Adaptive Localized Active Route Maintenance Mechanism to Improve Performance of VoIP over Ad Hoc Networks

Adriana Sofia Otero[1], Mohammed Atiquzzaman[2]
[1]University of Oklahoma, School of Electrical and Computer Engineering, Norman, OK
[2]University of Oklahoma, School of Computer Science, Norman, OK
Email: {sofyotero, atiq}@ou.edu

Abstract—**An important characteristic of a mobile ad hoc network routing protocol is connectivity maintenance. The protocol's ability to maintain routes and react to topology changes due to link failures affects the delay, packet loss, and throughput of real-time applications. However, the time required to recover from a link failure in the current AODV protocol is too long for real-time applications. The objective of this paper is to develop a mechanism to allow fast response to link breaks to enable real-time communications, such as Voice over IP. We propose AODVM-ALARM, an adaptive localized route maintenance mechanism, over an existing ad hoc multipath routing protocol. The mechanism dynamically monitors and maintains on-going communications by utilizing special control messages and packet analysis techniques to enable fast response to link breaks. The mechanism can recognize, respond and recover quickly from link breaks due to topology changes. AODV-ALARM has been tested in an ad hoc experimental testbed, and compared with AODV-UU and AODVM protocols. Results show that AODVM-ALARM significantly improves the response time to link failures and achieves an overall improvement in the network performance in terms of the failover delay, failover loss and throughput when compared with the previous ad hoc routing protocols. AODVM-ALARM which is easily deployable, fault tolerant and have a fast response to topology changes can be used to rapidly establish VoIP communications, playing a significant role in emergency response for disaster recovery when the network infrastructure might be broken.**

Index Terms— **AODV, protocol, route maintenance, mechanism, link failure, routing.**

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) is a rapidly deployable network consisting of mobile nodes connected by wireless links that do not rely on any preexisting infrastructure support [1]. Each of these nodes can act as both an end host and a router at the same time, which allows the transmission of messages via multiple hops. With the increasing proliferation of mobile devices, such as PDAs, laptops and mobile phones with wireless networking support, the demand for real-time applications, such as voice and video (VoIP or VVoIP) transmissions, is increasing in wireless environments. MANETs are envisioned to be used in various scenarios ranging from networks inside offices to spontaneous networks for providing emergency services in the event of natural disasters (such as earthquakes, hurricanes, fire, floods), military conflicts, or other events that cause the existing infrastructure to be destroyed [2]. These Mobile Ad Hoc networks must be able to handle link failures related to *topological changes* that affect the connectivity between network nodes. Therefore, an appropriate Ad Hoc routing protocol is needed to cope with sudden changes in the network that can degrade the quality of the voice transmission.

In a mobility environment, the *Ad Hoc On-Demand Distance-Vector* (AODV) [3] routing protocol and its multipath extension, the Ad hoc On-demand Distance Vector Multipath (AODVM), suffer from connection interruptions because the frequency of the hello messages *is too small* (one per second) *to be able to detect and respond to link breaks fast enough* to allow real-time communication to continue uninterrupted during link breaks. The AODV and AODVM protocols require a considerable amount of time, two and five seconds respectively, to detect a link failure and discover a new route to the destination. Our *objective* is to develop a new AODV-based routing protocol that significantly improves the detection and response time to reroute traffic after a link failure.

In an effort to improve link failure detection and response time, we propose the implementation of an Adaptive Localized Active Route Maintenance (ALARM) mechanism over a *multipath* ad hoc routing protocol. ALARM is based on monitoring the on-going communication by utilizing special surge-hello messages and packet analysis techniques.

Few studies have reported on the analysis of quality issues to improve real-time communication, such as VoIP over MANETs. Glaucia et al. [4] presented a simulation-based performance analysis of four typical ad hoc routing protocols, which include the DSDV (Destination-Sequenced Distance-Vector) and three on-demand routing protocols: AODV, DSR (Dynamic Source Routing) and TORA (Temporally-Ordered Routing Algorithm) in terms of packet delivery rate, end-to-end delay, and routing load. The authors concluded that the reactive protocols, AODV

Manuscript received June 6, 2010; revised November 11, 2010; accepted January 13, 2010.

© 2011 ACADEMY PUBLISHER
doi:10.4304/jcm.6.1.68-78
Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/13396

and DSR, outperform the proactive protocols and recognized that AODV presents an adequate behavior for videophone applications. Karoly et al. [5] compared AODV, DSR, DSDV, and OLSR to investigate the performance impact on real-time applications. They concluded AODV showed a decent performance for real-time traffic when compared to the other three protocols. Binod et al. [6] used simulation to investigate the deployment of VoIP traffic using the AODV routing protocol in terms of jitter, one-way delay, frequency of service interruptions and durations of the interruptions. They pointed out that the AODV protocol is not fault tolerant, and that it triggers a new route discovery every time a route fails.

In order to address this issue several multipath extensions have been proposed, such as AODVM (Ad hoc On-demand Distance Vector Multipath) [7], AOMDV (Ad hoc On-demand Multipath Distance Vector) [8] and NDMR (Node Disjoint Multipath Routing) [9]. They differ in the way route discovery is performed. We are particularly interested in AODVM which offers high fault tolerance due to its node-disjoint routes. It is able to detect and maintain multiple node-disjoints for each destination which makes it more reliable.

On the basis of these observations, we propose the implementation of a new strategy, the ALARM mechanism, over a multipath ad hoc routing protocol to provide a faster response time to link failures and make it more reliable for VoIP communications. We have incorporated the multipath extension AODVM [10] from the University of Bern to the existing single path AODV-UU [11] from Uppsala University to provide higher reliability with the multipath capability when transmitting real-time traffic. Our proposed protocol, AODVM-ALARM, *differs* from the existing ones in the way it handles the link failure detection and maintains the active route. Instead of detecting link failures by broadcasting hello messages, AODVM-ALARM unicasts surge hello messages along the active route that, in conjunction with packet analysis techniques, monitors the ongoing communication to respond quickly to topology changes.

The main *contribution* of this paper is the design, development and implementation of a new link failure detection mechanism over a multipath ad hoc routing protocol that dynamically maintains the connectivity between the nodes along the active route in an efficient way since it does not require extra control packets to monitor the ongoing communication in a complete VoIP session. The mechanism significantly improves the performance of VoIP in MANETs in terms of delay, packet lost and throughput.

The remainder of this paper is organized as follows. Section 2 provides background research about ad hoc routing protocols including its classification, the AODV and AODVM routing protocols and overview of Voice over IP which is the type of traffic we focus on in this paper. Section 3 presents the details of our proposed ALARM mechanism. Section 4 describes our experimental setup and Section 5 shows the results of performance comparison of the new proposed AODVM-ALARM protocol with AODV and AODVM. Concluding remarks are given in Section 6.

