An Experimental Study of the Channel Switching Cost in Multi-Radio Wireless Mesh Networks

Stefano Avallone and Giovanni Di Stasi, University of Naples

ABSTRACT

A large body of research has recently addressed the channel assignment problem in multi-radio wireless mesh networks. In order to reduce interference, many proposals require radio interfaces to (more or less frequently) switch channels to exploit the availability of multiple orthogonal channels. However, such proposals have been almost exclusively evaluated by means of simulations and the impact of switching channels on the network performance has not been experimentally evaluated so far. In this article, we aim to fill such a gap and present the results of a thorough experimental campaign we conducted on the ORBIT wireless testbed to study the effects of switching channels in a multi-radio wireless mesh network. Our experiments show that in common scenarios a channel switch causes a non-negligible interruption in the connectivity among nodes, which is on the order of 10 seconds. Also, our tests reveal that a rapid recovery from link failures caused by channel switches is prevented by a rather slow update of the advertised link quality. Inspired by the analysis of the experiments we conducted, we also propose and evaluate a preliminary technique to reduce the interruption in the delivery of packets caused by a channel switch.

INTRODUCTION

Wireless mesh networks (WMNs) are being deployed all around the world to both provide ubiquitous connection to the Internet and carry data generated by several services (video surveillance, smart grids, earthquake early warning systems, etc.). However, since wireless transmissions are involved, interference is a major concern. In order to mitigate the effects of interference, mesh routers are being equipped with multiple radio interfaces, which allow simultaneous transmissions on orthogonal channels (as provided by the IEEE 802.11 standard). Given that the number of available radio interfaces per node is usually less than the number of available channels, the problem of how to assign a channel to every radio interface arises. Channel assignment turns out to be a challenging problem, also due to its interdependence with the routing problem [1]. In order to optimize the network performance, a number of channel assignment algorithms (referred to as *load-aware* [2]) have therefore been proposed that assign channels to radios by taking link loads into account.

However, a channel assignment computed based on a given traffic profile may not be efficient if the network is offered a different traffic load [3]. Hence, a number of papers have recently addressed the problem of re-assigning channels to cope with a variation in the traffic load. Such recomputation of channels is constrained by the need to limit the disruption in connectivity following a number of channel switches.

In [4, 5], heuristics are proposed that minimize the decrease in the amount of traffic delivered to the destination under the assumption that a channel reconfiguration takes a fixed amount of time. The presented numerical simulations assume a value of 1 s for this amount of time. However, no evidence is given that such a value is achievable in practice. The approach proposed in [6] aims to recalculate and disseminate the channel assignment as frequently as once per second. Such a proposal is evaluated by using the ns-3 network simulator. However, the authors assume the availability of a single gateway node that, based on knowledge of the complete topology, traffic demands, and set of interfering links, computes the channel assignment and the routes toward all the network nodes in a centralized manner and disseminates them using a source routing paradigm.Thus, such results do not provide insights about the time required by a network to reach a steady state again after switching channels if a practical distributed hop-by-hop routing protocol is used.

To the best of our knowledge, no other work has presented an experimental study to systematically evaluate the impact of switching channels on the performance of a wireless mesh network. In [7], a *channel abstraction layer* has been implemented in the Linux kernel to experiment with a hybrid channel assignment algorithm. However, the proposed approach brings a number of modifications to the Linux kernel (a unicast table duplicating the routing table is implemented in the channel abstraction layer, a packet buffering mechanism is required to queue packets waiting for the interface to switch to the selected chan-

0163-6804/13/$25.00 © 2013 IEEE

nel, an additional function is implemented to correctly transmit broadcast frames, beaconing at the MAC layer is disabled), which make the presented results only meaningful for the specific architecture considered. In [8], the authors aim at evaluating the impact of a channel switch on the performance experienced by voice and video traffic. However, authors only present some experimental results and do not investigate the factors leading to the obtained results. More important, a multi-radio node is emulated by connecting two single-radio nodes via an Ethernet cable, which clearly affects the accuracy of the presented results.

The goal of this article is to systematically evaluate the impact of switching channels on the performance of a wireless mesh network and to identify the causes of the observed interruptions in the connectivity among nodes. To this end, we conducted a thorough experimental campaign on the ORBIT wireless testbed hosted by Rutgers University.[1] We selected Optimized Link State Routing (OLSR) as the routing protocol for our experiments. OLSR is a proactive routing protocol standardized by the Internet Engineering Task Force (IETF) RFC 3626, for which there is an implementation[2] that is considered to be stable and well tested since it is deployed on many community WMNs with hundreds of nodes. The analysis of the experiments we conducted reveals that a large part of the interruption in the connectivity following a channel switch is caused by the routing protocol, and in particular by the procedure employed to update the link quality. Since every routing protocol implements its own version of such a procedure, the indications we provide in this article are useful for tuning the configuration of any distributed hop-by-hop routing protocol. Based on the analysis of our experiments, we also propose and evaluate a preliminary technique that can be employed to reduce the duration of the interruption in the delivery of packets caused by a channel switch.

In the remainder of this article, after a brief introduction to OLSR, we present and analyze the results of the experiments we conducted and evaluate the technique proposed to reduce the interruption caused by a channel switch. The findings of our experimental study are then summarized.

Optimized Link State Routing

Optimized Link State Routing (OLSR)is a routing protocol for mobile ad hoc networks defined by the IETF Mobile Ad Hoc Network (MANET) working group (RFC 3626). OLSR is a *proactive* protocol, meaning that each node maintains a route for all known destinations at all times. Each node periodically (every 2 s, as proposed by RFC 3626) sends a HELLO message including the list of its neighbors. Such exchange of messages enables each node to discover its neighbors and two-hop neighbors, as well as to estimate how reliable a link to a neighbor is. An HELLO message has an associated validity time, so a node can discard a link to a neighbor if the validity of the last HELLO message received from that neighbor has expired. In order to discard links of poor quality, a mechanism called *link hysteresis* is defined, which makes use of two thresholds: a new link is considered as established when its quality exceeds the higher threshold, while an established link is considered as lost when its quality drops below the lower threshold. The values proposed by RFC 3626 are such that the successful (unsuccessful) reception of three consecutive HELLO messages is enough to mark the link as established (lost). A link is denoted asymmetric if it is only verified in one direction (i.e., a node has received a HELLO message from a neighbor) or symmetric if it is verified in both directions (i.e., a node has received a HELLO message from a neighbor which includes its address). OLSR only provides for the use of symmetric links. To allow every node to build the network topology, a topology control (TC) message is also sent periodically (the proposed period is 5 s). Such a message includes the list of neighbors of the sender and is retransmitted by every node receiving it according to the well-known flooding strategy. Based on the information gathered from the received HELLO and TC messages, each node can use the algorithm proposed by RFC 3626 to find the shortest (in terms of hop count) paths to all the other nodes in the network, thus determining the routes populating the routing table.

