Published in IET Communications
Received on 30th June 2009
Revised on 30th November 2009
doi: 10.1049/iet-com.2009.0134

In Special Issue on Video Communications over Wireless Networks

ISSN 1751-8628

Reliable multimedia multicast communications over wireless mesh networks

M. Iqbal[1] *X. Wang*[2] *D. Wertheim*[1]

[1]*Faculty of Computing, Information Systems and Mathematics, Kingston University, Kingston upon Thames KT1 2EE, UK*
[2]*School of Engineering, Swansea University, Swansea, UK*
E-mail: writeme@hotmail.co.uk

Abstract: Wireless mesh networks (WMNs) facilitate both data transfer and real-time applications over wireless medium. Owing to the shared nature of wireless frequencies, bandwidth limitation is a major challenge facing WMNs. If real-time multimedia applications, such as live video streaming, are shared among multiple clients using unicast communications, it could result in network resources starvation. Multicast transmission saves network resources by replicating live multimedia transmitted data from one source to multiple destinations using the same stream. The authors have developed a novel implementation of a multicast extension to *ad hoc* on-demand distance vector (MAODV) routing protocol in Linux kernel 2.6 user space, which is referred to as unidirectional link-aware MAODV (UDL-MAODV). Multicast video transmissions use user datagram protocol, which does not use implicit handshaking dialogues for guaranteeing reliability of data. Therefore the authors propose and have implemented modifications to the MAODV route discovery process to improve the reliability of multicast video transmissions. These modifications enable UDL-MAODV to ensure reliable route establishment for multimedia multicast communications over WMNs in the presence of UDLs. The authors describe in this study the software architecture of the UDL-MAODV implementation in the Linux kernel 2.6 user space, and also present multicast validation and results of performance tests using the SwanMesh WMN testbed. Furthermore, UDL-MAODV has been cross-compiled and tests are presented to compare the performance of the implementation using X86 and ARM architecture-based SwanMesh nodes. The test results show that the proposed algorithm is reliable and efficient.

1 Introduction

Multicast communication is the transmission of data from one sender to a group of hosts identified by a single destination address [1, 2]. It utilises network infrastructure efficiently by requiring the source to send a packet only once. This is particularly useful for the transmission of multimedia data over wireless networks where the bandwidth is limited. Previously, we have designed and developed a WMN testbed 'SwanMesh' for disaster and emergency services [3] and healthcare [4–6]. We have also developed a handheld mobile device for the SwanMesh WMN in healthcare [6]. The mobile device is capable of video streaming, Internet access and voice over wireless networks. The purpose of the development is to deliver real-time multimedia services wherever and whenever it is needed. SwanMesh can efficiently deliver data services such as broadband Internet using unicast communications, although there are noticeable throughput drops on each hop between the client and gateway nodes. Owing to the limited bandwidth resources, if unicast is used to deliver real-time multimedia services such as audio, video teleconferencing, and delivery of live video streaming, then this could result in network failure due to resource starvation. Therefore SwanMesh multicast operation is used to deliver the real-time applications.

Multicast routing in wireless mesh networking is a key issue. Similar to unicast, multicast mesh routing has also been adopted from mobile *ad hoc* networks (MANET). Several schemes have been proposed in different multicast routing protocols [7–17]. Comparative analysis studies [18–22] have shown that these multicast protocols perform well under specific scenarios considering mobility, traffic

© The Institution of Engineering and Technology 2010

loads, packet overhead and network conditions. One protocol may not be optimal in all scenarios [23, 24]. These protocols can be classified using two criteria [24, 25]. The first criterion is based on maintaining the routing state. Similar to unicast, it classifies routing mechanisms into two types, proactive and reactive. Proactive maintains a routing state whereas reactive works on demand. The second criterion is based on the multicast packet-forwarding global data structure, which has basically two further types: mesh-based and tree-based. A tree-based multicast routing protocol creates a multicast tree from each of the sources to all receivers, whereas a mesh-based multicast routing protocol sustains a mesh structure containing all the receivers of a group. Hybrid-based multicast routing combines the above two structures. A review of the MANET multicast routing protocols was presented in [25]. Multicast *ad hoc* on-demand distance vector (MAODV) [26, 27] and on-demand multicast routing protocol (ODMRP) [28] are two of the main reactive multicast routing protocols within the MANET working group at the Internet Engineering Task Force for *ad hoc* networks. Multiple tree MAODV (MT-MAODV) [29] is an extension to MAODV that establishes multiple trees to provide multiple routes for multicasting. Destination-driven ODMRP [30] is a destination-driven extension to ODMRP. MAODV is tree-based and ODMRP is mesh-based. Both are well-known protocols used for WMNs. ODMRP works independently whereas MAODV is an extension to the *ad hoc* on-demand distance vector (AODV) [31, 32] unicast routing protocol. Its route-discovery mechanism is based on AODV. MAODV also utilises the control messages that exist in AODV and employs the same route request (RREQ) and route reply (RREP) discovery cycle during its multicast route discovery operation. Thus, route information obtained during multicast route discovery operations increases unicast routing knowledge and vice-versa.

SwanMesh has shown good performance for its unicast communications [3–6]. Since SwanMesh uses reactive AODV user-space-based implementation [33], MAODV is the best option to implement multicast communications in our testbed. Previously, the University of Maryland has developed an MAODV implementation which is more than five years old and does not provide support for Linux kernel version 2.6 [34]. Therefore we have developed a novel Linux kernel user-space-based implementation of MAODV, which we refer to as unidirectional link-aware MAODV (UDL-MAODV). Our multicast implementation is based on [33] as a unicast base protocol. It runs in Linux kernel 2.6 as a dynamically loadable module.

MAODV forms a bidirectional shared tree for its multicast data transmission and does not support multicast over unidirectional links (UDLs). MAODV relies on its underlying unicast AODV protocol to avoid UDLs. During both multicast and unicast AODV operations assume that the links are bidirectional, but in the presence of UDLs AODV will fail [35]. Even the smallest network of two nodes cannot survive in the presence of a UDL between them during multicast operation, for example, if we run a multicast application on two nodes in the presence of a UDL between them, both nodes would become group leaders. This is because none of them would receive an RREP to the RREQ sent by them during the multicast route discovery operation owing to the UDL between them. After becoming the group leader both nodes will start broadcasting the periodic group hello message over UDL. This will initiate an indefinite process of tree merger which will result in network failure.

Therefore sensing and avoiding UDLs is crucial for reliable multicast transmission of multimedia applications over WMNs. Three basic mechanisms described in [36] are used by AODV to avoid UDLs: (i) exchange neighbour sets via an extension field with broadcast HELLO messages; (ii) receive *N*-consecutive broadcasted HELLO messages from another node before completing the neighbour detection process; (iii) using signal-to-noise ratio (SNR) as a threshold for control packets to obtain the signal quality before establishing a route. The major downsides of the above three approaches are that neighbour exchange HELLO messages become variable in size, and the success rate depends on how many neighbours a node detects. Another problem is caused by the different transmission rates between broadcast messages and unicast messages where unicast messages are sent at a higher rate. The lower rate messages can reach further than the higher rate. This approach uses broadcast HELLO messages and hence does not address the difference between unicast and broadcast. Similarly, *N*-HELLO messages also do not address this difference. Furthermore, *N*-HELLO messages introduce latency that may affect on-demand properties of the protocol. The problem with the SNR threshold technique is that it sets a minimum link quality to eliminate poor links; when there are no other routes available, routing data over a poor link (which is below minimum link quality) is better than no route.