II. BACKGROUND

A. Types of Ad Hoc Routing Protocols

The routing protocols for Ad hoc networks are broadly classified as proactive, reactive and hybrid [12] depending upon whether a route is updated continuously or acquired only when needed. Proactive protocols perform well in static scenarios, whereas reactive ones *show better performance in dynamic environments.*

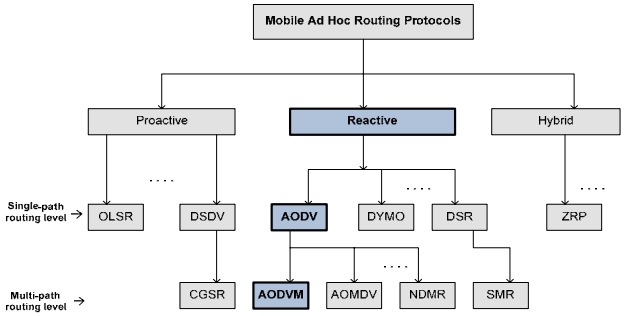


Fig. 1. Classification of Ad Hoc routing protocols

Fig. 1 shows the classification of the Ad hoc routing protocols into the three categories. Our targeted category includes reactive single path AODV and multipath extension AODVM routing protocols. These were the protocols selected because of AODV's adequate performance in simulations and static environments for real-time traffic when compared to other protocols and AODVM's ability to improve fault tolerance since it maintains more than one route per destination. They are discussed in the following sections.

B. AODV

*Ad hoc On-demand Distance Vector (*AODV) [13] is a reactive protocol defined in RFC 3561 [3] that combines mechanisms like hop-by-hop routing, periodic beacons, and sequence numbers to guarantee loop-free routing of the DSDV and the on-demand route discovery and maintenance mechanisms of DSR. AODV is based on distance vector principle where each node participating in a route only stores direction and distance. Fig. 2 shows a simplified entry of the routing table.

Dst Id	Next hop	Hop Count

Fig. 2. Simplified AODV Routing Table

AODV has two important phases: *route discovery* and *route maintenance.*

Route discovery phase

When a route does not exist between two nodes (such as when a route is required to a new destination, a link has broken, or a route has expired) the source node *broadcasts* RREQ messages to its neighbors to find the destination node. As the RREQs are forwarded, each node builds a reverse path to the source node. This process of forwarding RREQs continues until the destination node or a node with a valid route to the destination is discovered. When the destination receives the RREQ message, it sends back a RREP message along the reverse path

© 2011 ACADEMY PUBLISHER

containing the number of hops and the latest destination sequence number.

If the destination node receives duplicate copies of RREQ messages, the duplicate copies are discarded. As soon as the source receives the RREP, it can start sending data to the destination using the discovered path, stored in each routing table.

Route maintenance phase

In this phase, HELLO messages are periodically exchanged at a rate of one per second between the nodes in order to establish a list of neighbors at each node and also to detect broken links.

If a link breaks along an active route, the affected node generates an RERR to inform the rest about the unreachable destination. This RERR is propagated towards the source, which initiates a new route discovery process upon receiving the RERR message.

Link failure detection

Two key parameters control the link failure detection: HELLO_INTERVAL and ALLOWED_HELLO_LOSS. The HELLO_INTERVAL is the time interval between hello messages that by default is one second and the ALLOWED_HELLO_LOSS is the number of consecutive hello messages a node fails to receive before the link is considered broken; this value by default is two.

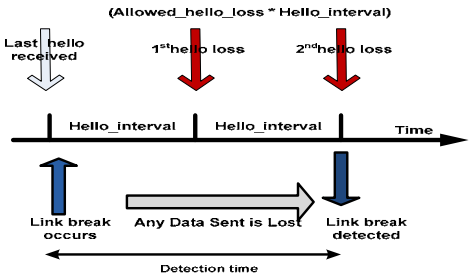


Fig. 3. Link failure detection time diagram

HELLO messages are broadcasted and utilized to detect broken links and maintain connectivity. Nodes are supposed to be receiving hello messages periodically to maintain connectivity.

Fig. 3 shows how the link failure detection is performed. Failure to receive two consecutive hello messages from a neighbor would indicate that the link to that neighbor is down, or connectivity has been lost. Therefore, AODV requires about two seconds to detect that the link is down.

C. AODVM

Ad hoc On demand Distance Vector Multipath (AODVM) [7] is based on the AODV protocol where the source is able to maintain *multiple node-disjoint* routes per destination in which no node can be shared between the multipath routes.

Route discovery phase

As in AODV, the source node broadcasts route request (RREQ) messages to all its neighbors, which are then forwarded by the intermediate nodes towards the destination. Instead of discarding the duplicate RREQ messages, the intermediate nodes record the RREQ information in their RREQ tables. Unlike AODV, the intermediate nodes are not allowed to send a RREP directly back to the source. For every received RREQ packet, the destination updates the sequence number and sends a RREP back to the source.

Route maintenance phase

When an intermediate node receives the RREP, it forwards the RREP through a neighbor in its RREQ table that has the minimum hop count to the source. In order to ensure that a node does not participate in multiple paths, if a node overhears a RREP transmission it removes the entry corresponding to that node from its RREQ tables.

For route maintenance and link break detection, HELLO/RERR messages are utilized in a similar way as in the AODV. However, the AODVM, by default sets the HELLO_INTERVAL to 500 ms and the ALLOWED_HELLO_LOSS to 10 with the intention of preventing protocol instability in their static environment which extended the failure detection time to 5 seconds. Therefore, the AODVM takes about 5 seconds to switch over to the alternative path when the active route fails, leaving a great potential (risk) for packet loss.

Problem Statement

As it was briefly mentioned in the introduction, the AODV/AODVM protocols take a considerable amount of time, between two and five seconds, to detect a link failure and discover a new route to reroute the traffic because the frequency of the hello messages is too small, one per second, to respond fast enough to link breaks to allow real-time communication to continue without interruptions. The reason for sending these hello messages at a low rate is because these hello messages are being broadcasted all the time, and if they were to be sent faster, they will potentially flood the network, causing congestion and affecting the network performance. Since broadcasting hello messages at a high rate to achieve a faster response is not very efficient due to bandwidth and power consumptions, there is a need for the implementation of an adaptive mechanism that allows the detection and recovery from link breaks rapidly in an efficient way to ensure a good VoIP communication on MANETs.

D. Voice over Internet Protocol (VoIP)

VoIP is the transmission of real-time voice traffic over a packet switched data network [14]. Voice transmission [4, 15] is very sensitive to the followings three key parameters:

- Time delay
- Packet loss
- Jitter

These three parameters influence the voice delay and voice clarity, which in turn determines the quality of the voice signal received at the destination. To achieve a decent voice quality, according to [14, 16-17], a loss rate greater than 10% is noticeable, the end-to-end delay should not be greater than 400 ms and the jitter not greater than 75 ms.

The long interruptions due to the poor link failure detection of the existing protocols dramatically affect the delay and packet loss which in turn will degrade the voice quality.

© 2011 ACADEMY PUBLISHER

In order to represent voice in a digital format appropriate for transmission a CODEC must be used. The characteristics of some codecs that are typically used by VoIP clients are shown in Table 1.