At the time of this writing, the IETF MANET working group is developing a second version of OLSR, which differs from the first one mainly because it allows the use of link metrics other than the hop count. Each node computes the quality of an incoming link from a neighbor by collecting statistics on the HELLO messages received from that neighbor and includes such information in the HELLO messages it sends. Thus, a node is aware of the quality of both the incoming and outgoing links to each of its neighbors. The metric associated with an (undirected) link is then computed as a function of the quality of both the incoming and outgoing links. TC messages also carry information on the link quality, so each node builds the network topology and computes the shortest paths to all the other nodes according to the link metric used. The OLSR daemon we used also implements OLSRv2 and different link metrics. In our experiments, we used the default link metric, expected transmission count (ETX) [9].

At the time of this writing, the IETF MANET working group is developing a second version of OLSR, which differs from the first one mainly because it allows the use of link metrics other than the hop count.

Experimental Results

Our testing methodology consists of investigating three different scenarios, each having a precise purpose. First, we consider two nodes and analyze two different cases of channel switching, with the purpose of measuring how long the connectivity between the two nodes is interrupted and identifying the causes of such interruption. Second, we consider a seven-node topology to investigate whether the interruption caused by a channel switch can be alleviated through the use of alternative paths and to give a better idea of what happens in topologies with multiple nodes. Third, we consider a larger topology (13 nodes) and perform a number of (random) channel switches varying from 1 to 5. The goal is to evaluate the impact of the number of channel switches on the network performance and the

[1] *http://www.orbit-lab.org.*

[2] *http://www.olsr.org*

	OLSRv1			OLSRv2		
	UDP		TCP	UDP		TCP
Case I-A	$T_{NEW\_IBSS} + 3 \cdot T_{HELLO}$	13.3 ± 0.6	20.5 ± 7.0	T_{NEW_IBSS}	9.0 ± 1.0	13.6 ± 0.0
Case I-B	$T_{JOIN\_IBSS} + 4 \cdot T_{HELLO}$	7.6 ± 0.8	13.6 ± 0.1	$T_{HELLO\_TIMEOUT}$	20.4 ± 1.0	27.5 ± 0.0
Case II, flow 3→6	$3\ T_{HELLO}$	4.9 ± 0.3	10.2 ± 3.4	$T_{HELLO\_TIMEOUT}$	18.5 ± 0.4	27.5 ± 0.0
Case II, flow 6→3	$T_{JOIN\_IBSS} + 4 \cdot T_{HELLO}$	8.3 ± 0.8	10.2 ± 3.4	$T_{TC\_TO\_SOURCE}$	21.5 ± 0.7	27.5 ± 0.0

Table 1. *Length (in seconds) of the interruption in the delivery of packets measured in all the experiments. T_{NEW_IBSS}: time to establish a new IBSS; T_{JOIN_IBSS}: time to join an existing IBSS; T_{HELLO}: HELLO message interval; $T_{HELLO\_TIMEOUT}$: HELLO message validity; $T_{TC\_TO\_SOURCE}$: time for a TC message to arrive to the mesh source node.*

improvement achieved by the technique we propose to reduce the interruption in the delivery of packets.

All the experiments described hereinafter have been performed with the aid of a centralized channel assignment server (CAS), which connects (via TCP) to each of the channel assignment clients (CACs) running on all the network nodes to communicate the channel switch information (old channel-new channel pairs). When all the CACs have been informed, the CAS sends a Channel Switch message to the CAC running on a selected node. Such a CAC rebroadcasts the received message (multiple times, to account for possible transmission failures) and then performs the required channel switches, and so do all the CACs receiving the Channel Switch message.

Each experiment has been repeated five times to take various sources of randomness into account. For instance, to avoid synchronization of control messages (e.g., HELLO and TC messages), RFC 3626 proposes that each node add a random amount of time (jitter) to the interval at which messages are generated. Hence, results can vary among different repetitions of the same experiment. In the following subsections we describe in detail a single repetition for each experiment, while we refer to Table 1 for a summary of the results achieved in all the repetitions.

All the mesh nodes have two IEEE 802.11 radio interfaces (Atheros chipset) operating in ad hoc mode and using the standard distributed coordinated function (DCF) as the channel access mechanism. To alleviate the adjacent channel interference, we used channels with center frequencies separated by at least 40 MHz. In the following, we just enumerate channels as 1, 2, 3, … instead of specifying the exact IEEE numbering. Every mesh node runs the Linux kernel (v. 3.2) and the OLSR daemon (v. 0.6.4) configured with the default values. The driver used for the radio interfaces is ath5k. In all the experiments, the WMN is traversed by low-bitrate traffic (100 kb/s) in order to decrease the probability of collisions. Traffic has been generated by D-ITG [10].

Scenario I: Two-Node Topology

We consider two mesh nodes running OLSR, and two host nodes acting as source and destination of a single traffic flow. The sender is connected to mesh node 0, and the receiver is connected to mesh node 1 (both via Ethernet cable). In the following, we present the experimental results achieved in the two cases we considered.

Case I-A: Fixed Neighbor-to-Interface Binding — Figure 1a shows the setup used for this experiment (the hosts acting as sender and receiver are not shown). For clarity, only the last two bytes of the IP addresses of the mesh nodes are shown (i.e., 0.1 instead of 10.0.0.1). Nodes 0 and 1 initially establish a link on channel 1, which is crossed by the traffic generated by the sender. At some point in time, both nodes switch their radios on channel 2. Thus, this experiment serves the purpose of analyzing the case of a fixed neighbor-to-interface binding; that is, each node continues to use the same radio interface to communicate with the neighbor after the channel switch. In such a case, no change in the route used to reach the neighbor is required on both nodes. We first comment on the results achieved by OLSRv1 (with the hop count metric) in the case of UDP traffic. Figure 1b shows that such a channel switch causes an interruption in the delivery of UDP packets to the destination that lasts for about 12.7 s. We looked at the traffic traces of the experiment to identify the causes of such a long interruption (the exchange of some relevant packets is illustrated in Fig. 1d). It turns out that no packets are successfully exchanged between the two nodes for about 7.8 s following the channel switch, which indicates a lack of connectivity at the medium access control (MAC) layer between the two nodes. The reason is that the IEEE 802.11 standard provides that nodes cannot trasmit data before they join a basic service set (BSS) in infrastructure mode or an independent BSS (IBSS) in ad hoc mode. Thus, after a channel switch, each node needs to join a new IBSS on the new channel. The implementation of such a procedure within the Linux kernel requires that a node listens on the specified channel (or scans all the channels if no channel is specified) at intervals of 2 s until either a beacon frame advertising an existing IBSS (or a specific IBSS in case a BSSID is specified) is received or a timer, whose duration is 7 s, expires. If the timer expires, the node creates its own IBSS and starts advertising it in the beacon frames it sends. Otherwise, the node joins one of the IBSSs that have been advertised.

