E-CLIC European Collaborative Innovation Centres for Broadband Media Service	Place for logotype

Multipath Routing in Mesh Networks

Case Study Final Report

EUROPEAN UNION
European Regional Development Fund

Investing in the future by working together for a sustainable and competitive region

The Interreg IVB North Sea Region Programme

I. Introduction

In wireless mesh networks (WMNs), nodes relay packets over the air interface towards the destination or internet gateways. WMNs are used already by several municipalities (such as Chaska in the USA) or user communities such as Freifunk (e.g. in Berlin, Germany or Vienna, Austria) in dense urban areas. Such mesh networks enable interesting services in addition to wireless internet access, such as content sharing, multicast video delivery, sensor network backhaul, or vehicular network infrastructure support. WMNs are an interesting architectural candidate for the future wireless internet because there is no need to wire the meshed access points. This allows a dense access network to be rapidly set up at reasonable cost using a different topology as compared to cellular systems. Such dense deployment is seen as a key mechanism to significantly increase capacity.

For higher capacity and reduced interference, it is mandatory that WMNs use multiple radios operating in parallel on a diverse set of channels [1]. In such multi-radio mesh networks, effective channel assignment (CA) is important as it impacts the topology and routing. Comparing to static CA schemes, semi-dynamic CA schemes re-assign channels on longer timescale (e.g. minutes or hours) to cope with external interference [2] or changes in traffic demand [3]. However, such schemes result in low performance due to problems with routing and network disconnections. Hybrid approaches [4] solve such problems by allowing an interface to switch dynamically among the fixed channels of neighbor nodes to support full connectivity while the other interface is fixed to avoid multi-channel deafness problems. Using more channels ideally increases channel diversity and capacity but will result in higher switching overhead. While switching cost is reduced with current and future hardware developments, switching on a per packet basis is still too costly. As a solution, once a node switches to a channel, it stays there for some time (e.g. 20 ms) [4]. This requires a layer 2.5 mechanism to buffer packets which are to be sent on a channel which is currently not tuned in. As a result, using more channels leads to inherent higher switching delay which can have a negative impact on e.g. voice services.

Typical routing protocols for wireless mesh networks such as Optimized Link State Routing (OLSR) use a single path to send packets from source to destination. This path is pre-computed based on link state information received through network wide dissemination of control packets. The consideration of more information than hop-count in the routing process has shown to be beneficial as for example link quality and physical layer data rate determines the quality of the end-to-end path. Example of metrics considering link quality are ETX [5], ETT [6] and WCETT [7]. In multi-channel mesh networks, also channel switching overhead and channel diversity needs to be considered as a routing metric [4]. However, a major drawback of current approaches is that a path is pre-computed and used as long as the path is available and shows a good enough metric. As a result, short term variations on link quality or actual channel switching state are not considered.

Recently, anypath routing [8] has been considered as an interesting alternative using multiple next hops for each destination, where each packet is broadcasted at MAC layer. The packet will be received eventually by one or more neighbor nodes, where one of those forwards the packet towards the destination. This approach is effective when dealing with different and fluctuating link qualities but requires a coordination mechanism at the MAC layer in order to determine, which neighbor should forward the packet. This might be very difficult to achieve in multi-channel mesh networks as different neighbors use different channels making it more difficult to overhear packet transmissions. On the other hand, multipath routing mechanisms have been developed where the source maintains multiple paths towards each destination. While multipath routing can achieve effective load balancing, switching cost cannot be reduced effectively as each sub-path still is subject to channel switching. Also, each path is used as long as it does not break, even if it becomes sub-optimal over time.

The key contribution of this paper is the development of a novel multi-channel anypath routing and forwarding approach for multi-radio mesh networks. The key idea of our approach is that the routing layer maintains for each destination at each node a set of alternative next hop candidates, where each candidate has a ”good enough” path to the destination and is listening on a different channel. A novel layer 2.5 scheduling and forwarding component decides then for each packet which of those forwarding candidates to send the packet to based on the channel switching state. Using channel switching scheduling information is beneficial as packets can be forwarded earlier minimizing the total end-to-end delay and reducing the likelihood of packet loss due to buffer overflow. However, a forwarding mechanism based on local information cannot guarantee a global improvement. The approach was implemented and tested on a multi-radio mesh test bed KAUMesh [9]. A detailed performance evaluation using different topologies and traffic patterns shows that both delay and jitter can be reduced significantly.

The remainder of this paper is structured as follows: Section II reviews the concept of hybrid multi-channel multi-radio mesh networks and shows the design of our routing and forwarding approach. Section III gives an overview on our test bed and measurement results. Finally, section IV concludes the paper.

II. Multi-Channel Anypath Routing

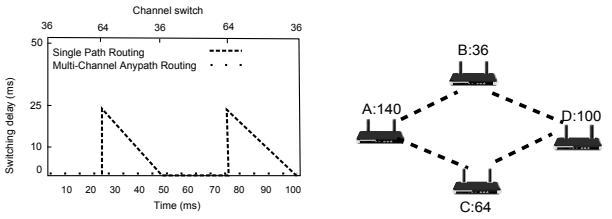
In this section, we review routing over hybrid multi-channel multi-radio systems and present the design of our routing and forwarding approach.