TABLE I. VOIP CODEC COMPARISON [15, 18]

Codec	GSM	G.711	G.723.1	G.726	G.729*
Technology	RPE	PCM	ACELP MP-MLQ	ADPCM	CS-ACELP
Bit rate (Kbps)	13.2	64	5.3/6.3	32	8
Frame interval (ms)	20	20	30	20	10
Payload (Bytes)	33	160	20/24	80	10
Packets/sec	50	**50**	**33**	50	**50**
MOS		4.1	3.65	4.0	3.92

*To decrease packet overhead, is more efficient to place three frames per packet

G.711 is the codec that we chose to use for the majority of the experiments since is the one that provides the highest voice quality. However, we also use the G.726 and G.729 for comparison proposes. To transport these voice packets over the network, they are encapsulated in Real time Transport Protocol (RTP) packets and transported over User Datagram Protocol (UDP).

III. AODVM-ALARM

In an effort to improve the performance of real-time traffic, such as VoIP over mobile ad-hoc networks, we propose the implementation of an adaptive localized route maintenance mechanism over multipath ad hoc networks. This is done by monitoring the on-going communications, by utilizing special surge-hello messages and packet analysis techniques, to assist in maintaining the route and providing a faster response time when the topology changes in a dynamic network. In this section, we will discuss the ALARM mechanism and its operation in our ad-hoc test bed.

Our goal was to achieve a failover delay of less than 0.4 seconds to meet the requirements of VoIP traffic [17]. We conducted a large number of experiments in our ad-hoc test bed running the AODVM routing protocol. Based on our experiments and independent research reported in [19] and [20], the results suggested that the HELLO_INTERVAL and ALLOWED_HELLO_LOSS parameters could be adaptive in response to real-time network conditions in order to achieve better performance. Therefore, we considered the implementation of a two state algorithm, a passive and active state. We considered the implementation of the active state by setting the hello rate higher than the default to provide faster detection of topology changes and route failures, as well as faster rerouting of the network traffic. On the other hand, if there is not traffic flowing from source to destination the protocol does not need to maintain any of the routes, then a low hello rate would be appropriate to avoid unnecessary waste of bandwidth and it is considered as the passive state.

On the basis of these observations we decided to use the presence or absence of traffic as the criterion for changing the state of the ALARM mechanism between *passive* and *active*. Hereafter, we shall at times refer to the *active state* as the *surge state*.

Active/Surge State

A node is considered in an active state when it is actively participating in routing traffic from a source to a destination, including the destination. In this state, it is critical for the nodes to maintain connectivity with adjacent nodes along the active route.

The active state is based on flow of traffic. If there is traffic flowing from a source to a destination, but no traffic in the reverse direction, it is defined as one-way traffic.

We use a special control message, a *surge hello*, with the HELLO_INTERVAL set to a lower value (100ms), to achieve a faster reactivity to topology changes. These surge-hellos are only generated *in the case of one-way traffic*. Having said that, the specific implementation of the active state depends on whether the route is supporting *one-way* or *two-way traffic* as will be explained in Sections 3.1.1 and 3.1.2.

Passive State

All the nodes which are not in the active state are considered to be in the passive state. The passive state uses a HELLO_INTERVAL of 1000 ms, which sets the rate for the normal hellos that are broadcasted to keep link connectivity in a dynamic topology (nodes join and leave). This passive state is the default state for all nodes within the ad hoc network.

For both states we set the ALLOWED_HELLO_LOSS to 4 to make the hello messages tolerant of some hellos messages being lost.

A. Operation of the ALARM Mechanism

To illustrate the operation of the ALARM mechanism, we will start with all nodes in the network in a passive state as shown in Fig. 4.

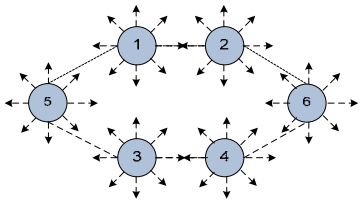


Fig. 4. Passive State.

In this state, all nodes in the network will be broadcasting hello messages at a rate of one per second in order to maintain connectivity with the neighbors. Thus, the network in the passive state keeps waiting for the establishment of a new route while monitoring the start of traffic which will change the state from passive to active. Our mechanism really starts to take action in the active state.

As discussed earlier, routes are maintained using hello messages. However, large hello messages result in slow reactivity to link breaks. To achieve a faster reactivity to link breaks, for two way traffic communication, traffic in the reverse direction is monitored to determine connectivity. Because of the lack of reverse traffic in one-way communication, we introduced surge hellos going in the reverse direction in order to maintain connectivity along the active route.

© 2011 ACADEMY PUBLISHER

We will start by explaining our strategy for one-way traffic and then we will explain the two way traffic approach.

1) One-way traffic scenario

In this scenario, VoIP traffic is generated in only one direction from source to destination as used when streaming audio. If there is traffic from the source node to the destination node, then a *surge state* exists from the destination node to the source node as illustrated in Fig. 5.

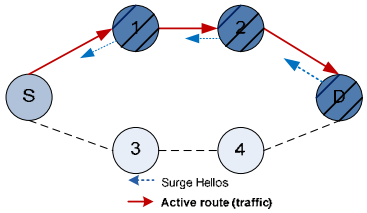


Fig. 5. Active State – One-way traffic.

Once there is a need for traffic to be sent from a source to a destination, the node that becomes the source node executes a route discovery process as previously discussed in Section 2.3. During this process, the destination sends RREPs back to the source through the routes it discovered. Once a route is selected by the source, it is considered active, and the source starts sending traffic through that route. As soon as the transmission of the first packet is detected, the source sends a SURGE_REQUEST to the next node in the route, requesting the initiation of a SURGE_HELLO towards the source as shown in Fig. 6 (a)

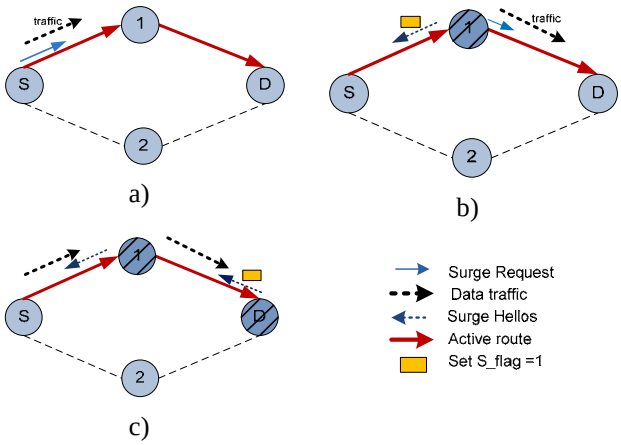


Fig. 6. Example of one-way traffic implementation.

The intermediate node starts sending SURGE_HELLOS back to the previous node along with a flag to indicate the rate that the previous node needs to be able to handle. The intermediate node in turn propagates a SURGE_REQUEST to the next hop in the route to the destination (see Fig. 6(b)). This way, SURGE_HELLOS are sent from the destination through all nodes along the route back to the source to maintain the active route as shown in Figs. 5 and 6 (c). By sending SURGE_HELLOS to the source, the source will be aware if the communication with the intermediate node or destination node is lost and it will be able to react quickly by switching to an alternative route in the event of a link failure.