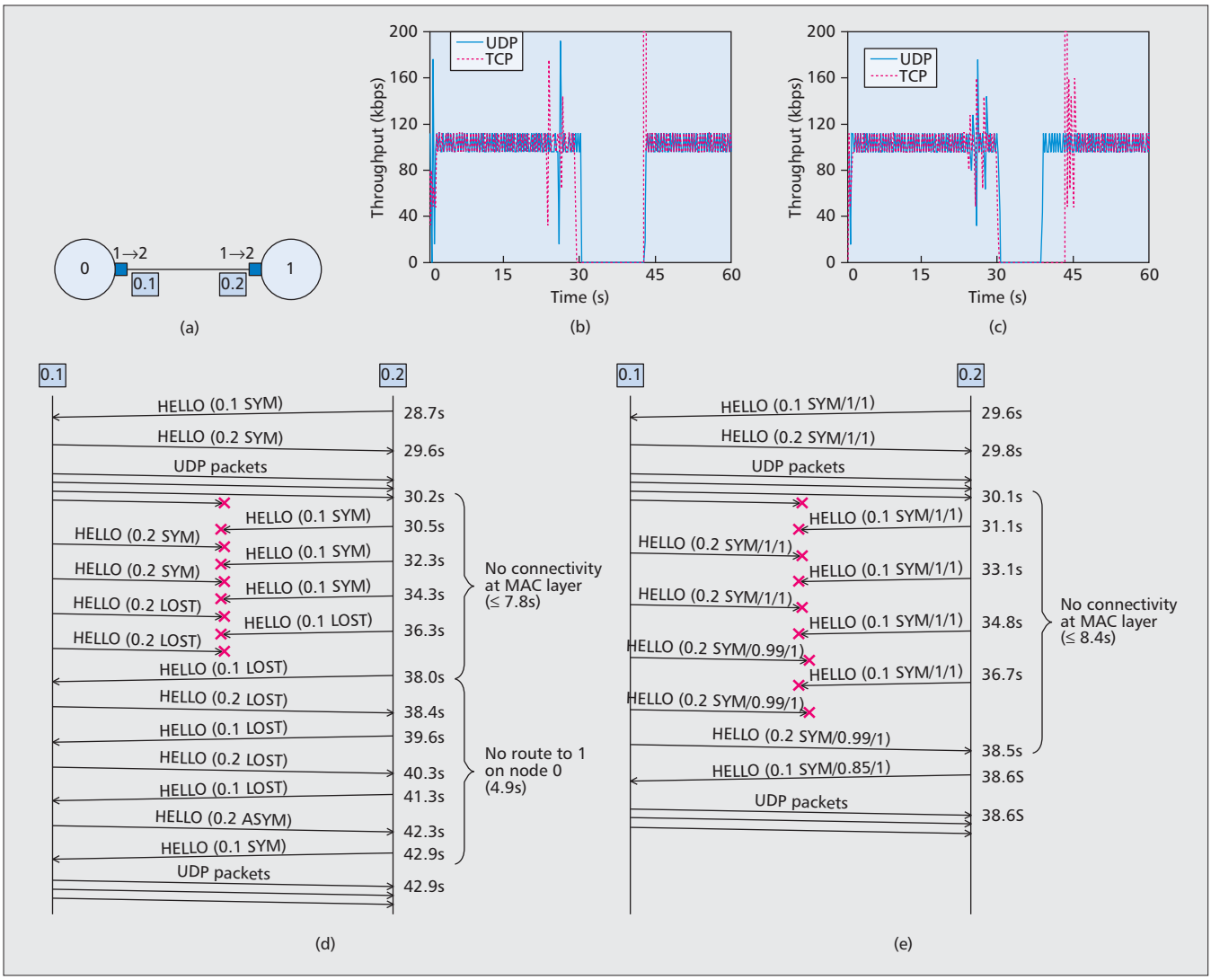

Figure 1. *Case I-A: fixed neighbor-to-interface binding: a) network topology; b) OLSRv1 (hop count); c) OLSRv2 (ETX); d) OLSRv1 (hop count); e) OLSRv2 (ETX).*

In our experiment, both nodes wait 7 s in the attempt to receive beacon frames announcing an IBSS on the new channel. Then one node creates and advertises a new IBSS, which is joined by the other node, and the connectivity at the MAC layer is restored. At this point, however, each node no longer has a valid route to the other node. Indeed, the lack of connectivity at the MAC layer lasts long enough to cause the loss of three consecutive HELLO messages, which, according to the hysteresis mechanism used by OLSRv1, induces the routing protocol to consider the link to the neighbor as lost. Thus, once the connectivity at the MAC layer is restored, the link to the neighbor must be established again. To this end, each node has to receive three consecutive HELLO messages before the link can be considered symmetric and a route to the neighbor can be inserted into the routing table. Thus, the interruption in the delivery of packets between two nodes lasts for the time required to establish a new IBSS plus the time required to establish a new link (three times the HELLO interval, in case of OLSRv1 using default values). Rather than making the process of marking a link as lost slower (which would affect the ability of the routing protocol to rapidly react to link failures), a viable solution for avoiding the additional time needed to reestablish a link could be to set the timer used by the MAC layer to determine that no IBSS is available to join at a value less than the time required by the hysteresis mechanism to mark a link as lost (since no value for such a timer is mandated by the IEEE 802.11 standard).

When OLSRv2 (with the ETX metric) is used, the delivery of UDP packets is interrupted for about 8.5 s after the channel switch (Fig. 1c). As explained above, the creation of a new IBSS takes at least 7 s, during which a number of HELLO messages get lost (Fig. 1e). However, contrary to what happens with OLSRv1, the link quality used by OLSRv2 degrades rather slowly, and the lack of connectivity at the MAC layer does not last long enough to make the link quality drop below the threshold associated with a usable link (0.1 by default). Hence, in this case, the interruption in the delivery of packets basically coincides with the interval required to create a new IBSS at the MAC layer, and there are