We have addressed the above problems by proposing modifications to MAODV. We have proposed and implemented a UDL detection process which works in an on-demand fashion and is invoked only if needed during the multicast route discovery process and also tackles the broadcast unicast transmission difference by using unicast control messages for handshake.

The remainder of this paper is organised as follows. In Section 2, we describe the software architecture of our implementation. In Section 3, our proposed modifications to enhance the reliability of MAODV are presented. In Section 4, we present the results of validation and performance tests using our testbed to ensure the correct functionality of modifications and performance of our implementation. Finally, the paper is concluded in Section 5. A detailed description of MAODV operations can be found in [26, 27].

© The Institution of Engineering and Technology 2010

2 Software architecture of multicast implementation

Our MAODV implementation is based on AODV-UU 0.9.5, a recent kernel user-space implementation developed by Uppsala University, Sweden [33]. During its multicast operation our implementation shares the information maintained by AODV in the unicast route table. Multicast extension maintains a multicast route table to store the route information gathered during multicast operation. Similar to AODV, MAODV also discovers multicast routes whenever needed using RREQ and RREP cycle during its route discovery process. MAODV control packets also share the user datagram protocol (UDP) port used by AODV. Fig. 1 describes the software architecture of our implementation in kernel 2.6 user space.

MAODV uses the following control messages during its multicast route discovery operations:

- RREQ message;
- RREP message;
- Group HELLO (GRPH) message;
- Multicast activation (MACT) message.

The first two messages are the same as those used by AODV during its unicast operation and MAODV utilises them with additional procedures controlled by new flags and extensions. These control messages are of UDP type and the normal Internet protocol (IP) header processing applies to these messages.

Many studies were conducted [37–41], since the release of Linux kernel version 2.6 have shown its improved performance compared to Linux kernel version 2.4 [42]. Netfilter is a framework within the Linux kernel to intercept and manipulate network packets. A Netlink socket is a mechanism used to communicate information between the kernel and user-space processes. Similar to unicast, we also used a Netlink socket to send multicast route entry control messages. These control messages are to add, delete and update multicast route table entries in user space. The detailed software architecture of our MAODV kernel 2.6 user-space implementation is shown in Fig. 1.

MAODV establishes a bidirectional tree for each multicast group, which comprises a group leader, group members and router nodes. Each multicast group has a group leader that is a member of the group and is also responsible for maintaining the group. It broadcasts group hello messages to the network after every GROUP_HELLO_INTERVAL milliseconds so that the other nodes on the network can update their multicast route table information accordingly. Group member nodes are those either sending or receiving the multicast data and the router nodes are not group members but play an important role in forwarding the multicast data

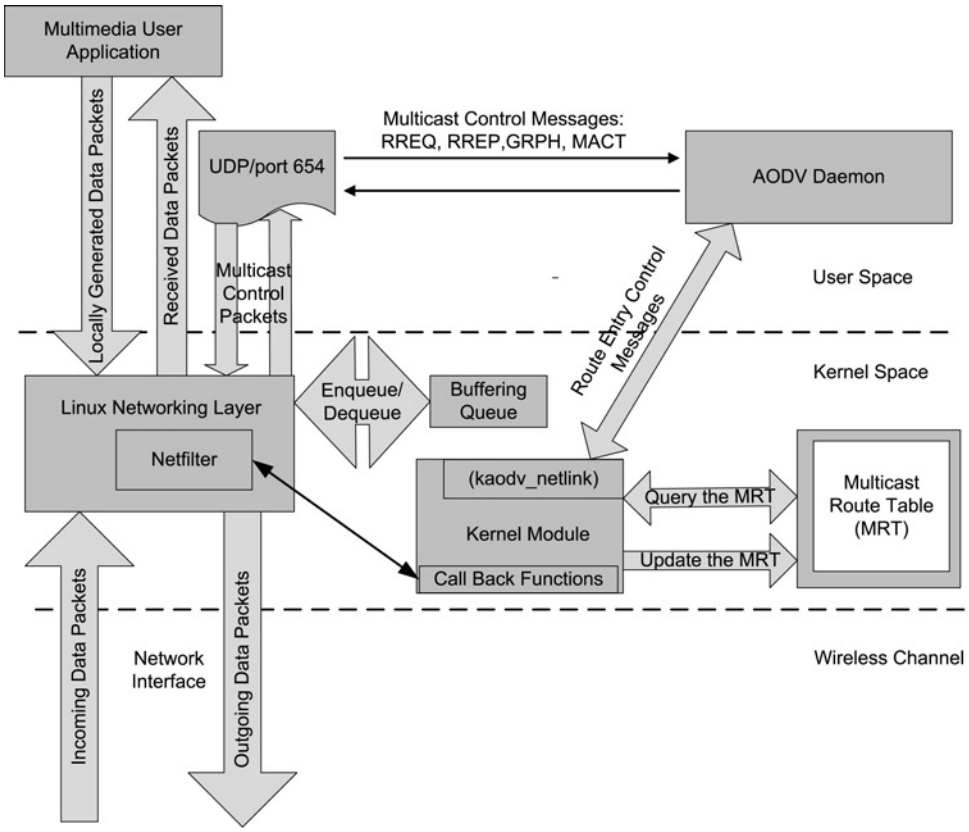

Figure 1 *MAODV kernel 2.6 user-space implementation software architecture*

© The Institution of Engineering and Technology 2010

doi: 10.1049/iet-com.2009.0134

in order to form the multicast group tree for group communications. In the multicast tree structure each node has an UPSTREAM neighbour node or a DOWNSTREAM neighbour node or both. The UPSTREAM neighbour node has a lower hop count to the group leader than the DOWNSTREAM neighbour. Therefore it is closer to the group leader. The DOWNSTREAM neighbour is further from the group leader and thus has a higher hop count to the group leader. The group leader obviously has only DOWNSTREAM nodes.

When a node wishes to join a multicast group in order to send or receive multicast data and does not have a route to the group, it originates and broadcasts an RREQ_ control message. If the join flag is set in the multicast RREQ_ control message then only the nodes which are members of the group will reply to the message by sending back an RREP control message to the originator.

To prevent unnecessary network-wide dissemination of RREQ_ broadcasts, the originating source node of an RREQ_ uses an expanding ring search technique. Using this technique, the source node originates the first RREQ_ with its IP header time to live (TTL) value set to TTL_START. The source node sets the timeout for receiving a corresponding RREP to RING_TRAVERSAL_TIME. The value of RING_TRAVERSAL_TIME equals 2*NODE_TRAVERSAL_TIME*(TTL_VALUE + TIMEOUT_BUFFER), where NODE_TRAVERSAL_TIME is a conservative estimate of the average one hop traversal time for packets and includes queuing delays, interrupt processing times and transfer times. The default value for the NODE_TRAVERSAL_TIME parameter is set to 40 ms [32]. The TIMEOUT_BUFFER is a configurable parameter, which is by default set to 2 [32]. Its purpose is to provide a buffer for the timeout so that if the RREP is delayed because of congestion, a timeout is less likely to occur while the RREP is still routing back to the source. The TIMEOUT_BUFFER can be omitted by setting its parameter value to 0.

