadcast, to continue the search for a route to the destination.

$$RREQ.RxSignalStPath = min(RREQ.RxSignalStPath, RSSRead) \quad (1)$$

The reply to the route request, whether made by the destination node or by an intermediate node that has a valid route to the destination, is performed by sending a *Route Reply (RREP)* packet. The information in the *RREQ* packet, updated during its retransmission by intermediate nodes along the path, contains the final end-to-end delay (*delayPath*) and the minimum received signal strength (*RxSignalStPath*) of the path found. This information will be placed in the response *RREP* packet, and this packet will be sent in unicast directly to the source node using the reverse of the route traversed by the *RREQ* packet. In this way, the intermediate nodes validate the routing table entries to the destination node, while retransmitting the RREP packet to the source node. The information on end-to-end delay and the minimum received signal strength, is also inserted into the respective routing table entry and validated by the intermediate nodes. Thus if a node receives a request from another source node to that destination, it can check if the end-to-end delay can be met, as well as inform on the stability of this path.

3.3 Path Selection

Among all node-disjoint paths found that meet the end-to-end delay requirement, the most stable one is chosen for transmission. The stability of a path is based on the minimum received signal strength observed on all its links (*RxSignalStPath*). This is the primary criteria for route selection. The best route is the one with higher *RxSignalStPath* value. Only in case of more than one route have the highest stability, the preference is given to the path with the lowest end-to-end delay.

3.4 Route Maintenance

As a consequence of a route discovery operation, nodes in discovered paths store routes to the destination node and to the source node on their routing tables. Topology changes can however occur due to, for instance, node mobility. Therefore routes can became unavailable and information stored on the routing table obsolete. Whenever a node in a path verifies that the the next hop node is not in transmission range, it must send a *Route Error (RERR)* packet to notify the source node of a link failure. This *RERR* packet is forwarded by all nodes on the path back to the source. Before forwarding the packet, each node must remove the invalid route from its routing table. After receiving a *RERR* packet, the source node checks if the broken route is the active one in use for the destination, and changes the active route to the best stable alternative available on its routing table. When no alternatives exist, a new discovery procedure must be initiated. If the broken route is an alternative route not currently in use, it is simply removed from the routing table without any further action.

A QoS verification procedure must also be carried on all paths discovered and stored in the routing tables. When the end-to-end delay restriction is violated, routes are also removed. Two new packets are used to check for metric changes in the path: $InspectPath$ and $ReplyInpectPath$. The verification process is similar to the one used in the discovery phase. The $InspectPath$ is forwarded along the path accumulating the new values of delay and signal strength. All nodes in the path use the information carried in the $InspectPath$ to update the routing table with the new metric. If the restriction is not met, an *RERR* packet is sent back to the source, notifying all intermediate nodes that the route is no longer usable. Each node that receives the *RERR* packet removes the invalid entry.

4 Implementation

The Network simulator NS-3 [8] was chosen as the implementation tool, among other well known open source alternatives, like NS-2 [9] and OMNeT++ [10]. While NS-2 is still under heavy usage, NS-3 is currently presented as its future replacement, eventually with better performance [11] and more efficient resource usage. In current NS-3 modules we can find some Ad-Hoc routing protocol implementations (like AODV, for instance) but none related to multipath and/or QoS routing proposals. This drawback was in fact faced as a challenge towards an effective move to NS-3. A previous implementation of Ad Hoc On-Demand Multipath Distance Vector (AOMDV) [3], available for NS-2, was used as inspiration and a reference for our implementation on NS-3.

4.1 Data Structures

The routing table structure is based on the structure used in AODV implementation, carefully modified to include multiple entries for the same destination.