The hatched nodes in Fig. 6(c) continue sending SURGE_HELLOS at this higher rate towards the source as long as they are part of an active route. Once the source has completed sending traffic, the route is no longer considered active, and the nodes return to the passive state.

Link break response

AODVM-ALARM utilizes SURGE_HELLOS in order to detect link failures along the route. By utilizing SURGE_HELLOS, the protocol is able to refresh the routing table with alternative routes more frequently, thus allowing a faster response to recover from a link failure.

The failover delay can be analytically expressed in terms of the time that it takes for a link failure to be detected.

$$\textit{Failover Delay} = T_{\textit{Link failure detection}} = \textit{ALLOWED_HELLO_LOSS} \times \textit{SURGE_HELLO_INTERVAL} \quad (1)$$

The link failure detection time depends on ALLOWED_HELLO_LOSS and SURGE_HELLO_INTERVAL. Since the nodes that are in an active state are expecting surge hellos, when a node fails to get four consecutive surge hellos (ALLOWED_HELLO_LOSS) with an interval of 100ms each (SURGE_HELLO_INTERVAL), then the link is considered broken. This results in a maximum failover delay of 400ms for the source to switch to an alternative route.

2) Two-way traffic scenario

In this case, there is traffic flowing in both directions, such as in a VoIP session. A similar idea as in one way traffic is applied but in a more efficient way. Instead of generating surge-hellos to monitor and maintain the route, the traffic flowing in the opposite direction is utilized in place of surge-hellos.

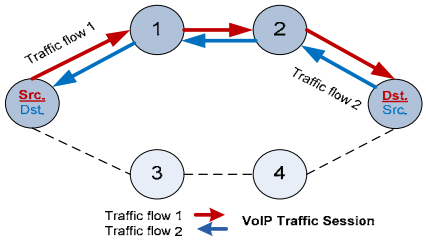


Fig. 7. Active State – Two-way traffic.

Referring to Fig. 7, packets in *traffic flow 2* serve as surge hellos to monitor the active route connectivity for *traffic flow 1*. On the other hand, the packets in *traffic flow 1* serve as surge hellos to monitor the connectivity between the nodes for *traffic flow 2*. As long as we have a bidirectional traffic flow, there is no need to generate additional surge-hellos. However, if the traffic that is being used as surge-hellos pauses or hiccups for some reason, surge-hellos are generated within 100ms after the traffic has stopped. To accomplish this, we incorporate packet analysis techniques to monitor the on-going communication in order to determine when to use surge-hellos in the absence of traffic flowing in the opposite direction. The libpcap library was installed and incorporated in to the program by modifying the makefile

© 2011 ACADEMY PUBLISHER

and the main module to include the header, definition, structures and functions necessary to make the library functional. We also had to add a callback function for packet detection and a timeval structure variable to keep track of the time in which the packets are received by the nodes along the reverse path. In contrast to the timer timeout structure used for scheduling the surge hellos in the one-way traffic scenario, traffic flowing in the opposite direction is tracked by a timestamp to measure the time between consecutive packets, If the timestamp is recent enough (less than 100ms) surge hellos are not sent and the protocol continues monitoring the active route by using traffic; otherwise surge hellos are sent.

Link break response

The link break response for the two-way traffic scenario behaves similarly to the one way, except that in two-way traffic scenarios the mechanism keeps track of the packets going in the opposite direction in place of surge hellos. With that said, the link failure detection time depends on the packets that are being counted as surge hellos. If a node along the path fails to get four consecutive of these packets with an interval of 100 ms each, then the link is considered broken.

IV. EXPERIMENTAL TESTBED FOR AODVM-ALARM IMPLEMENTATION

This section provides the details about the experimental testbed environment, describing the different hardware, software and configuration settings used. The metrics for the performance evaluation of the AODV, AODVM and AODVM-ALARM protocols in the experimental testbed will be presented.

In order to conduct the experimental evaluation and analysis of VoIP traffic over the AODV network, the experimental testbed illustrated in Fig. 6.1 was implemented. It consists of three laptops and one desktop arranged in the topology of two paths, with two hops each, from a source to a destination. Every wireless card in each node was configured to work properly in ad hoc mode by utilizing the ifconfig and iwconfig commands.

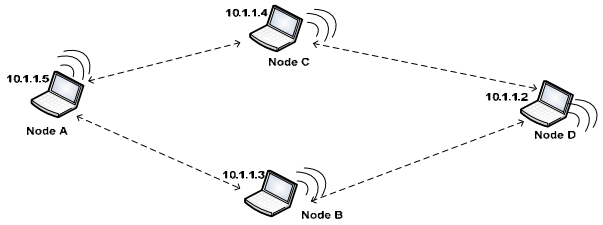


Fig. 8. Experimental testbed setup.

A. Hardware

The specifications of the nodes are displayed in Table 2. For wireless connectivity, the laptops came equipped with internal wireless cards. All wireless cards were configured to work in ad hoc mode and support data rates from 1 to 54 Mbps in accordance with IEEE 802.11g. The cards were also configured to use channel 1 for communications with no encryption algorithms.

TABLE II. HARDWARE SPECIFICATIONS

Nodes	CPU (Processor)	Memory[MB] / Frequency[GHz]	Wireless cards
A	Intel®core(TM)2Duo	4000/2.5	Intel PRO/ wireless 3945 ABG chipset
B	Intel®Pentium®M	1000/1.73	Intel PRO/ wireless 2200 BG chipset
C	Intel®Pentium®M	512/1.6	Intel PRO/ ”
D	Intel®Pentium®D	2000/2.8	SMC2802W PCI / prism GT chipset

B. Software

All nodes were setup with Fedora Core 9 (FC9) using Linux kernel 2.6.22. Once the nodes were configured and functioning in an ad hoc network, we compiled and installed version 0.9.5 of the AODV-UU protocol with the incorporated multipath extension (AODVM) and the ALARM mechanism. Due to the lack of space, we choose to perform our experiments in a more controllable environment where all nodes were actually placed in the same room. To control the connectivity between the nodes, we used the MAC filtering capabilities of the iptables tool. Table 3 shows the iptables rules used for the configuration of our topology.

TABLE III. TOPOLOGY CONFIGURATION USING IPTABLES RULES

Nodes	IP tables Rules	MAC
A 10.1.1.5	*iptables -A INPUT -m mac –mac-source (nodeD) –i wlan2 -j DROP*	00:04:E2:64:17:BA
B 10.1.1.3	*iptables -A INPUT -m mac –mac-source (nodeC) –i wlan0 -j DROP*	00:0E:35:D9:74:AC
C 10.1.1.4	*iptables -A INPUT -m mac –mac-source (nodeB) –i eth1 -j DROP*	00:16:6F:90:B5:92
D 10.1.1.2	*iptables -A INPUT -m mac –mac-source (nodeA) –i wlan0 -j DROP*	00:1F:3C:A6:1D:92

Each of the firewalls in the nodes were configured to drop the packets from the nodes that they should not be able to see, thereby in effect simulating a situation where the two nodes are not within each other’s range.

C. Performance Evaluation Metrics

The following metrics were considered to evaluate the effectiveness of the AODVM-ALARM implementation:

Failover delay: Failover delay is defined as the total delay between the time a link failure occurs and the time in which the sending node switches over to the alternative route.