If an RREP is not received within RING_TRAVERSAL_TIME, the source node may broadcast another RREQ_ with its TTL incremented by TTL_INCREMENT. The source node keeps originating the RREQ_ if the corresponding RREP is not received up to a maximum of RREQ_RETRIES. Each time a new value for the TTL and RING_TRAVERSAL_TIME is calculated as described above.

If the RREQ_ originating node does not receive an RREP within RREP_WAIT_TIME, then it considers itself the only member of the group and becomes group leader. If the node receives an RREP then it sends a MACT message with a J (Join) flag along the selected best route to activate it. The intermediate nodes along the path to the multicast group will activate their UPSTREAM and DOWNSTREAM neighbour nodes to form the tree. The MACT message is also used with the P (Prune) flag if a node wishes to leave a multicast group. While assembling the multicast control messages during the multicast route discovery process, the participating nodes use different procedure control flags and extensions based on different scenarios.

3 Modifications

We have made modifications to the MAODV algorithm to help to improve the reliability of the protocol in the presence of UDLs. A UDL detection process is proposed in order to avoid UDLs on the network during the multicast route discovery operation.

3.1 UDL detection process

UDLs can be caused by many reasons, including hidden terminal problems [43]. Differences in transmission rate and packet size and fluctuating links are also some of the reasons that can cause UDLs [36]. Additionally, UDLs may also occur due to differences in sophistication or configuration of the device, or due to the signal interference caused by transmissions from other nodes or devices such as jammers.

We propose a UDL detection process which is only activated if there is a multicast application running on the network. The multicast route table has a field called next hops that maintains a linked list structure [27] to store information about the next hop neighbours of the node. We have made modifications by adding a new flag field to the linked list structures; we refer to this as the UDL detection flag. The flag is used to store a value to indicate information on the link between the node and its next hop neighbour. Flag B indicates a bidirectional link whereas flag U indicates a UDL.

During the multicast route discovery process, when a node receives a multicast control message from its next hop neighbour it first looks into its multicast route table to check UDL flag entry before processing the control message. This is to make sure that a bidirectional link exists to the neighbour. If the entry for the next hop neighbour does not exist in multicast route table or it has the entry but the status is inactive due to its being out of the expiry timer's time, then the node will initiate the UDL detection process, which is a two-way handshake process. A node initiates it by sending an acknowledgment message (LINK_DETECTION) to the next hop neighbour node from which it has received the multicast control message. If it receives an acknowledgment message (UDL-ACK) from the next hop neighbour within UDL_WAIT_TIME, then the link is considered a bidirectional link. If the node does not receive UDL-ACK after UDL_WAIT_TIME, it will further make UDL-RETRIES number of attempts by sending LINK_DETECTION messages again. If the UDL_ACK is still

© The Institution of Engineering and Technology 2010

not received, the node will consider the link between itself and the next hop node as unidirectional. At the end of the process an entry for the next hop neighbour into the multicast route table is made indicating the link status as U or B. The UDL entry becomes inactive after LINK_DETECTION_EXP_TIME milliseconds. LINK_DETECTION_EXP_TIME is a variable defined by the user to set the expiry timers on UDL detection entries, which expires on timeouts. This enables the user to increase or decrease the UDL entry expiry time depending on the operating environment and required frequency of the link detection accordingly.

The proposed UDL processes is integrated into MAODV to improve its reliability during route discovery by detecting and avoiding unreliable and UDLs. Thus, these modifications enable UDL-MAODV to ensure a reliable route establishment during multimedia multicast communications over WMNs in the presence of unidirectional links. The process is only invoked during the multicast operation, thus this approach minimises the operating overheads by invoking the process in an on-demand fashion. Furthermore, we have integrated the UDL handshaking process into the MAODV RREQ–RREP cycle in such a way that it does not cause any delay during the RREQ–RREP cycle at the forwarding nodes, only at destination node RREQ–RREP cycle is affected by the UDL handshaking. More details on this are provided in Section 4.1.

Our proposed UDL detection mechanism uses unicast control messages for handshaking, which also takes care of problems introduced by broadcast and unicast transmission differences.

4 Evaluation tests

We used SwanMesh (our WMN testbed) to perform evaluation tests. Mackill (an open source MAC filter utility developed by Uppsala University in Sweden) is a utility tool which can force different connectivity configurations of mesh nodes without the nodes being required to be physically separated. We have used this tool to establish our network topology scenarios during the tests. We have cross-compiled and set up our testbed using several wireless router application platform (WRAP) board [44] and the Liod270 development kit [45] nodes. Fig. 2 shows a WRAP board and the Liod270 development kit. The WRAP boards are based on X86 architecture, whereas the Loid 270 development kit nodes are based on the ARM architecture. We present tests to evaluate the scalability and performance of our UDL-MAODV implementation in terms of delay during the route establishment process in Section 4.1. We have used five WRAP board-based SwanMesh nodes of the same specifications to conduct tests presented in Section 4.1. In Section 4.2, we present our validation tests to ensure the correct functionality of modifications made to MAODV during our implementation. We used four WRAP board-based

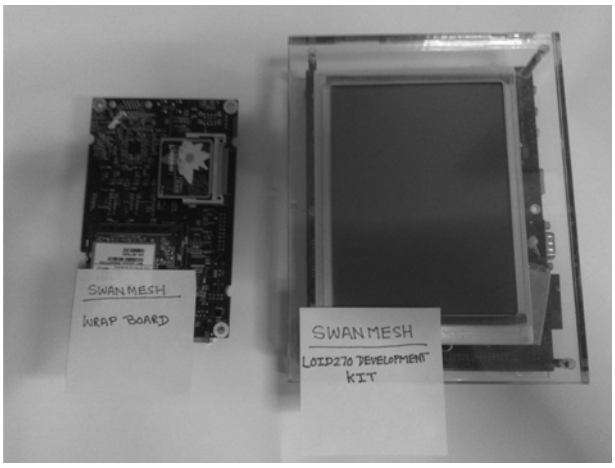

Figure 2 *X86 (WRAP board) and ARM (Loid 270 development kit) architecture-based SwanMesh nodes*

SwanMesh nodes to perform validation tests. To evaluate the cross-platform performance of our implementation, we have performed multicast performance tests using both the ARM and X86 architectures. In Section 4.3, we present a cross-platform performance comparison analysis of these tests.

4.1 Route establishment evaluation

In this section, we present tests to evaluate the scalability and performance of our UDL-MAODV in terms of delay during the route establishment process. MAODV reactively uses an RREQ and RREP cycle to find routes between a source node and a destination node. During the multicast operation when a source node needs to find a route to the destination group, it initiates the route discovery process and broadcasts an RREQ message. A node which is a member of a multicast group or has a route to the multicast tree may respond to the RREQ by sending an RREP back to the source node. All the other nodes process and forward the RREQ by re-broadcasting it.