Routing entries are stored in a map ($std::map < Ipv4Address, RoutingTableEntry > m\_ipv4AddressEntry$) that uses the IP Address as key. Values associated with the key are of type $RoutingTableEntry$, that contain the data fields and methods associated with a route entry. For each entry, the end-to-end delay and the minimum signal strength measured are also stored. The following fields are used: *destination*, *sequence number*, *advertised hop count*, *next hop*, *last hop*, *hop count*, $rxSSPath$, *delay*, $lifetime$ and a *list of alternative routes*. The list of alternative routes is a vector ($std::vector < QmrsRoute > m\_routeList$) that stores, for each alternative route, the following fields: *next hop*, *last hop*, *hop count*,
$rxSSPath$, *delay* and $lifetime$.

4.2 Neighbour State Information

Neighbour nodes must exchange *Hello* messages between them, stating that they are available and accessible. This information is very useful for the route discovery and maintenance procedures. *Hello* messages are in fact special *RREQ* sent from the node address to the node address and with $TTL = 1$. For each received message, nodes must get the received signal strength from physical layer, through a cross-layer call. The method $GetRxPowerDbmPhy(receiver)$ available on class $RoutingProtocol$, initiates that call that will invoke the $GetRxPowerDbm$ on the *yans-wifi-phy* module. This file was changed to provide the required value.

All methods that process messages must also keep a $delayNode$ value updated. The value is used to update the end-to-end cumulative delay of route requests. The value is computed using the NS-3 packet "tags" feature that allows for the sharing of 20 bytes between layers. The tag is created with the simulation time obtained by calling $tag.Set(Simulator::Now().GetMicroSeconds()$ and $packet-> AddPacketTag(tag)$ in sequence. When the tagged packet arrives at the physical layer for transmission, the time delay is computed and added to the physical transmission delay that can be obtained by calling the method $txDuration.GetMicroSeconds()$. The $delayNode$ value is therefore very accurate.

4.3 Route Discovery

Route discovery functions are implemented with the exchange of *route request messages (RREQ)*(see table) 2a) and *route reply messages (RREP)*(see table 2b). A few details on *RREQ* send and receive methods are therefore provided here.

The *sendRequest()* method is called to broadcast a *RREQ* message on the network. The request must include the delay restriction $delayReq$ that is set using the $SetDelayReq()$ method. In order to get the correct end-to-end delay, the $delayNode$ value must be obtained from the node by calling $GetAverageDelayPhy()$ and set in the *RREQ* message using $SetDelayAcc()$. Before the transmission, packet is tagged with current simulation time as previously described. Each node receives the *RREQ* message with the method $recvRequest()$. The path delay value must be

updated in the message using $rreqHeader.SetDelayAcc(rreqHeader.GetDelay Acc() + delayNode)$.

4.4 Route Maintenance

A source node can proceed with data transmission on an alternative route after receiving a route error notification. But alternate routes can only be fast effective replacements in case they are kept up to date. The end-to-end delay and the minimum receive signal strength metrics must be updated regularly, and the end-to-end delay restriction must be verified. For this maintenance procedure, two new messages were created: *InspectPath* and *ReplyInspectPath*. A periodic call to a method *SendInspectPath*() must be issued by the on every node-disjoint path kept in table. Nodes in the path, receive the message using the method *RecvInspectPath*() to accumulate the new delay and signal strength values. A reply message is sent using *SendReplyInspectPath*() only when the delay restriction stay valid. The *ReplyInspectPath* message is received by all nodes in the path from the destination to the source, using the method *RecvReplyInspectPath*(), and is also used to update the metric values in the routing table entries for the destination.

5 Results

The Network Simulator 3 (NS-3) [8] was used to evaluate QMRS. As a starting point, a multipath routing protocol, called AMR (AdHoc Multipath Routing-protocol) was implemented and used in result analysis. The implementation of AODV already included in NS-3 was also used as a comparative reference.