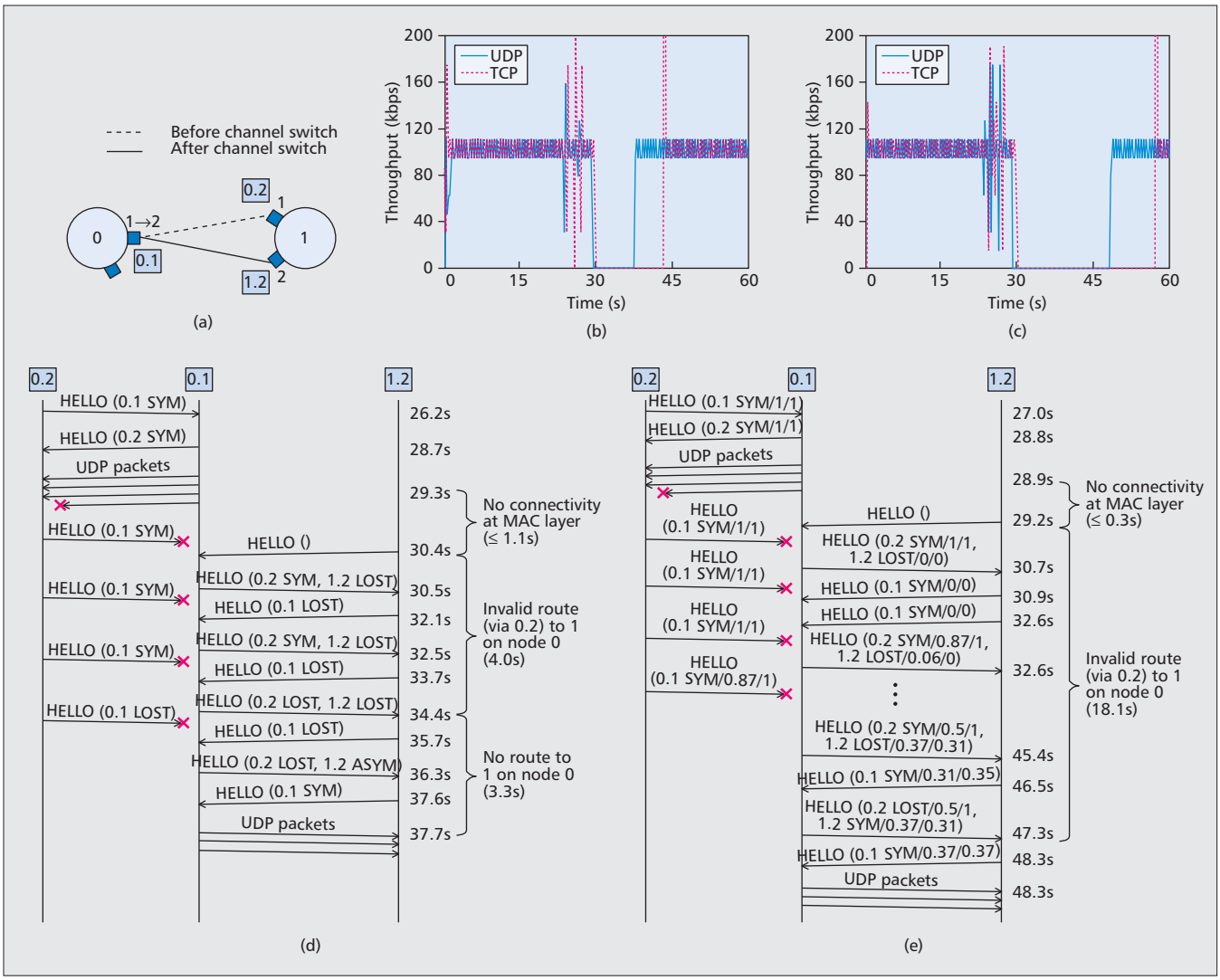

Figure 2. *Case I-B: changing neighbor-to-interface binding: a) network topology; b) OLSRv1 (hop count); c) OLSRv2 (ETX); d) OLSRv1 (hop count); e) OLSRv2 (ETX).*

no additional delays introduced by the routing protocol.

In case of TCP traffic, the interruption in the delivery of packets to the destination may be prolonged by the TCP congestion control mechanism. Indeed, if packets are not acknowledged before the re-transmission timer expires, the TCP sender stops transmitting new packets, re-transmits the packets that have not been acknowledged and doubles the re-transmission timer. In the case of OLSRv1 (Fig. 1b), valid routes are restored just before the first unacknowledged packet is retransmitted for the sixth time (13.63 s after its initial transmission). However, for some repetitions (2 out 5) of this experiment, valid routes are restored just after the sixth retransmission of the first unacknowledged packet, thus making it fail again. In such a case, the packet is retransmitted again after 13.84 s, thus making the interruption in the delivery of TCP packets last for about 27.4 s. This behavior explains the high deviation in the values reported in Table 1. In the case of OLSRv2 (Fig. 1c), the time at which the new IBSS is established (about 7 s since the channel switch) falls in between the first unacknowledged packet is sent for the fifth time (6.7 s since the initial transmission) and the sixth time (13.63 s since the initial transmission). Hence, the interruption in the delivery of TCP packets lasts 13.6 s in all the repetitions of this experiment.

Case I-B: Changing Neighbor-to-Interface Binding — The setup considered in this case is shown in Fig. 2a. Nodes 0 and 1 initially establish a link on channel 1, which is crossed by the traffic generated by the sender. At some point in time, node 0 switches its radio operating on channel 1 to channel 2. Consequently, the two interfaces involved in the previously established link no longer hear each other. Instead, the interface on node 0 can now hear a different interface on node 1, and a new link must be established between them. Thus, in this case, each node has to change its route to the other: node 0 has to change the next hop (1.2 instead of 0.2), while node 1 has to change the interface used to reach the neighbor. We first comment on the results[3] achieved by OLSRv1 (with the hop count metric) in the case of UDP traffic.

[3] *The analysis of these experiments led us to identify and fix a bug in the implementation of OLSR regarding the selection of the link to be used to connect to a neighbor in case of multiple available links. All the results presented in this paper have been obtained by running the fixed version of OLSR.*

Figure 2b shows that the channel switch described above causes an interruption in the delivery of UDP packets to the destination that lasts for about 8.4 s. By looking at the traffic traces (Fig. 2d), we notice that the interruption in the connectivity at the MAC layer between the two nodes is much shorter (less than 1.1 s) than in the previous case I-A. This is because when the interface on node 0 switches to channel 2, it soon receives a beacon frame from the interface on node 1, which is already operating on channel 2, and thus promptly joins the advertised IBSS. The remaining part of the interruption is due to the routing protocol. Indeed, the OLSR daemon on node 0 still considers the link to 10.0.0.2 as symmetric and hence uses this IP address as the next hop for the destination. The IP address is resolved to the MAC address of the interface on node 1 operating on channel 1, which is clearly not reachable by node 0 (whose interface is now operating on channel 2). Node 0 no longer receives HELLO messages from 10.0.0.2, and hence, according to the hysteresis mechanism, it marks the link to such interface as lost after a few seconds and removes the corresponding route from the routing table. Meanwhile, node 0 receives HELLO messages from 10.0.1.2; therefore, a few seconds later, a new link between 10.0.0.1 and 10.0.1.2 is established and used to exchange traffic between nodes 0 and 1.