We have used a chain topology of five SwanMesh nodes test scenario as shown in Fig. 3 to calculate the time that RREQ and RREP take to complete a route discovery cycle. All the five nodes have the same specifications. As shown in Fig. 3, node E is a source node which wishes to find a route to the destination node A. The distance between the source and destination nodes is four hops. Intermediate nodes B, C and D act as router nodes to process and forward the RREQ and RREP control messages to complete the cycle during the multicast route discovery process.

During our test, we first run multicast application on node A, which joins the multicast group as a group leader. The group leader broadcasts group hello messages across the network every GROUP_HELLO_INTERVAL. This enables other nodes to maintain up-to-date information about who the multicast group leader is. When node E receives the group hello message, it learns the hop count to

© The Institution of Engineering and Technology 2010

doi: 10.1049/iet-com.2009.0134

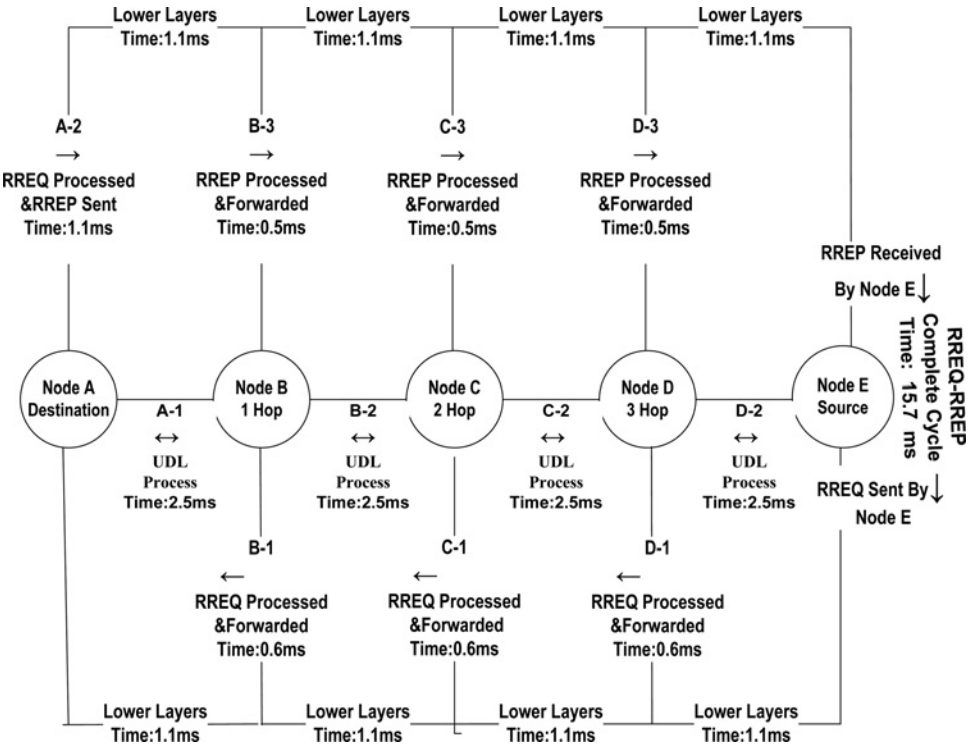

Figure 3 *UDL-MAODV multicast route discovery RREQ–RREP cycle*

the group leader; however, node E is not aware of the route to group leader. Therefore when we run a multicast application on node E, it broadcasts the RREQ message to find the route to the group in order to join it. Since node E is aware of its distance to the group leader, it would not use TTL_START value for the first RREQ. Instead it sets the TTL value to the group leader hop count + TTL_INCREMENT. The value for TTL_ INCREMENT is by default set to 2 [32]. Thus, it broadcasts the first RREQ with a TTL value set to 6. The source node E sets the waiting timeout for receiving a corresponding RREP to RING_TRAVERSAL_TIME, which is calculated as 640 ms during the RREQ–RREP cycle shown in Fig. 3. The formula to calculate the value of RING_TRAVERSAL_TIME [32] is described in Section 2.

If within 640 ms node E does not receive an RREP, it would originate another RREQ. Therefore the total time it takes to establish a route between source and destination nodes depends on how much time the RREQ–RREP cycle takes to complete. Fig. 3 shows details of the period of time taken at several stages to complete the entire RREQ–RREP cycle. All these individual intervals of time at different phases add up to the total time taken to complete the RREQ–RREP cycle during the multicast route discovery process. We have calculated the average processing and travelling time for RREQ and RREP message at each node with the help of time stamps and debugging reports based on several tests. We also describe how much extra delay UDL causes while detecting UDLs. Furthermore, we have cross-referenced the total time taken by the RREQ–RREP cycle using the difference in time stamp values which was logged at source node E at the time of the originating RREQ and receiving RREP.

As shown in Fig. 3, the lower layers time is 1.1 ms which is the delay caused at lower layers during an RREQ or RREP control message transmission between two nodes. It is the average time that the RREQ or RREP message takes to travel after an AODV socket has sent it out from one node and received by the AODV socket at the next hop node. The RREQ processed and forwarded time is 0.6 ms, which is the average time taken at each intermediate node to process the RREQ and re-broadcast it after processing it. The RREQ processed and RREP sent time is 1.1 ms, which is the average time that the destination node takes to process the RREQ and send an RREP. The RREP processed and forwarded time is 0.5 ms, which is the average time taken at each intermediate node to process and forward the RREP back to the source node. The UDL process time is 2.5 ms, which is the maximum time UDL takes to complete the two-way handshake to detect a bidirectional link and update the multicast route table flag accordingly.

We have integrated UDL into the multicast route discovery in such a way that when the first RREQ is received by a node which is not a destination, it processes the RREQ and re-broadcasts it before initiating the UDL process to detect the link between itself and the node from which it has received the RREQ. Thus, it does not cause any delay as the RREQ travels towards the destination. We

© The Institution of Engineering and Technology 2010

numbered the processes in Fig. 3 to show the sequence in which a node performs each process during the RREQ–RREP cycle, for example, node D will process D1 first and once D1 is completed it processes D2. When it receives the RREP, it would initiate D3 to process and forward the RREP. D2 represents the UDL process which is initiated by node D to detect the next hop link on the reverse route of RREQ. The time it takes to complete D2 is less than the time it would take before node D has to perform D3 when it receives an RREP, even if the next hop node is the destination. Therefore when the RREP travels back to the intermediate node to reach the source node, the intermediate node will have information on the link. The node will only forward the RREP if it has a bidirectional link to the next hop node on the reverse route; otherwise it will discard the RREP. Thus, the source node will not receive an RREP for the route with a UDL.

The only delay that is caused by the UDL process is at the destination node. This is because when the destination node receives the RREQ, it would initiate the UDL process to detect the link between itself and the node from which it has received the RREQ before sending an RREP.

We disabled the UDL process to compare the performance of UDL-MAODV with basic MAODV. The average time recorded for MAODV to complete the RREQ–RREP cycle is 13.2 ms using the scenario shown in Fig. 3. UDL-MAODV has taken 2.5 ms extra compared to MAODV to complete the RREQ–RREP cycle, which is due to delay caused at the destination node to complete the UDL process.