5.1 Simulation Scenario

A total of 60 simulations were run for each protocol in all scenarios. Therefore, results presented in figures 2, 3 and 4, for each node speed, show the average of the obtained 60 values and the correspondent 95% confidence interval. At simulation start, all 80 nodes are placed randomly in an area of $600m$x$1500m$. Nodes then move at a maximum speed of 0 to 10 m/s, randomly in that area, according to the random waypoint mobility model. All nodes have a transmission range of $160m$. Table 2 shows all parameters used in simulations.

During simulation time ($200s$) a total of 15 CBR (Constant Bit Rate) flows were generated, between randomly selected pairs of source and destination nodes. The UDP protocol carried CBR packets of a 64 *bytes* in size, at transfer rate of $2Kbps$. Three metrics were calculated and used in results analysis: the observed end-to-end delay (in miliseconds), the packet delivery ratio (in percentage) and effective transmission rate (in *kbps*).

Parameter	Value
Area	$600m$x$1500m$
Number of nodes	*80*
Mobility model	*Random Waypoint*
Node position	*Random*
Transmission range	$160m$
Traffic type	*Constant Bit Rate*
Packet size	*64 bytes*
Number of flows	15
Transmission rate	$2Kbps$
Wifi specifications	*IEEE 802.11b, freq. 2.4Ghz Transfer rate up to* $2Mbps$
Simulation time (s)	200

Table 2. Parameters used in simulations

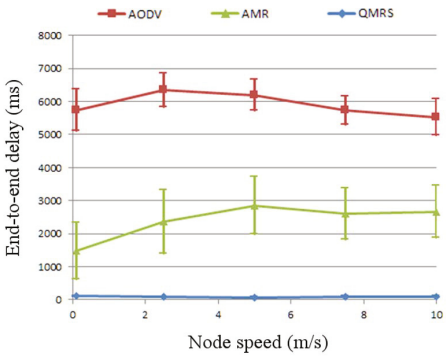

Fig. 2. Results: end-to-end delay

5.2 End-to-end Delay

Figure 2 shows the results obtained for the end-to-end delay metric. QMRS presents and average end-to-end delay value between 0 and 150 ms, while AODV exhibits much higher values. AODV has no concerns regarding delay and it only discovers one route per request, having no way to deal with failures except by restarting the discovery process. On the other hand, AMR computes multiple paths, and is therefore more stable when disruptions occur. It selects routes based on stability without considering delay. It shows less average end-to-end delay than AODV, but higher than QMRS. QMRS is the only protocol in this set that uses delay as a metric.

5.3 Delivery Ratio and Transmission Rate

Figure 3 shows the results obtained for the packet delivery ratio metric. The average delivery ratio for QMRS, measured at the destination node, is in the 70% to 80% range. Values for AODV are placed between 10% and 20% and for AMR between 45% and 60%. Figure 4 shows the exact same tendency for the effective transmission rate observed. QMRS presents values around the $2kbps$, while AODV remains under $1Kbpbs$. AMR exhibits intermediate values, above $1.5kbps$ and below $2kbps$.

QMRS goal is to ensure path stability in the first place, while respecting the end-to-end delay requirements. Nodes that belong to its route paths are therefore less congested. AODV however, considers no delay restrictions and may choose paths that are more congested. Since it only discovers one route per request it has to deal with failures. AMR shows intermediate values because it uses multiple node-disjoint paths, but it doesn't consider any delay metric.