Failover loss measurements: Failover loss is defined as the number of packets lost during the failover. These are the number of packets that were sent from the source but that never made it to the destination because of packet loss that occurs while the protocol fails over to an alternative route.

Packet loss per trail: Packet loss per trail is the fraction of transmitted packets from the source that were not received at the destination for the entire test run (interval of the trial). It is expressed as a percentage.

Throughput: Throughput is defined as the total number of bits that were transmitted and received successfully at the receiver per unit of time and it is expressed in Kbits/sec.

© 2011 ACADEMY PUBLISHER

V. RESULTS

We used Iperf to generate traffic flows to simulate VoIP communications with the G.711 codec for the experiments. While utilizing Iperf for generating VoIP traffic, we also captured the statistic logs at the destination in order to analyze and parse out the measurements of interest to us. Wireshark [21] was configured to collect and store the trace files at the source and the destination in order to analyze the traffic more closely once the tests were completed. Based on the results obtained with Iperf and Wireshark, we were able to analyze and extract information to produce results to quantify the network performance.

The experiments were performed using the testbed topology shown in Fig. 8. Each trial, consisting of two simulated link failures, ran for 100 seconds. Each experiment was executed five times. The five trials were then averaged to obtain a more accurate result. The results were divided into two scenarios, the one-way traffic and the two-way traffic scenarios.

A. Measurements in one-way traffic scenario

Fig. 9 shows the way that the traffic is established in this scenario. Since our ALARM mechanism behaves differently depending on the scenario and this scenario occurs regularly in real-world situations such as when streaming audio, it is important to verify the one-way scenario to ensure that the mechanism performs well when there are sudden path breaks.

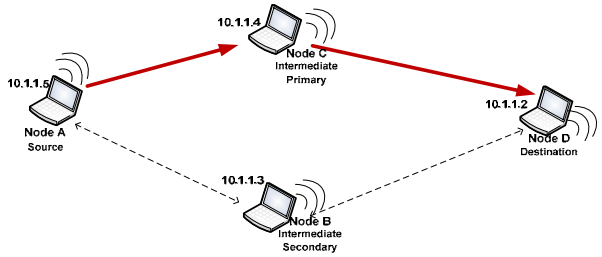


Fig. 9. Experimental Setup for one-way traffic scenario.

In order to produce sudden link failures, we created a script to assist us in our tests to simulate mobility of a single node to the point that the node is out of range of other nodes. The script is executed on both intermediate nodes at the start of a trial. It utilizes the "sleep" command in order to wait for a specified amount of time before it takes the interface down, simulating a 'link break' or 'out of range' effect. After a period of time, the interface is brought back online and this intermediate node is available for routing traffic again.

1) Failover Delay

The analysis of the failover delay in one-way traffic was performed by turning off the wireless interface of the forwarding node used in the active path between the two endpoints, thus forcing a switch over to the alternative route. Note that this action is used to emulate a forwarding node moving out of the coverage area of the sender, or a node being turned off. To measure the protocols' response to link failures, Fig. 10 compares the average failover delay for each protocol.

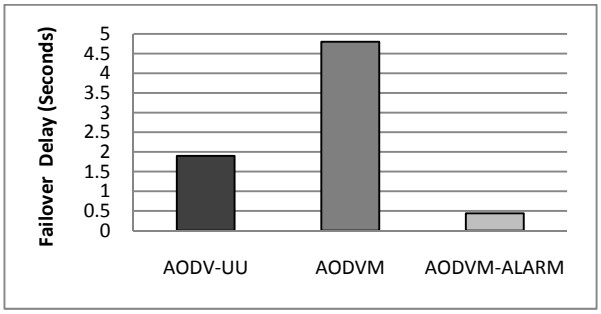


Fig. 10. Comparison of the Average Failover Delay.

The AODVM-ALARM shows the lowest average failover delay of 0.44 seconds, which is a reduction of 80% when compared to AODV-UU and of 90% when compared to the AODVM, resulting in a significant improvement in the response time of the AODVM-ALARM protocol to link failures.

2) Failover loss

Fig. 11 illustrates the average failover loss for the three protocols.

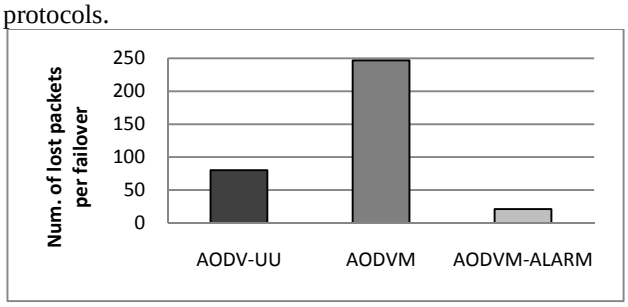


Fig. 11. Average number of packets lost during a failover.

AODV-UU takes about 2 seconds to recover from a failover with an average packet loss of 80 packets per link break and AODVM takes about five seconds which results in an average packet loss of 246. However, the AODVM-ALARM takes about 0.44 seconds with an average packet loss of 21 packets per link break, resulting in a noticeable improvement in performance of AODVM-ALARM when compared to the other protocols.

3) Packet loss per trail

Fig. 12 compares the average percentage of packet loss per trial of the three protocols. AODVM-ALARM shows the lowest packet loss at 0.87% outperforming the AODV-UU and AODVM protocols by 2.3% and 9%, respectively.

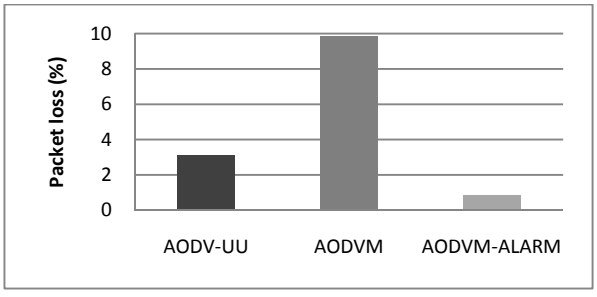


Fig. 12. Comparison of the average percentage of packet loss per trial.

© 2011 ACADEMY PUBLISHER

4) Throughput

Fig. 13 compares the throughput of AODVM-ALARM with AODVM and AODV-UU. Results demonstrate that the AODVM-ALARM performs better than the other two protocols since it responds faster to a link break allowing for a drop in throughput of only 37% when compared to the 90-100% drop of throughput of the other protocols.

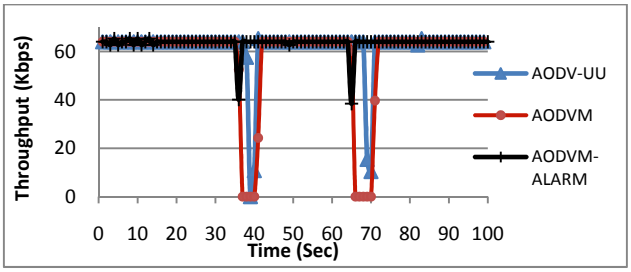


Fig. 13. Throughput measurements comparison.