MAODV forms a bidirectional shared multicast tree connecting multicast sources and receivers within connected portion of the network. It reactively finds a route between source and destination nodes using an RREQ–RREP cycle. The total time required to complete the RREQ–RREP cycle during the route establishment process for any new node depends on its hop distance to the destination group tree member. Therefore the time it takes to find a route may vary depending on the node's location within the network. Based on experience gained during our UDL-MAODV implementation in the SwanMesh testbed, we have proposed a method to calculate the time UDL-MAODV takes to complete the RREQ–RREP cycle for any node in the network in order to find a route to the multicast destination. In order to provide an accurate calculation, the method requires that all the nodes taking part in the network transmission has the same hardware specifications such as architecture, power and wireless card, and so on. $T_{\text{RREQ\_RREP-cycle}}(S)$ is the total time it would take for the RREQ–RREP cycle to complete at a given source node S, which is calculated as follows

$$T_{\text{RREQ\_RREP-cycle}}(S) = T_{\text{RREQ\_R}}(D) + (H-1)\,(T_{\text{RREQ\_F}}(I) + T_{\text{RREP-F}}(I)) + T_{\text{LL}} + T_{\text{UDL-process}} \quad (1)$$

where $T_{\text{RREQ-R}}(D)$ is the average RREQ processed and RREP sent time taken at the destination node D; $T_{\text{RREQ-F}}(I)$ is the average RREQ processed and forwarded time taken at the intermediate node I; $T_{\text{RREP-F}}(I)$ is the average RREP processed and forwarded time taken at the intermediate node I and H is the hop count distance between source node S and destination node D.

$T_{\text{UDL-process}}$ is the maximum time UDL takes to complete the two-way handshake to detect a bidirectional link and update the multicast route table flag accordingly. T_{LL} is the total delay caused at lower layers during an RREQ–RREP cycle and is calculated as follows

$$T_{\text{LL}} = H(T_{\text{LL-RREQ}} + T_{\text{LL-RREP}}) \quad (2)$$

where $T_{\text{LL-RREQ}}$ is the average delay time caused at lower layers during an RREQ control message transmission between two nodes, $T_{\text{LL-RREP}}$ is the average delay time caused at lower layers during an RREP control message transmission between two nodes and H is the hop count distance between source node S and destination node D.

The $T_{\text{RREQ\_RREP-cycle}}(S)$ calculation assumes that the network is not congested and WMN nodes are fixed. Mobility is not a major issue while providing networking among wireless mesh router nodes as mesh router nodes are either fixed or have a very low mobility in WMNs [46]. The mesh client's devices which connect to mesh router nodes wirelessly may be mobile. In order to control congestion during multicast operation of UDL-MAODV, we have proposed a quality of service (QoS) scheme which is being published in the paper entitled 'A QoS scheme for multimedia multicast communications over WMNs' in this special issue of IET Communications [47].

4.2 Validation tests

We performed a few validation tests to ensure correct functionality of our UDL-MAODV implementation. We have verified and cross-referenced the UDL-MAODV operation using a multicast route table and debugging outputs to ensure its correctness. To cover all the aspects of the network behaviour our validation tests go through the following stages.

4.2.1 First stage: In the first stage of our test, we turn on two nodes A and B. We create a UDL using Mackill. We run a multicast application on node A, which initiates a multicast route discovery process by sending a multicast RREQ as shown in Fig. 4. Node B receives the multicast RREQ control message through the UDL from node A. Node B processes the first multicast RREQ and initiates the UDL detection process. It learns that the link between itself and the next hop node is unidirectional; therefore node B makes an active entry into the multicast route table for the next hop neighbour node A with the UDL flag set to U, indicating the UDL. Node B ignores all the further RREQ control messages sent by node A after checking the UDL entry.

© The Institution of Engineering and Technology 2010

doi: 10.1049/iet-com.2009.0134

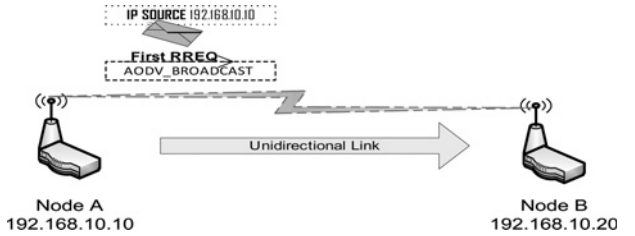

Figure 4 *First multicast RREQ control message received by node B over a unidirectional link*

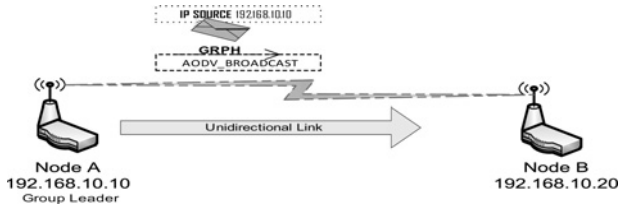

Figure 5 *First GRPH multicast control message received by node B over the unidirectional link*

On not having received a multicast RREP message during the route discovery process, node A becomes group leader of the multicast group itself. Node A starts broadcasting multicast GRPH control messages with GROUP_HELLO_INTERVAL milliseconds. On receiving the GRPH message over the UDL as shown in Fig. 5, node B ignores the GRPH message after going through the UDL checks as the UDL flag indicates a UDL to node A.

4.2.2 Second stage: In the second stage of our validation test, before running a multicast application on node A, we unblock node B's MAC address on node A using Mackill. This changes the link from unidirectional to bidirectional. During the route discovery process when node B learns through the UDL process that the link between its next hop neighbour A and itself is bidirectional, it makes the UDL flag entry to indicate a bidirectional link between the two nodes and processes all the control messages it received after link detection including the periodic GRPH control message.

4.2.3 Third stage: In this stage, we turn on two more nodes C and D to test tree merger over a UDL. We use Mackill to create a topology where nodes C and D can only see each other and cannot hear any communication from nodes A and B. Similarly nodes A and B can only communicate to each other. This creates two groups of nodes AB and CD, where nodes in one group cannot communicate to the nodes in the other group. We start the multicast application on nodes B, C, D and A using the same multicast group address. After going through the route discovery process and UDL checks, we have established the topology where node B becomes a group member for which node A becomes the group leader. Similarly node C becomes the leader of the group and node D becomes a group member.

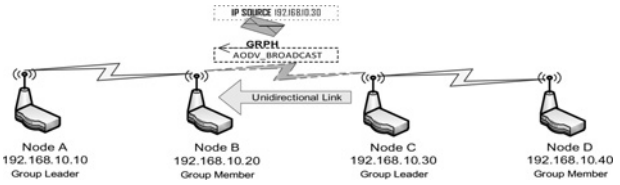

Figure 6 *Node B receives a GRPH over the unidirectional link during tree merger*

Now we use Mackill to unblock node C's address on node B to create a UDL between node B and node C. This enables node B to receive periodic GRPH messages sent by node C as shown in Fig. 6 but node C cannot receive GRPH forwarded by node B. When node B receives the first GRPH message from node C carrying information about another group leader, it processes GRPH and initiates the UDL detection process as it does not have an entry for node C in its multicast route table. Therefore it needs to ensure a bidirectional link to the source node from which it has received the GRPH message before it further process the tree merger. After realising the one way link between itself and node C, it updates its multicast route table with the UDL flag entry indicating a UDL to node C. Node B ignores all the further control messages it receives related to this tree merger. Node B will keep receiving the GRPH control messages from node C after every GROUP_HELLO_INTERVAL over the UDL. It keeps ignoring the GRPH control message after making a UDL check at the multicast route table's UDL flag entry. Thus, it avoids the tree merger process over a UDL.