We now comment on the results achieved by OLSRv2 (with the ETX metric). Figure 2c shows that the delivery of UDP packets is interrupted for about 19.4 s after the channel switch. Again, the lack of connectivity at the MAC layer lasts for a short time (less than 0.3 s, Fig. 2e), and most of the interruption in the delivery of packets is due to the routing protocol. Once the connectivity at the MAC layer is restored, node 0 exchanges HELLO messages with 10.0.1.2, while no longer receiving HELLO messages from 10.0.0.2. Hence, the quality of the link between 10.0.0.1 and 10.0.0.2 decreases, while the quality of the link between 10.0.0.1 and 10.0.1.2 increases. However, as noted while describing case I-A, the quality of a link improves and degrades rather slowly according to the mechanism used by OLSRv2. For this reason, the quality of the failed link between 10.0.0.1 and 10.0.0.2 stays higher than that of the new link between 10.0.0.1 and 10.0.1.2 for a long time; hence, the new link is not used. Eventually, the old link is discarded in favor of the new one not because its quality is exceeded by that of the new link, but because the validity (20 s) of the last HELLO message received from 10.0.0.2 expires, and the old link is marked as lost. In fact, at the time the old link is marked as lost, its quality still exceeds that of the new link. As soon as the old link is marked as lost by node 0, the only remaining link to node 1 is the link between 10.0.0.1 and 10.0.1.2. A corresponding route is added to the routing table of node 0, and the traffic starts flowing again between nodes 0 and 1. We observe here that the slow degradation of the quality of a failed link is accentuated by the fact that the quality of the link in the outgoing direction remains fixed to 1. Indeed, this value can only be measured by the neighbor and communicated to the node via HELLO messages. Since no HELLO messages are received from the neighbor after the channel switch, the quality of the link in the outgoing direction remains fixed to the last value communicated by the neighbor.

Contrary to the case of fixed neighbor-to-interface binding (case I-A), in this case the lack of connectivity at the MAC layer is negligible, and most of the interruption in the delivery of packets consists of the time the routing protocol takes to prefer the link on the new channel over the link on the previous channel. Thus, routing protocols that quickly update the link quality (as done by OLSRv1, in comparison to OLSRv2) enable to (relatively) rapidly switch to the new link, thus reducing the interruption in the delivery of packets to the destination. However, we note that the interruption in the delivery of packets cannot be made arbitrarily small by tweaking the protocol parameters. Indeed, reducing the HELLO interval has the drawback of increasing the overhead, while reducing the number of consecutive HELLO messages that need to be received (lost) to consider a link as established (lost) has the drawback of making the establishment/removal of links too sensitive to temporary degradations of the wireless channel (which can frequently occur in practice), with the risk of introducing instabilities at the routing layer.

In the case of TCP traffic, we again observe an increase in the duration of the interruption with respect to UDP traffic. For OLSRv1 (Fig. 2b), the time at which the new route is added to the routing table usually falls in between the first unacknowledged packet is sent for the fifth time (6.7 s since the initial transmission) and the sixth time (13.63 s since the initial transmission). Hence, the duration of the interruption in the delivery of TCP packets is about 13.6 s in all the repetitions of this experiment. For OLSRv2 (Fig. 2c), the time at which the new route is added to the routing table usually falls between the first unacknowledged packet being sent for the sixth time (13.63 s since initial transmission) and the seventh time (27.47 s since initial transmission). Hence, the duration of the interruption in the delivery of TCP packets is about 27.5 s in all the repetitions of this experiment.

Finally, we remark that the only aspect of OLSR determining the results shown in this section is the procedure adopted to consider links as established/lost. Indeed, we have shown that the duration of the interruption in the delivery of packets depends on how fast such a procedure is. Therefore, we believe that the comments made in this section are not specific to OLSR, but they can be applied to any distributed hop-by-hop routing protocol.

Scenario II: Seven-Node Topology — We consider seven mesh nodes (Fig. 3a) running OLSR, and two host nodes acting as the source and destination of a pair of traffic flows. One flow is generated by the host connected to node 3 and destined to the host connected to node 6, while the other flow follows the opposite direction. At some point in time, a channel switch of the same kind as case I-B takes place. In particular, node 3 switches its radio operating on channel 1 to channel 2 so that such a radio can

Contrary to the case of fixed neighbor-to-interface binding (case I-A), in this case the lack of connectivity at the MAC layer is negligible, and most of the interruption in the delivery of packets consists of the time the routing protocol takes to prefer the link on the new channel over the link on the previous channel.