Experimental results show that AODVM-ALARM considerably improves the overall performance of real-time traffic in MANETs when link failures occur.

B. Measurements in two-way traffic scenario

For this scenario, we simulated a VoIP session by generating bidirectional traffic flows. One going from node A to node D and the other one from node D to node A (see Fig. 14).

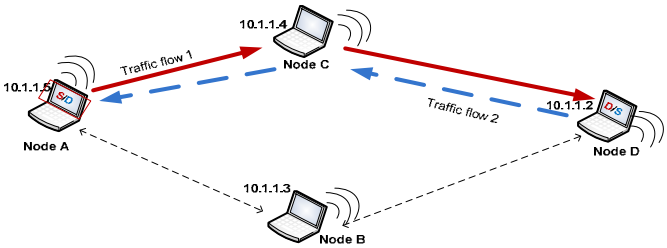


Fig. 14. Experimental Setup for two-way traffic scenario.

To measure the impact of a link failure on a complete VoIP session, we gathered information from our analysis tools at both of the end nodes.

To quantify the performance of the AODVM-ALARM protocol, we will start with the failover delay measurement.

1) Failover delay

The results show that protocols in the two-way traffic scenario respond similar to the one-way traffic scenario and that the AODVM-ALARM performance is better than the other two since it keeps the lowest average failover delay of approximately 0.5 sec.

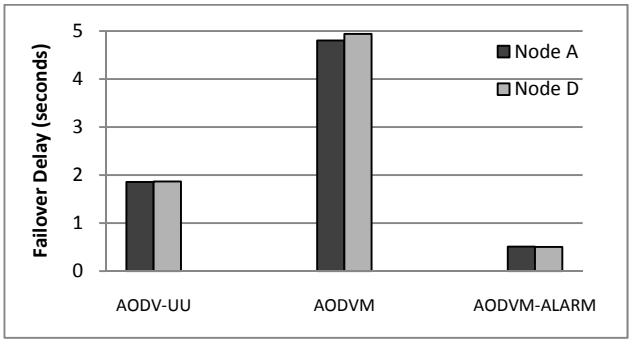


Fig. 15. Comparison of the average failover delay in a two-way traffic scenario.

The AODVM-ALARM protocol in the two-way traffic scenario, simulating a VoIP session, achieves 75% and 95% reduction in the failover delay when compared to AODV-UU and AODVM, respectively.

We verified that the strategy of using the traffic flowing in the opposite direction in place of surge-hellos to maintain the route and obtain a faster response works as anticipated. Furthermore, this approach is more efficient because there is no need to create control messages to control the surge hellos, resulting in less overhead for the protocol and less processing for the nodes.

2) Failover loss

Fig. 16 illustrates the averaged failover loss at each end node for the three protocols.

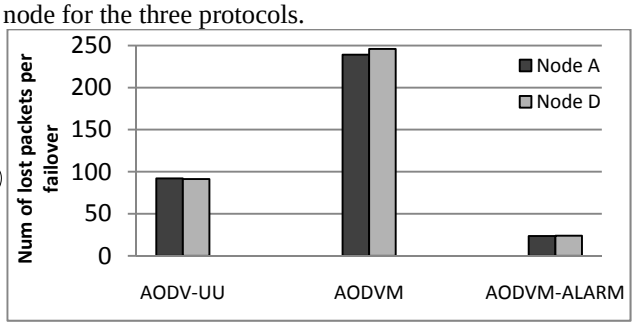


Fig. 16. Average failover loss in a two-way traffic scenario.

For AODV-UU a failover delay of approximately 2 seconds results in about 90 packets lost on average. For AODVM a failover of about 5 seconds results in approximately 245 packets lost and in AODVM-ALARM a failover of about 0.5 causes a packet loss of 24. Therefore, the AODVM-ALARM, in the two-way traffic scenario shows the lowest average packet loss per failover when compared to the other two protocols.

3) Packet loss per trail

Fig. 17 presents a comparison of the average percentage of packet loss per trial for each of the protocols. The AODVM-ALARM shows the lowest average percentage of packet loss at 0.99%, outperforming the other protocols by about 3% and 9% for AODV-UU and AODVM, respectively.

© 2011 ACADEMY PUBLISHER

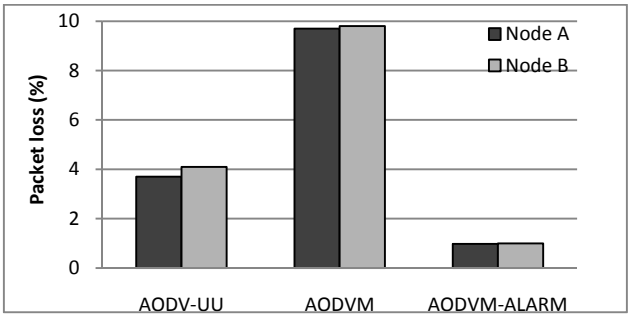


Fig. 17. Average percentage of packet loss per trial in a two-way traffic scenario.

4) Throughput

Fig. 18 illustrates the loss of throughput during the two link breaks for each of the protocols.

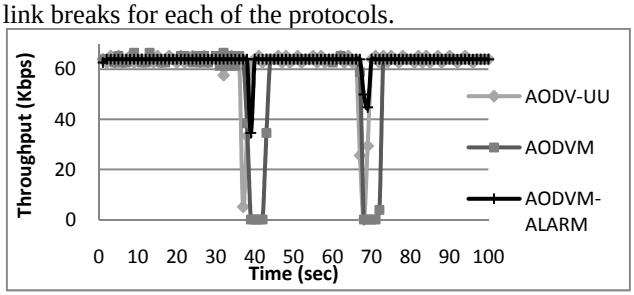


Fig. 18. Throughput measurements comparison

The results show that AODVM-ALARM achieves better performance than the other two protocols since it responds faster to a link break allowing for a drop in throughput of only 37% when compared to the 90-100% drop of throughput of the AODV-UU and AODVM, respectively

C. Impact of the reduction of the surge interval on protocol performance

In this session we evaluate the performance of the AODVM-ALARM protocol when the SURGE_HELLO_INTERVAL is reduced from 100ms to 50ms.

1) Failover delay

The measurements taken at each end node are presented in Fig. 19 which shows that the failover delay at 50 ms is reduced to approximately 300ms, resulting in a failover delay reduction of 40% when compared to the one at 100ms.

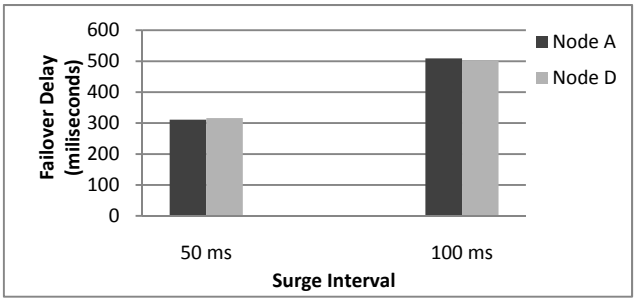


Fig. 19. Impact of the surge interval on the average Failover Delay.