The above scenario is an example which could cause network failure without our proposed UDL detection process. Without the UDL process the above situation will create an indefinite loop of tree merger. This is because when node B receives a GRPH from node C, it will initiate a tree merger process. Node B will never receive an RREP from node C for the RREQs sent during the tree merger process due to the UDL. Node B will keep initiating the tree merger every GROUP_HELLO_INTERVAL on receiving another GRPH, which will flood the network with the indefinite tree merger process and eventually will fail the network.

4.3 Performance tests

We have conducted the performance tests using both ARM and X86 architecture-based mesh nodes. We have used the Iperf tool to measure the performance of our multicast implementation on both platforms. Iperf is a Linux tool to measure both multicast and unicast network performance. In order to evaluate network performance of our UDL-MAODV implementation with different multicast UDP sender rates, we have performed five sets of Iperf tests in each of the hop scenarios shown in Fig. 7. We ran the Iperf sender application on the multicast sender node to send the multicast data using the following five sender bit rates in each

© The Institution of Engineering and Technology 2010

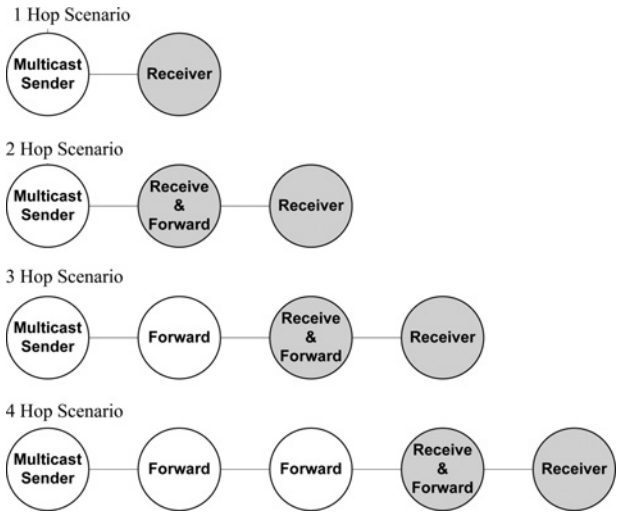

Figure 7 *Multicast performance test scenarios*

hop scenario (1 = 250 Kbps, 2 = 200 Kbps, 3 = 150 Kbps, 4 = 100 Kbps and 5 = 50 Kbps). This results in a total of 20 performance tests each of which is run for 15 min. We performed 20 tests on each platform to conduct a cross-platform multicast performance comparison of our UDL-MAODV implementation. We have used five WRAP boards and five Loid 270 development kits-based SwanMesh nodes during these tests. We formed the multicast chain scenarios shown in Fig. 7 using the Mackill utility.

During the tests we have logged the multicast receiver rate and packet loss for the node labelled as receiver in each hop scenario shown in Fig. 7. We compiled these results to take average values and present the graphical cross-platform comparison of UDL-MAODV performance in Fig. 8.

We observed that on both the platforms the multicast receiver rate drops as the hop distance between sender and receiver increases. Similarly, the packet loss also increases as the sender rate and hop distance between sender and receiver increases. The results show that the packet loss grows as the multicast sender application feeds more packets into the network on longer routes. This is because there are more forwarding nodes on the longer routes. As a result, these nodes may contend for the wireless channel within carrier-sensing range of each other. This leaves each node on the route with a different available link capacity due to flows self-interference. Therefore if the multicast sender application generates packets at a rate greater than the bottleneck bandwidth on the route, from source to destination this results in heavy packet loss. By observing the packet loss results for certain hops with a particular bit rate, we can clearly define a relationship for a hop-count-dependent sender bit rate.

Our UDL-MAODV performs better on the ARM architecture compared to the X86 architecture. There are several reasons for the higher performance. The Loid 270 ARM-based nodes that we have used have higher power compared to the WRAP boards. Additionally, we have used a very much lighter version of the Linux operating system on the Loid 270 nodes compared to the WRAP boards. We have installed a Linux Debian system on the WRAP boards

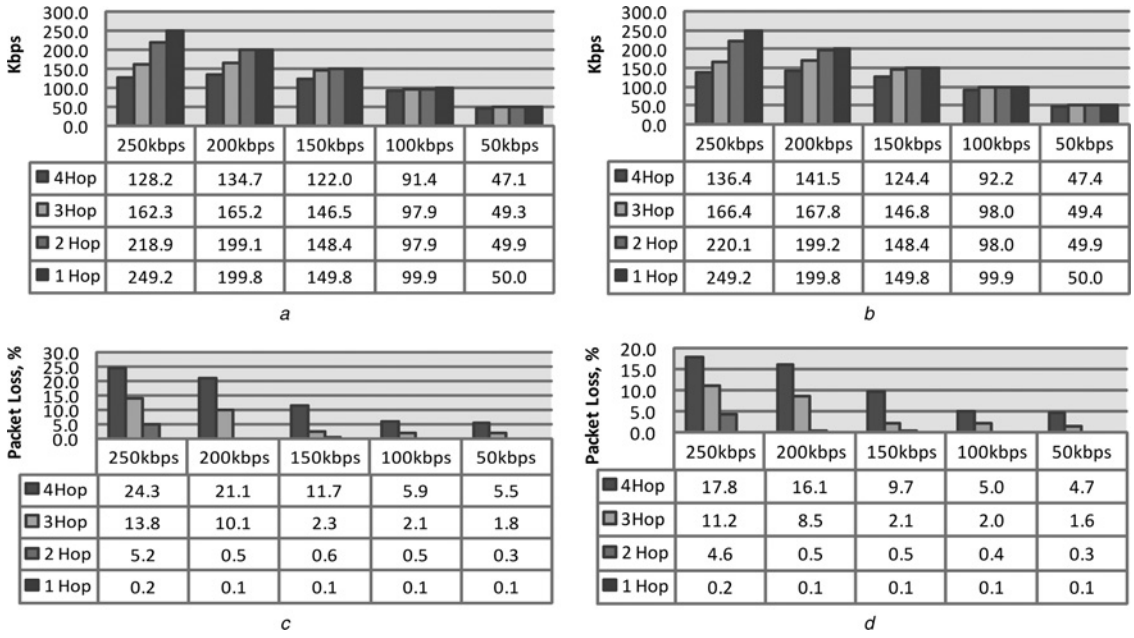

Figure 8 *Cross-platform multicast performance test comparison*

a SwanMesh X86 multicast receiver rate
b SwanMesh ARM multicast receiver rate
c SwanMesh X86 multicast receiver packet loss, %
d SwanMesh ARM multicast receiver packet loss, %

© The Institution of Engineering and Technology 2010

doi: 10.1049/iet-com.2009.0134

which comes with a lot of built-in modules. This slows down the performance of the operating system, especially on embedded and low power nodes. On the Loid 270 nodes we have used a boot loader and a root file system and a very light version of the Linux kernel 2.6. We have selected only required options during kernel compilations from source. Apart from these factors difference in wireless card and antenna sophistication may also play a part in performance.