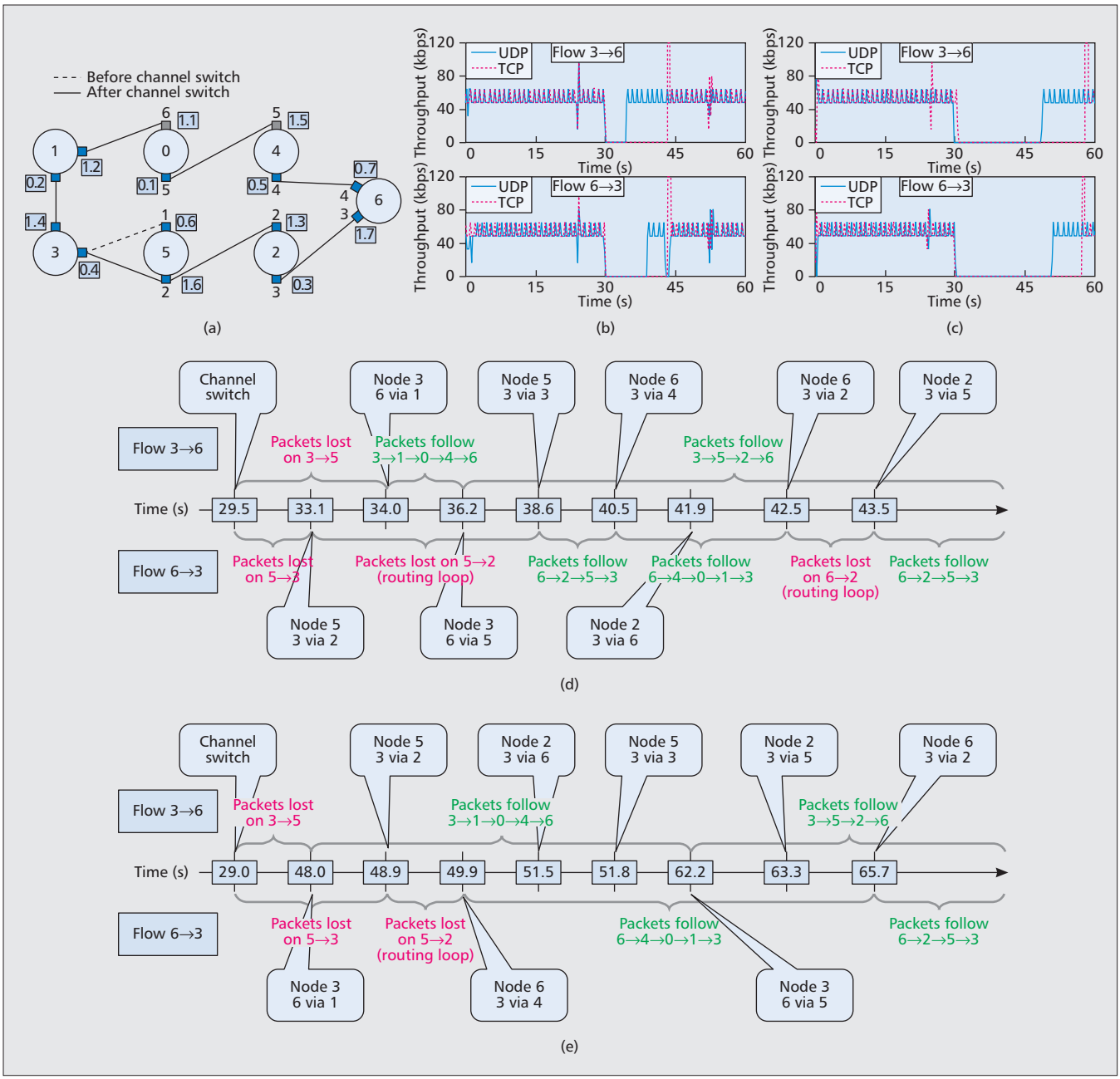

Figure 3. *Scenario II: seven-node topology: a) network topology; b) OLSRv1 (hop count); c) OLSRv2 (ETX); d) OLSRv1 (hop count); e) OLSRv2 (ETX).*

hear a different radio on node 5. We first comment on the results achieved by OLSRv1 (with the hop count metric) in the case of UDP traffic. First, we consider the flow from node 3 to node 6, which is initially routed along the path with the minimum hop count (3-5-2-6). As explained earlier, node 3 continues to send packets to the interface of node 5 operating on channel 1 (10.0.0.6) until the link to this interface is considered lost, which usually happens 6 s after reception of the last HELLO message. At this point (34.0 s in Fig. 3d), since the new link to interface 10.0.1.6 is not yet symmetric, node 3 chooses an alternative path (3-1-0-4-6) to the destination. When the new link is established, node 3 starts sending packets along the shortest path again. This experiment shows that an alternative path, if available at the node immediately upstream of the link that fails due to the channel switch, is used as soon as such link is marked as lost, thus reducing the interruption in the delivery of packets (4.5 s as shown in Fig. 3b).

Concerning the flow from node 6 to node 3, we observe that node 6 (like all the other nodes but 3 and 5) is unaware of the lack of connectivity between nodes 3 and 5 until it receives one of the TC messages generated by nodes 3 and 5 when the link between them becomes lost. To limit overhead, TC messages are not forwarded immediately, but are grouped with other received TC messages and transmitted in a single packet along with a HELLO message. Such a mechanism delays the spread of information, with the consequent formation of routing loops. Indeed,

when node 5 marks its link to node 3 as lost, it starts sending packets to node 2 in an attempt to find an alternative path. Node 2, however, has not yet received an updated TC message and continues sending packets back to node 5. It is interesting to note that an updated TC message arrives at node 6 (40.5 s in Fig. 3d) when the new link between nodes 3 and 5 has been already established (38.6 s). Thus, although the current path is valid, node 6 decides to send packets destined to 3 along an alternative path (6-4-0-1-3). Later (42.5 s), node 6 receives a TC message carrying the information that the connectivity between nodes 3 and 5 has been restored, and switches back to sending packets along the minimum hop path. However, not all nodes have received the latest topology update, and hence, a new routing loop occurs, which leads to a second brief interruption (1.0 s) in the delivery of packets. This experiment shows that nodes upstream of the link that fails due to the channel switch attempt to find an alternative path to the destination as soon as they are informed of the link failure (via TC messages). However, routing loops can occur due to nodes having inconsistent views of the network topology. Also, route oscillations are generated that cause network instability. In this regard, the slow propagation of TC messages (to limit the overhead) accentuates such problems.

We now comment on the results achieved by OLSRv2 (with the ETX metric) in the case of UDP traffic, starting with the flow from node 3 to node 6. By using a link metric other than the hop count, it is in theory possible to switch to an alternative path before a link on the current path is marked as lost. It suffices that the cumulative quality of the alternative path is better than that of the current path. However, this does not happen in our experiment. Indeed, node 3 prefers the alternative path over the current path (which includes a failed link) only when the validity of the last HELLO message received from 10.0.0.6 expires (48.0 s in Fig. 3e). The reason is that the quality of a link degrades very slowly, as described earlier. As a consequence, the availability of alternative paths is not exploited, and the interruption in the delivery of packets is the same as that observed in case I-B. Likewise, the alternative path continues to be used well beyond the time when the new link between nodes 3 and 5 is established (51.8 s). This is due to the fact that the quality of the new link increases very slowly. Similar observations can be made for the flow from node 6 to node 3. Due to the slowly decreasing quality of the link between 10.0.0.4 and 10.0.0.6 (which is carried in the TC messages sent by nodes 3 and 5), node 6 continues to use the path including such link for a long time (up to 49.9 s). Thus, flows in both directions experience a long interruption (around 20 s) and do not benefit from the availability of alternative paths.

In case of TCP traffic, Figs. 3b and 3c show that the same observations as before can be made: the interruption in the delivery of TCP traffic lasts longer than that of UDP traffic due to the TCP congestion control mechanism. Finally, we performed experiments in a seven-node topology with a channel switch of the same kind as case I-A (fixed neighbor-to-interface binding). We have shown that, provided the routing protocol is not too quick at removing links, this kind of channel switch does not cause changes in the routing tables. Hence, routing loops do not occur, and the interruption in the delivery of packets coincides with the lack of connectivity at the MAC layer. When multiple mesh nodes are present, there is a chance that the connectivity at the MAC layer is restored quicker than described earlier. Indeed, if there is another radio operating on the new channel that can be heard by either of the two radios switching channels, these radios perform one channel scan and decide to join the IBSS advertised by the other radio, thus considerably reducing the duration of the interruption (around 1 s).

If there exists another radio operating on the new channel that can be heard by either of the two radios switching channel, these radios perform one channel scan and decide to join the IBSS advertised by the other radio, thus considerably reducing the duration of the interruption.

Scenario III: 13-Nodes Topology

We now consider a network of 13 mesh nodes crossed by 6 traffic flows (3 additional hosts acting as senders and 2 hosts acting as receivers are connected to the mesh network). We perform five different experiments, each characterized by a number of channel switches varying from one to five. Each experiment is repeated 20 times, each time with a distinct set of radio interfaces switching channel. Every channel switch changes the neighbor-to-interface binding, given that with fixed neighbor-to-interface binding it is possible to ensure that the interruption in the delivery of packets only lasts for the time required to restore the connectivity at the MAC layer. For each repetition of all the experiments, we collect the duration of the interruption in the delivery of packets experienced by each flow. The plots in Fig. 4 summarize the distribution of the values collected for each experiment. In particular, a white box spans from the first quartile to the third quartile, circles represent outliers (i.e., values greater [less] than the third [first] quartile plus [minus] 1.5 times the inner quartile range), a vertical line spans from the minimum to the maximum values (excluding outliers), and a horizontal segment indicates the median.