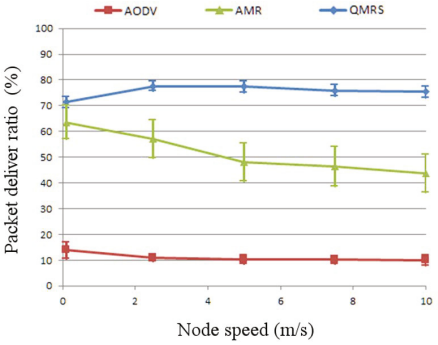

Fig. 3. Results: packet delivery ratio

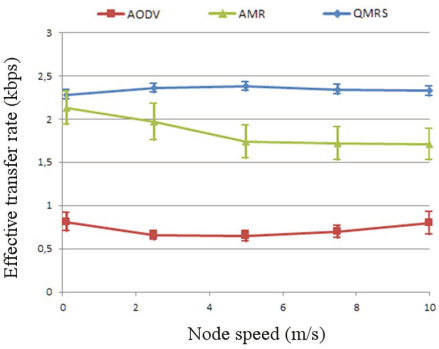

Fig. 4. Results: effective transfer rate

6 Conclusion

In this paper a new protocol called Ad hoc QoS On-Demand Multipath Routing with Route Stability (QMRS) was proposed. QMRS provides mechanisms to discover, in a single discovery attempt, multiple node-disjoint paths between source and destination nodes that can satisfy the specified end-to-end delay restriction. The source node chooses the most stable path for transmission, in order to maintain the route feasible for longer periods of time, without the need to change to an alternate path or re-initiate the route discover procedure. Only in presence of multiple routes with equal stability, the lower end-to-end delay is used as a second criteria of choice. The discovery of multiple alternative node-disjoint paths allows the source node to use an alternative route in case of a failure in the main route eventually caused by node movements. Route stability is based on measured signal strength, which is computed as the minimum value observed in all links that constitute the path. The path with the higher value is the most stable.

The proposed protocol was implemented in Network Simulator 3 (NS-3). Simulation results were obtained and compared with the standard implementation of AODV included in NS-3, and also with a variant of the AODV capable of discovering multipath node-disjoint routes, implemented from scratch on NS-3. Results show that QMRS, with its discovery/maintenance/recovery and delay verification mechanisms, provided lower end-to-end delay in all data transmissions and higher packet delivery ratio and effective transmission rate.

Future work includes further simulation analysis. New traffic models, besides simple CBR traffic, and new simulation scenarios, should also be considered. Alternative metrics may also be used. Another goal is to evaluate the usage of QMRS within a class-of-service model.

Acknowledgments. This work has been suported by FCT - Fundação para a Ciência e Tecnologia in the scope of the project: PEst-OE/EEI/UI0319/2014.

References

1. Clausen, T., Jacquet, P.: RFC 3626 - Optimized Link State Routing Protocol (OLSR) (2003)
2. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on demand distance vector (AODV) routing (RFC 3561). IETF MANET Working Group (August 2003)
3. Marina, Mahesh K., Das, Samir R.: Ad hoc on-demand multipath distance vector routing. Wireless Communications and Mobile Computing **6**(7), 969–988 (2006)
4. Youngki Hwang, Y.H., Varshney, P.: An adaptive QoS routing protocol with dispersity for ad-hoc networks. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (2003)
5. Xue, Q., Ganz, A.: Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks (2003)
6. Sarma, N., Nandi, S.: A Route Stability Based Multipath QoS Routing (SMQR) in MANETs. In: 2008 First International Conference on Emerging Trends in Engineering and Technology (2008)
7. Liu, S., Liu, J.: Delay-aware multipath source routing protocol to providing QoS support for wireless ad hoc networks. In: 2010 IEEE 12th International Conference on Communication Technology, pp. 1340–1343. IEEE (November 2010)
8. Henderson, T.R., Roy, S., Floyd, S., Riley, G.F.: Ns-3 Project Goals. In: Proceeding from the 2006 Workshop on ns-2: the IP Network Simulator - WNS2 '06, p. 13 (2006)
9. Fall, K. Varadhan, K.: The network simulator (ns-2). http://www.isi.edu/nsnam/ns (2007)
10. Varga, A.: Microsoft Visual, Remote Omnet, and Statistical Synchronization Method. The OMNeT++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference ESM'2001, vol. 42, pp. 319–324 (2001)
11. Weingartner, E., Vom Lehn, H., Wehrle, K.: A performance comparison of recent network simulators, pp. 1–5. IEEE (2009)