5 Conclusions and future work

In this paper, we have presented our implementation of MAODV routing protocol based on the Linux kernel 2.6. We are not aware of any other MAODV implementation which has support for the Linux kernel 2.6. The implementation provides multicast multimedia operation in our WMN. We described the software architecture of our Linux kernel 2.6 user-space implementation. We have proposed and implemented some modifications to MAODV to improve and enhance the reliability of the multicast video transmission. We introduced a UDL detection process which is only invoked when there is a multicast application running on the network. We presented tests to evaluate the performance of our UDL-MAODV in terms of delay caused at several stages of a complete RREQ–RREP cycle during the route establishment process. We also describe how much extra delay UDL causes while detecting UDLs during that cycle. We have integrated UDL into multicast route discovery process in such a way that it does not cause any delay at intermediate nodes during the RREQ–RREP cycle. The only delay that is caused by the UDL is at the destination node.

The total time required to complete an RREQ–RREP cycle for any node to find a route depends on its hop distance to the destination. Therefore the time may vary depending on the node's location within the network. Based on experience gained during our UDL-MAODV implementation in the SwanMesh testbed, we have proposed a method to calculate the time UDL-MAODV takes to complete the RREQ–RREP cycle for any node in the network in order to find a route to the multicast destination.

Validation tests were performed to ensure correct functionality of our algorithm. We have also performed cross-platform performance tests and presented a comparison analysis. The test results show that UDL-MAODV performs better on the platform with a lighter version of the operating system and higher power nodes.

The performance tests help us to understand the relationship between network performance and the hop count-based multicast sender rate. The packet loss increases and the receiver rate decreases, as the multicast application feeds more packets into the network on longer routes. This is because there are more forwarding nodes on the longer routes. As a result, these nodes may contend for the wireless channel within carrier-sensing range of each other. This leaves each node on the route with a different available link capacity due to a flow's self-interference. Therefore if the multicast sender application generates packets at a rate greater than the bottleneck available bandwidth on the route from source to destination, this may result in heavy packet loss.

Multimedia video communication over WMNs use the UDP port; therefore these applications may accept some degree of packet loss. This is because UDP multicast communication does not detect network congestion. Lack of congestion control or a congestion avoidance mechanism results in packet loss and poor quality of communications. This problem can be solved using a cross-layer sender-based admission control for UDP real-time traffic; this should be capable of dynamically regulating the new and admitted real-time traffic in order to adapt to the changing network conditions of shared wireless frequency. Thus, the multimedia multicast video sender application would generate packets at a rate no greater than the bottleneck bandwidth on the route from source to destination. This would noticeably improve the video quality over wireless networks.

We have investigated and designed such a QoS scheme which is being published in this special issue of IET Communications [47].

6 References

[1] AGRAWAL D., ZENG Q.A.: 'Introduction to wireless and mobile systems' (Brooks/Cole, 2004, 2nd edn.)

[2] STALLINGS W.: 'Data and computer communications' (Pearson Education, 2003, 7th edn.)

[3] IQBAL M., WANG X.H., WERTHEIM D., ZHOU X.: 'SwanMesh: a multicast enabled dual-radio wireless mesh network for emergency and disaster recovery services', *J. Commun.:* Special Issue on Wireless Communications for Emergency Communications and Rural Wideband Services, 2009, 4, (5), pp. 298–306

[4] IQBAL M., WANG X.H., WERTHEIM D., ZHOU X.: 'Load balanced multiple gateway support in wireless mesh networks for broadband services'. Proc. Eighteenth Wireless and Optical Communications Conf. (WOCC 09), NJIT, Newark, New Jersey, USA, 1–2 May 2009

[5] WANG X.H., IQBAL M., ZHOU X.: 'Design and development of a dual radio wireless mesh network for healthcare'. Proc. Fifth Int. Conf. on Information Technology and Applications in Biomedicine (ITAB 2008), Shenzhen, China, 30–31 May 2008, pp. 300–304

© The Institution of Engineering and Technology 2010

[6] ZHOU X., WANG X.H., IQBAL M., YAN L.: 'A handheld mobile device for wireless mesh networks in healthcare'. Proc. Second IEEE Int. Symp. on IT in Medicine and Education (ITME 2009), Jinan, China, 14–16 August 2009, pp. 1070–1073

[7] CAI S.B., YAO N.M., WANG N.B., YAO W.B., GU G.C.: 'Multipath passive data acknowledgement on-demand multicast protocol', *Comput. Commun.*, 2006, **29**, (11), pp. 2074–2083

[8] JIN J., ZHANG D., WEI G., WAN B.: 'Ripple: an efficient team multicast protocol in wireless ad hoc networks using directional antennas'. Proc. China–Ireland Int. Conf. on Information and Communications Technologies, Beijing, China, 26–28 September 2008, pp. 494–498

[9] YI Y.J., PARK J.S., LEE S.W., LEE Y.Z., GERLA M.: 'Implementation and validation of multicast-enabled landmark ad-hoc routing (M-LANMAR) protocol'. Proc. IEEE Military Communications Conf., Boston, MA, USA, 13–16 October 2003, pp. 1024–1029

[10] PAN D.R., XUE Y., ZHAN L.J.: 'A multicast wireless mesh network (WMN) network routing algorithm with ant colony optimization'. Proc. Sixth Int. Conf. on Wavelet Analysis and Pattern Recognition, Hong Kong, China, 30–31 August 2008, pp. 744–748

[11] PATIL A., ESFAHANIAN A.H., LIU Y.H., XIAO L.: 'Resource allocation using multiple edge-sharing multicast trees', *IEEE Trans. Veh. Technol.*, 2008, **57**, (5), pp. 3178–3186

[12] PATHAN A.S., MONOWAR M., RABBI M., ALAM M., HONG C.: 'NAMP: neighbor aware multicast routing protocol for mobile ad hoc networks', *Int. Arab J. Inf. Technol.*, 2008, **5**, (1), pp. 102–107

[13] DAS S.M., PUCHA H., HU Y.C.: 'Distributed hashing for scalable multicast in wireless ad hoc networks', *IEEE Trans. Parallel Distrib. Syst.*, 2008, **19**, (3), pp. 347–362

[14] KANG N., OH J., KIM Y.: 'A novel approach to overlay multicasting schemes for multi-hop ad-hoc networks', *IEICE Trans. Commun.*, 2008, **91**, (6), pp. 1862–1873

[15] LIU B.H., HUANG P.C., TSAI M.J.: 'Distributed reformation of core-based group-shared multicast trees in mobile ad hoc networks', *J. Parallel Distrib. Comput.*, 2008, **68**, (5), pp. 582–595

[16] GUO S., YANG O.: 'Maximizing multicast communication lifetime in wireless mobile ad hoc networks', *IEEE Trans. Veh. Technol.*, 2008, **57**, (4), pp. 2414–2425