Our results show that at 50ms and 100ms, the failover delay were 50% and 25% higher, respectively, than the estimated values calculated from Equation (1). A possible reason for this to happen could be that the interaction between the operating system and the protocol does not allow for a fast enough response to send surge messages at these short intervals. Therefore, the messages cannot be sent at the expected interval, resulting in a longer delay than expected.

2) Failover loss

Fig. 20 shows that at 50 ms, the failover loss was reduced from 23.7 to 14.2 which corresponds to a reduction of 40%.

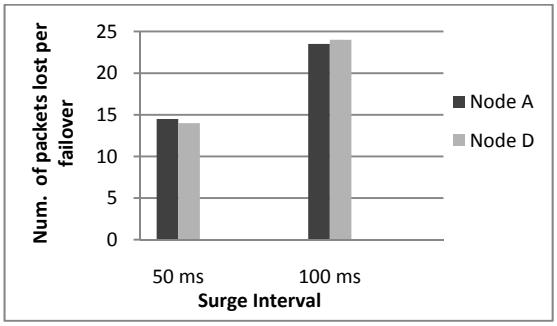


Fig. 20. Impact of the surge interval on the average number of packets lost.

3) Packet loss per trail

Fig. 21 compares the average percentage of packet loss per trial. It shows that by using a 50ms surge interval the packet loss is actually reduced by 44% for the entire test.

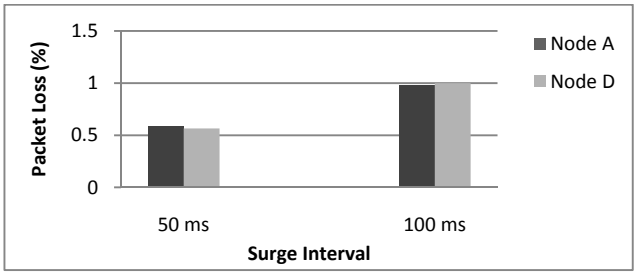


Fig. 21. Impact of the surge interval on the average percentage of packet loss.

4) Throughput

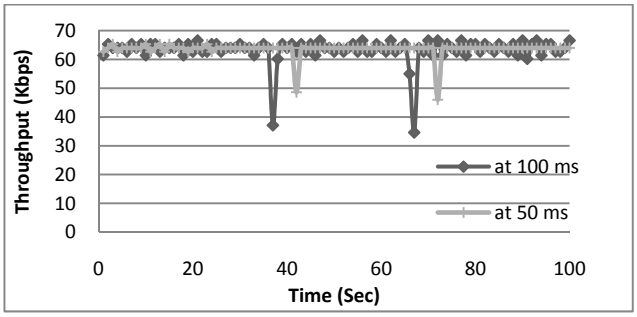


Fig. 22. Throughput measurements comparison

The results show that there is a 21.8% and 37% drop in throughput for the 50ms and 100ms intervals, respectively proving that a 50 ms interval offers a 27% improvement in throughput over the 100ms interval.

We were able to validate further improvement of the response time that we can achieve when utilizing a surge interval of 50ms. Based on the above experimental results,

© 2011 ACADEMY PUBLISHER

we decided to keep this parameter in order to fine tune the overall performance of our protocol.

D. Impact of the codec selection on protocol performance

We further explore in this section, the impact of codec selection on the performance of the AODVM-ALARM protocol. We have selected three voice codecs, G.711, G.729 and G.723 that are generally supported in most VoIP applications. The same four metrics described in section IV (C) are considered in this section with the surge interval set to 50 ms as this resulted in a faster response time as shown in section V (C).

1) Failover delay

Fig. 23 compares the failover delay measured at each of the end nodes for the three tested voice codecs.

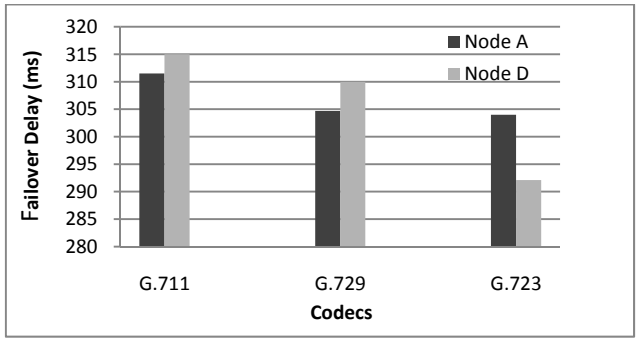


Fig. 23. Impact of the codec on the average Failover Delay.

Results show that, for the three codecs, the failover delay is similar since it is a function of *ALLOWED_HELLO_LOSS* and *SURGE_HELLO_INTERVAL* only. Consequently, the codec selection, which changes the data rate, has negligible effect on the failover delay.

2) Failover loss

Fig. 24 shows the average failover loss at each end node for the three audio codecs. It is noticeable that the G.723 codec exhibits the lowest packet loss of the three since at both of the end nodes only 6 packets were lost while about 9 and 15 were lost when using the G.729 and the G.711 audio codecs, respectively. This is due to the fact that the G.723 codec has the lowest data rate. Therefore, fewer packets are lost per failover.

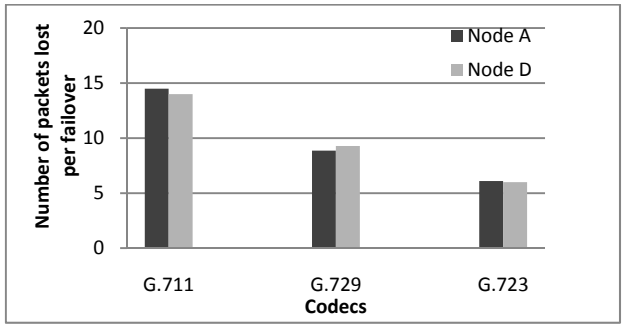


Fig. 24. Impact of the codec on the average number of packets lost.

3) Packet loss per trail

Fig. 25 compares the average percentage of packet loss per trial. The G.723 shows the lowest packet loss of 0.51% at both of the nodes when compared to the rest of codecs.

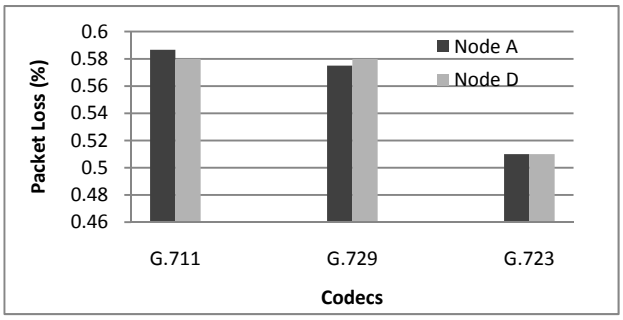


Fig. 25. Impact of the codec on the average percentage of packet loss.

4) Throughput

It is observed in Fig. 26 that since the data rate of the G.723 is the lowest, it has the lowest loss of throughput during a link break. The G.723 consumes less bandwidth than the other two. However, it requires more CPU usage and it does not have the same voice quality as the G.711 codec. Therefore, there is a tradeoff between the voice quality and bandwidth consumption; ultimately it will be a decision best left up to the end-user.