In this section, we also present and evaluate a technique inspired by the analysis of the experiments illustrated in the previous sections, which aims at reducing the duration of the interruption caused by a channel switch. We observe that, if the neighbor-to-interface binding changes, the link established on the previous channel fails in a permanent manner upon channel switching. Hence, instead of waiting for the link quality to drop below the threshold indicating a usable link, it would be more efficient for the nodes switching channels to immediately mark the link as lost. Our proposed technique, which we refer to as *link invalidation*, provides that nodes involved in a channel switch immediately mark the link on the previous channel as lost and advertise that by sending out both a HELLO message (to inform neighbors) and a TC message (to inform all other nodes). In such a way, the nodes involved in a channel switch can immediately use an alternative path, if available, and the link failure is reported sooner to the nodes upstream of the failed link, which can find alternative paths as well. We remark that link invalidation would be counterproductive in fixed

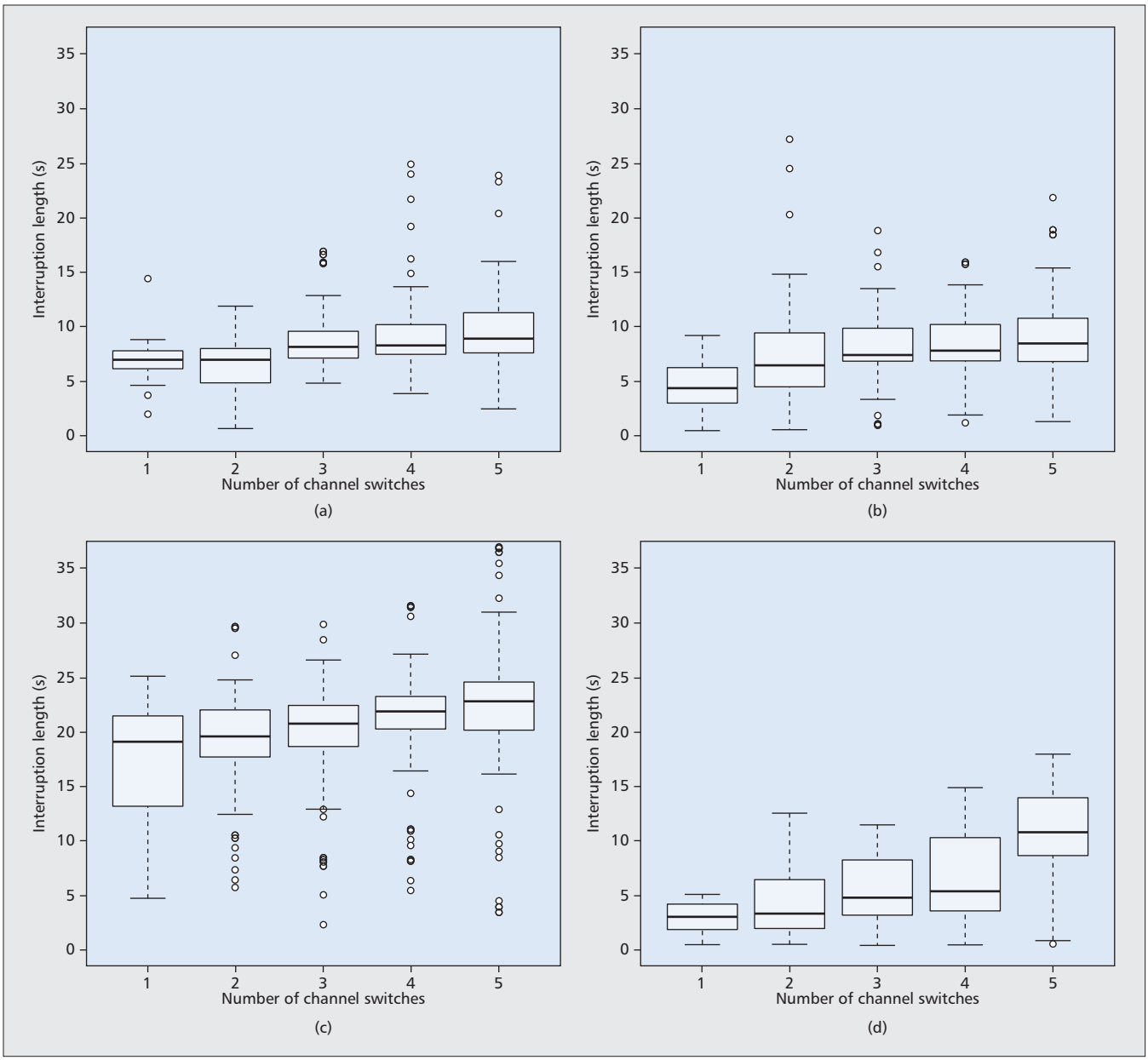

Figure 4. *Scenario III: 13-nodes topology: a) OLSRv1 (hop count); b) OLSRv1 (hop count) with link invalidation; c) OLSRv2 (ETX); d) OLSRv2 (ETX) with link invalidation.*

neighbor-to-interface binding (where the link failure is temporary). Hence, link invalidation should only be used when the neighbor-to-interface binding changes. However, we believe that nodes are able to determine whether or not they need to apply the link invalidation. Indeed, in centralized channel re-assignment (as in our experiments), the type of channel switch can be communicated by the centralized entity, while in distributed channel re-assignment, nodes are able to recognize the type of channel switch by themselves.

The results shown in Fig. 4 suggest the following observations:

- For all the routing protocols, the median value of the duration of the interruption typically increases with the number of channel switches. This result is rather expected, since the higher the number of channel switches, the higher the number of links that fail, and hence the more difficult it is to find alternative paths that are not affected by such link losses.
- The link invalidation technique applied to OLSRv1 (Fig. 4b) allows some gain over plain OLSRv1 to be attained (Fig. 4a). Such a gain decreases with the number of channel switches. The reason is that the advantage of the link invalidation technique is alternative paths can be used sooner. However, finding alternative paths becomes harder with the increase in the number of channel switches.
- The link invalidation technique applied to OLSRv2 (Fig. 4d) allows a considerable gain over the plain OLSRv2 to be achieved (Fig. 4c). The reason is that OLSRv2, due to the slow degradation of the link quality, is heavily penalized by the time required to discard a failed link, while the link invalidation technique enables the failed link to immediately be discarded.

• OLSRv2 with link invalidation performs slightly better than OLSRv1 with link invalidation. This is likely due to the fact that the threshold used by OLSRv2 to denote a usable link is very low, and hence, the link on the new channel is used sooner by OLSRv2 than OLSRv1.

Finally, we note that we observed frequent interruptions in the delivery of packets when using OLSRv1, even in the absence of channel switches. This is likely due to the fact that the loss of a few control messages is sufficient to discard a link. Loss of packets is not unlikely in wireless networks, due to environmental noise and collisions (caused, e.g., by hidden nodes or simultaneous expiration of the backoff timer on multiple nodes). Hence, the procedure employed by OLSRv1 to discard links turns out to be *excessively* quick.