[17] GUO S., YANG O.: 'Localized operations for distributed minimum energy multicast algorithm in mobile ad hoc networks', *IEEE Trans. Parallel Distrib. Syst.*, 2007, **18**, (2), pp. 186–198

[18] PANDEY M., ZAPPALA D.: 'A scenario-based performance evaluation of multicast routing for ad hoc networks'. Proc. Sixth IEEE Int. Symp. on World of Wireless Mobile and Multimedia Networks, Taormina, Italy, 13–16 June 2005, pp. 31–41

[19] ALMOBAIDEEN W., MIMI H.M., MASOUD F.A., QADDOURA E.: 'Performance evaluation of multicast ad hoc on-demand distance vector protocol computer communications', *Comput. Commun.*, 2007, **30**, (9), pp. 1931–1941

[20] NGUYEN U.T.: 'On multicast routing in wireless mesh networks', *Comput. Commun.*, 2008, **31**, (7), pp. 1385–1399

[21] NGUYEN U.T., XU J.: 'Multicast routing in wireless mesh networks: minimum cost trees or shortest path trees?', *IEEE Commun. Mag.*, 2007, **45**, (11), pp. 72–77

[22] SANTOS R.A., GONZALEZ A., VILLASENOR L., GARCIA-RUIZ M., RANGEL V., EDWARS A.: 'Analysis of topological and geographical multicast routing algorithms on wireless ad hoc networks', *J. Electron. Electr. Engng. (ELEKTRONIKA IR ELEKTROTECHNIKA)*, 2008, **2**, (82), pp. 23–28

[23] VISWANATH K., OBRACZKA K., TSUDIK G.: 'Exploring mesh and tree-based multicast routing protocols for MANETs', *IEEE Trans. Mobile Comput.*, 2006, **5**, (1), pp. 28–42

[24] MURTHY C.S.R., MANOJ B.S.: 'Ad hoc wireless networks: architecture and protocols' (Prentice-Hall, 2004, Special edn.)

[25] LUO J.H., XUE L., YE D.X.: 'Research on multicast routing protocols for mobile ad-hoc networks', *Comput. Netw.*, 2008, **52**, (5), pp. 988–997

[26] ROYER E.M., PERKINS C.E.: 'Multicast operation of the ad-hoc on-demand distance vector routing protocol'. Proc. Fifth Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking Seattle, Washington, USA, 15–19 August 1999, pp. 207–218

[27] ROYER E.M., PERKINS C.E.: 'Multicast ad hoc on-demand distance vector (MAODV) routing', Internet Engineering Task Force (IETF) INTERNET-DRAFT <draft-ietf-manet-maodv-00.txt>. Available at http://tools.ietf.org/html/draft-ietf-manet-maodv-00, accessed November 2009

[28] YI Y., LEE S.J., SU W., GERLA M.: 'On-demand multicast routing protocol (ODMRP) for ad hoc networks', Internet Engineering Task Force (IETF) INTERNET-DRAFT <draft-yi-manet-odmrp-00.txt>. Available at http://tools.ietf.org/html/draft-yi-manet-odmrp-00, accessed November 2009

[29] CHOW C.O., ISHII H.: 'Multiple tree multicast ad hoc on-demand distance vector (MT-MAODV) routing protocol for video multicast over mobile ad hoc networks', *IEICE Trans. Commun.*, 2008, **91B**, (2), pp. 428–436

© The Institution of Engineering and Technology 2010
doi: 10.1049/iet-com.2009.0134

[30] TIAN K., ZHAO Z., ZHANG B., LIU H., MA J.: 'Destination-driven on-demand multicast routing protocol', Internet Engineering Task Force (IETF) INTERNET-DRAFT <draft-ke-dodmrp-00>. Available at http://tools.ietf.org/html/draft-ke-dodmrp-00, accessed November 2009

[31] PERKINS C.E., ROYER E.M.: 'Ad-hoc on-demand distance vector routing'. Proc. Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, USA, 25–26 February 1999

[32] PERKINS C.E., BELDING-ROYER E.M., DAS S.R.: 'Ad hoc on-demand distance vector (AODV) routing', Internet Engineering Task Force (IETF) INTERNET-DRAFT <draft-ietf-manet-aodv-13.txt>. Available at http://tools.ietf.org/html/draft-ietf-manet-aodv-13, accessed November 2009

[33] AODV-UU-0.9.5: Available at http://core.it.uu.se/core/index.php/AODV-UU, accessed November 2009

[34] Linux kernel version 2.6: Available at http://www.kernel.org/, accessed November 2009

[35] PRAKASH R.: 'Unidirectional links prove costly in wireless ad hoc networks'. Proc. Third Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Seattle, Washington, USA, August 1999, pp. 15–22

[36] LUNDGREN H., NORDSTRÖ E., TSCHUDIN C.: 'Coping with communication gray zones in IEEE 802.11b based ad hoc networks'. Proc. Fifth ACM Int. Workshop on Wireless Mobile Multimedia, Atlanta, Georgia, USA, September 2002, pp. 49–55

[37] Open Source Development Laboratory: 'Linux process scheduler improvements in version 2.6.0', 2003. Available at http://devresources.linux-foundation.org/craiger/hackbench/, accessed November 2009

[38] LARSON P.: 'Kernel comparison: improved memory management in the 2.6 kernel', 2004. Available at http://www.ibm.com/developerworks/library/l-mem26/, accessed November 2009

[39] WILLIAMSON R.: 'Kernel comparison: networking improvements in the 2.6 kernel', Available at http://www.ibm.com/developerworks/linux/library/l-net26.html, accessed November 2009

[40] KEAT H.K., JING W., HU Z.: 'Linux VM: comparing virtual memory performance between Linux versions 2.4 and 2.6 on low memory system', Florida State University, USA, 2004. Available at http://seantoh.tnlsolutions.my/project/hidden/LinuxVM.doc, accessed November 2009

[41] BHATTACHARYA S.P., APTE V.: 'A measurement study of the Linux TCP/IP stack performance and scalability on SMP systems'. Proc. Int. Conf. on Communication System Software and Middleware, New Delhi, India, 2006, pp. 1–10

[42] Linux kernel version 2.4: Available at http://www.kernel.org/, accessed November 2009

[43] TOBAGI F., KLEINROCK L.: 'Packet switching in radio channels: Part II – the hidden terminal problem in carrier sense multiple access and the busy tone solution', *IEEE Trans. Commun.*, 1975, **23**, (12), pp. 1417–1433

[44] Available at http://www.pcengines.ch/, accessed November 2009

[45] Available at http://www.emertxe.com/content/view/72/126/, accessed November 2009

[46] AKYILDIZ I.F., WANG X., WANG W.: 'Wireless mesh networks: a survey', *Comput. Netw.*, 2005, **47**, (4), pp. 445–487

[47] IQBAL M., WANG X.H., LI S., ELLIS T.: 'A QoS scheme for multimedia multicast communications over wireless mesh networks', *IET Commun.*, 2010, **4**, (11)

© The Institution of Engineering and Technology 2010

Copyright of IET Communications is the property of Institution of Engineering & Technology and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.