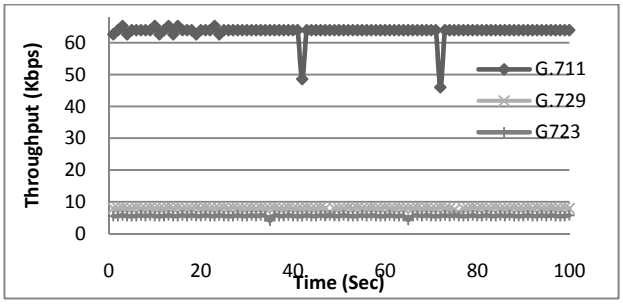


Fig. 26. Impact of the codec on the throughput.

Experimental results showed that AODVM-ALARM outperforms the other two protocols (AODV-UU and AODVM) in terms of failover delay, failover loss and throughput.

VI. CONCLUSIONS

In this paper, we presented the design, development and implementation of the new ALARM mechanism over a multipath routing protocol, referred to as AODVM-ALARM, to improve VoIP communication in dynamic ad hoc networks.

AODVM-ALARM is a very suitable routing protocol for real-time communications, such as VoIP, since it responds quickly to link breakages, reestablishes routes quickly, minimizes packet loss, and due to its framework being based on a multipath approach, is fault tolerant. The protocol's adaptive localized route maintenance mechanism was designed to dynamically monitor the on-going communications in order to recognize, respond and recover quickly from a link failure. It was demonstrated that with the proper tuning of its key parameters, and the ability of our adaptive mechanism to monitor the on-going communication, the failover delay is significantly reduced

© 2011 ACADEMY PUBLISHER

and thus improves the overall performance of the protocol over MANETs.

AODVM-ALARM was experimentally evaluated and compared with its two predecessors in order to validate its effectiveness. In the *one-way scenario*, it reduced the failover delay by 80% and 90% when compared to the AODV-UU and the AODVM protocols respectively, resulting in a reduction in the number of packets lost from 246 and 90 in AODV and AODVM, respectively, to just 21. The overall packet loss was reduced by 2.3% and 9% when compared to AODV and AODVM, respectively, and the throughput only drops to 37% during a link break instead of 90-100% as with the other protocols.

The experimental results in the *two-way scenario* were similar; AODVM-ALARM achieved 75% and 89% reduction in failover delay, and 3% and 9% reduction in packet loss when compared to AODV-UU and AODVM, respectively. Based on these results, we can conclude that the AODVM-ALARM protocol handles both scenarios very well, achieving a significant improvement in the response time to link failures, making it suitable for real-time communications such as VoIP.

In summary, for a VoIP session using G.711 codec and a surge interval of 50ms, the failover delay is 313ms with 14 packets lost during failover and a failover loss per trial of 0.57%.

For a VoIP session using G.723 codec, the failover delay is 298ms with 6 packets lost per failover and a failover loss per trial of 0.51%.

Protocols like AODVM-ALARM which are easily deployable, fault tolerant and have a fast response to topology changes are very suitable not only for disaster recovery but also for emergency response or military operations where the infrastructure might be damaged or destroyed. It can also be useful for setting up communications on the fly for conferencing or sharing documents, home usage or exploration missions utilizing mobile robots in hostile environments.

VII. REFERENCES

[1] S. J. Lee and M. Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks," in *IEEE International Conference on Communications*, pp. 3201-3205, June 11-14, 2001.

[2] C. Thursby. *Ad Hoc Networks*, Available: http://www.acorn.net.au/telecoms/adhocnetworks/adhocnetworks.cfm

[3] C. Perkins, E. B. Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing Protocol", RFC 3561, 2003.

[4] G. Campos and G. Elias, "Performance Issues of Ad Hoc Routing Protocols in a Network Scenario used for Videophone Applications," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Big Island, Hawaii, p. 321, October, 2005.

[5] K. Farkas, D. Budke, B. Plattner, O. Wellnitz, and L. Wolf, "QoS Extensions to Mobile Ad Hoc Routing Supporting Real-Time Applications," in *IEEE International Conference on Computer Systems and Applications*, pp. 54-61, March, 2006.

[6] N.-Y. K. Binod Vaidya, Soon-Suck Jarng, SeungJo Han, "Investigating Voice Communication over Multipath Wireless Mobile Ad hoc Network," in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pp 528-532, January, 2008.

[7] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A routing framework for providing robustness to node failures in mobile ad hoc networks," *Ad Hoc Networks*,vol. 2, no. 1, pp. 87-107, January 2004.

[8] M. K. Marina and S. R. Das, "Ad hoc on-demand multipath distance vector routing," *Wireless Communication and Mobile Computing*, vol. 6, no. 7, pp. 969-988, November 2006.

[9] X. Li and L. Cuthbert, "On-demand node-disjoint multipath routing in wireless ad hoc networks," in *29th Annual IEEE International Conference on Local Computer Networks*, Los Alamitos, CA, pp. 419-420, 2004.

[10] T. Staub, S. Ott, and T. Braun, "Experimental Evaluation of Multi-Path Routing in a Wireless Mesh Network," Technical Report: ECEASST, Electronic Communications of the EASST, 2009.

[11] E. Nordström, *AODV-UU, Ad-hoc On-demand Distance Vector Routing, Uppsala University* 2007. Available: http://core.it.uu.se/core/index.php/AODV-UU

[12] L. M. Feeney, "A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks," Technical Report: October 1999.

[13] C. E. Perkins, *Ad Hoc Networking*. Upper Saddle River, NJ: Addison Wesley, 2001.

[14] H. M. Chong and H. S. Matthews, "Comparative Analysis of Traditional Telephone and Voice-over-Internet Protocol (VoIP) Systems," in *IEEE International Symposium on Electronics and the Environment*, pp. 106 -111, May 2004.

[15] S. Ganguly, *VoIP Wireless, P2P and New Enterprise Voice over IP*. Chippenham, England: John Wiley, 2008.

[16] D. Collins, *Carrier Grade Voice over IP*. 2nd ed. New York: McGraw-Hill, 2003.

[17] J. Tyre and R. Britt, Performance Characteristics of Voice over IP Networks. 2003, available: http://www.techdata.com/content/tdenterprise/whitepapers/03AugNortelNetworksPerformanceCharacteristicsofVoiceoverIPNetworks.pdf

[18] W. Wang and S. C. Liew, "Solutions to Performance Problems in VoIP over 802.11 Wireless LAN," *IEEE Transactions On Vehicular Technology*, vol. 54, no. 1, January 2005.

[19] C. Gomez, M. Catalan, X. Mantecon, J. Paradells, and A. Calveras, "Evaluating Performance of Real Ad-hoc Networks Using AODV with Hello Message Mechanism for Maintaining Local Connectivity," in *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communication*, pp. 1327-1331, September, 2005.

[20] I. D. Chakeres and E. M. Belding-Royer, "The Utility of Hello Messages for Determining Link Connectivity," in *Proceedings of the 5 th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 504 - 508, October, 2002.

[21] *Wireshark*, Available: http://openmaniak.com/wireshark.php

© 2011 ACADEMY PUBLISHER