Conclusions

In this article we present the results of a thorough experimental study we conducted to evaluate the impact of channel switches on the performance of a multi-radio wireless mesh network using a distributed hop-by-hop routing protocol. The main findings of our study can be summarized as follows:

• Our experiments showed that a channel switch causes a non-negligible interruption in the connectivity among nodes, which is on the order of 10 s. The analysis of these experiments revealed that results are determined by one particular aspect of the routing protocol: the procedure adopted to consider a link as established/lost. Such a procedure is usually based on a measure of the link quality as a function of the percentage of control messages (e.g., HELLO messages) received.

• In the case of fixed neighbor-to-interface binding, the link failure is temporary; hence, routes should not be modified. To this end, the link must not be marked as lost before the connectivity at the MAC layer is restored. Restoring the connectivity at the MAC layer may take the time required to perform one channel scan and join an IBSS (in the order of hundreds of milliseconds), in case another node is already advertising an IBSS using the new channel, or the time required to conclude that there is no available IBSS to join and create a new IBSS (a few seconds) otherwise. Thus, in the latter case, routing protocols that are quick at marking a link as lost if no messages are received on that link may incur longer interruptions due to the additional time required to re-establish the link. To avoid such additional time, the time it takes the MAC layer to determine that no IBSS is available to join should be set to a value less than the time the routing protocol takes to consider a link as lost if no message is received on that link. In such a way, the interruption in the delivery of packets coincides with the time required to restore connectivity at the MAC layer, and no change is made to the routing tables, even for routing protocols that are quick at marking a link as lost.

• In the case of neighbor-to-interface binding changes, there is the need to update the routing tables, since the link on the previous channel fails permanently and must be discarded in favor of the link on the new channel. In this case, therefore, routing protocols that are quicker at establishing/removing links exhibit shorter interruptions in the delivery of packets. Also, routing protocols that quickly update the link quality manage to efficiently exploit the availability of alternative paths. Indeed, the faster the quality of the link on the previous channel degrades, the sooner an alternative path, if available, is used. However, the information that a link has been established/removed takes some time to propagate among all the network nodes. Hence, distinct nodes may have inconsistent views of the network topology, which likely leads to routing loops and route oscillations. This situation is clearly even worse for multiple simultaneous channel switches. In the case of OLSR, the attempt to reduce overhead by aggregating multiple TC messages into a single packet is counterproductive, because delaying the propagation of TC messages can only accentuate such routing problems.

• Some link metrics (e.g., the one used by OLSRv2 in our experiments) are a function of the link quality in both directions. A node can directly measure the quality in the incoming direction and receive the quality in the outgoing direction from its neighbor. However, in case of link failure, the quality in the outgoing direction is no longer updated, thus contributing to the slow degradation of the quality of the failed link. Our experiments suggest that a slow degradation of the link quality is not desirable; therefore, we believe that proper countermeasures should be taken in the design of the routing protocol.

• A routing protocol that is too quick at discarding links has the drawback that frequent interruptions may occur due to the loss of few control messages (caused by environmental noise or collisions). Hence, making the degradation of link quality too fast is not a viable solution to reduce the interruption in the delivery of packets following a channel switch. As a preliminary solution, we propose a link invalidation technique (to be activated in case the neighbor-to-interface binding changes), which has the advantage that it does not interfere with the process of updating the link quality in normal conditions. Through experiments, we proved that such a technique allows even protocols that exhibit slow degradation of the link quality to contain the duration of the interruption in connectivity following a channel switch.

For all the routing protocols, the median value of the duration of the interruption typically increases with the number of channel switches. This result is rather expected, since the higher the number of channel switches, the higher the number of links that fail, and hence the more difficult it is to find alternative paths that are not affected by such link losses.

References

[1] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *ACM Mobile Computing and Commun. Rev.*, vol. 8, no. 2, Apr. 2004, pp. 50–65.

[2] J. Crichigno, M. Wu, and W. Shu, "Protocols and Architectures for Channel Assignment in Wireless Mesh Networks," *Ad Hoc Networks*, vol. 6, no. 7, Sept. 2008, pp. 1051–77.

[3] S. Avallone, G. Di Stasi, and A. Kassler, "A Traffic-Aware Channel and Rate Re-Assignment Algorithm for Wireless Mesh Networks," *IEEE Trans. Mobile Computing*, vol. 12, no. 7, July 2013, pp. 1335–48.

[4] A. Kanagasabapathy, A. Franklin, and C. Murthy, "An Adaptive Channel Reconfiguration Algorithm for Multi-Channel Multi-Radio Wireless Mesh Networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 10, Oct. 2010, pp. 3064–71.

[5] A. Franklin, A. Balachandran, and C. Murthy, "Online Reconfiguration of Channel Assignment in Multi-Channel Multi-Radio Wireless Mesh Networks," *Computer Commun.*, vol. 35, no. 16, pp. 2004–13, 2012.
[6] J. Galvez, P. Ruiz, and A. Skarmeta, "TCP Flow-Aware Channel Re-Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks," *Proc. IEEE MASS*, 2011, pp. 262–71.
[7] C. Chereddi, P. Kyasanur, and N. H. Vaidya, "Design and Implementation of a Multi-Channel Multi-Interface Network," *Proc. REALMAN*, 2006, pp. 23—30.
[8] P. Li *et al.*, "How to Effectively Use Multiple Channels in Wireless Mesh Networks," *IEEE Trans. Parallel and Distrib. Sys.*, vol. 20, no. 11, Nov. 2009, pp. 1641–52.
[9] D. D. Couto *et al*. "High-Throughput Path Metric for Multi-Hop Wireless Routing," *Proc. ACM MobiCom*, Sept. 2003, pp. 134–46.
[10] S. Avallone *et al.*, "Performance Evaluation of an Open Distributed Platform for Realistic Traffic Generation," *Performance Evaluation: An Int'l. J.*, vol. 60, no. 1–4, May 2005, pp. 359–92.

Biographies

Stefano Avallone (stefano.avallone@unina.it) received his M.S. degree in telecommunications engineering (2001) and Ph.D. degree in computer networks (2005) from the University of Napoli Federico II. He is currently an assistant professor with the Department of Computer Engineering at the University of Napoli. His research interests include computer networks, traffic engineering, QoS routing, and wireless mesh networks. He was a visiting researcher at the Delft University of Technology (2003–2004) and the Georgia Institute of Technology (2005). He is on the Editorial Board of Elsevier *Ad Hoc Networks*.

Giovanni Di Stasi (giovanni.distasi@unina.it) is a postdoctoral fellow at University of Napoli Federico II. He received his Laurea degree in computer engineering from the same university in 2007. He was a visiting researcher at INRIA Sophia Antipolis, France (2009), and Karlstad University, Sweden (2010). His current research interests include experimental reserch infrastructures and testbeds, routing and channel assignment algorithms for wireless mesh networks, and peer-to-peer traffic optimization.