A. Routing and Forwarding in Hybrid Mesh Networks

In standard wireless mesh networks, each node forwards a packet over a single next hop towards the destination.

This forwarding is based on routing information which is maintained either proactively (using e.g. OLSR) or on demand (e.g. AODV). When using hybrid multi-radio multi-channel mesh networks, every node is equipped with at least two interfaces [4]. A fixed interface is used for receiving packets and a switchable interface can dynamically use any channel. A channel assignment protocol is used to determine, which channel to use for the fixed interface. Possible approaches are aware of interference caused by forwarding or taking into account traffic demand. Once a channel has been (re-)assigned to the fixed interface, this information is broadcasted to the two-hop neighbors so that they can associate the neighbor information with the fixed channel they use for receiving packets.

While switching time is reduced due to better hardware capabilities (e.g. on our Atheros platform we achieve a channel switching time of 3 ms), it is still high overhead to switch on a per packet basis. Therefore, nodes maintain a queue per channel and use a table to decide which channel a given packet should be sent to [4]. This table is updated based on broadcasted control information indicating the fixed channel of the neighbor nodes. Therefore, if a packet is to be sent from node A to neighbor B, which has the fixed interface tuned to channel 36, node A just puts the packet into the queue associated with channel 36. A channel scheduler is then responsible to switch the interface to a certain channel (if there are packets in the given queue). Once tuned to a specific channel, it will stay there for some time (e.g. 20 ms) to reduce the switching overhead and send the packets which have been enqueued for the specific channel. Such channel scheduler can operate in a round-robin fashion [4] or take into account priorities for data flows [10].



(a) Switching delay at node A (b) Topology

Fig. 1. Example: Switch delay

A specific problem of such hybrid multi-channel multi-radio mesh networks is the high delay they impose due to channel switching, which accumulates over multiple hops. While using more channels increases the channel diversity and capacity, it also leads to higher switching delay, as more channels need to be served per node. In Figure 1(b), node A sends packets to node D as destination. Let us assume the routing protocol selects the route via node B and let us assume there is additional background traffic between nodes A and C. Therefore, node A need to switch between channels 36 and 64. In single path routing, after 20 ms packets from A to D experience large delay as the switchable interface at node A switches from channel 36 to 64 to serve the background traffic. Therefore, all packets sent from A to D are queued until the switchable interface is tuned again to channel 36. This leads to the delay profile shown in Figure 1(a).

For the given topology, node A has however two paths to the destination which have the same quality. Ideally, node A could transmit packets towards destination D via both B or C using any available path given that it is good enough. Once the interface is tuned to channel 36, we would send the packets then via the upper node B, once the interface is tuned to channel 64, we would send it via the lower neighbor node C. In that way, we could in this example completely eliminate the switching delay. So our goal is to develop a novel Multi-Channel Anypath Routing and Forwarding approach based on the principle to use any available "good" path and let the forwarding module decide on a per packet basis which next hop/channel to use based on local information. While such forwarding can be very flexible designed based on utility functions, we present a simple version in this paper that minimizes delay in order to increase the user satisfaction for VoIP traffic. However, future versions may also consider load balancing or interference.

B. Routing Algorithm

The key idea of our routing approach is to identify multiple alternative path segments towards a destination which might be flexibly combined on a per packet basis. Therefore, we need to identify for each destination a set of multiple next hops, which have good enough paths towards the destination. Ideally, those next hops are reachable via different channels to dynamically later select the channel with smallest cost. The set of next hops found by the routing algorithm forms a candidate set S_d for the given destination d. This allows the forwarding module later on to select for each packet a different next hop out of the candidate set.

In multi-radio multi-channel mesh networks, the cost towards the destination is composed of the path cost (which can be estimated using the Expected Transmission Time - ETT routing metric [6]) and the switching cost ([11]) accumulated along the given path. The information on ETT does not change frequently and is updated with new routing information. However, the local node has a very precise view on the state of its channel switch operation, which changes on a per packet basis. Therefore, when a packet is to be forwarded, the candidate with the lowest cost for the specific packet is selected and so the path taken is determined on-the-fly. While in single path routing we maintain for each destination a single next hop neighbor, we need to develop an algorithm which allows us to identify for each destination AND channel a next hop which will be inserted into the candidate set. Therefore, the candidate set for each destination will have at most n entries with n equal number of channels.

We now present the design of our routing algorithm. Given a graph $G = (V, E)$ and a source node a, the algorithm

calculates the fastest path for every channel $f \in F$ to each destination d.

Algorithm 1 Routing(G, a)

```
1: for all nodes n ≠ a in V do
2:    S_n ← ∅
3:    for all channels f in F do
4:       C_{n,f} ← ∞
5:       H_{n,f} ← ∅
6:    end for
7: end for
8: C_{a,∅} ← 0
9: U ← V
10: R ← ∅
11: while U ≠ ∅ do
12:    n ←extract_min(U)
13:    R ← R ∪ {n}
14:    for all edges (n, d) in E do
15:       if hops(n) = 0 then
16:          f ← get_channel((n, d))
17:          new_cost ← ett((n, d))
18:          if new_cost < C_{d,f} then
19:             C_{d,f} ← new_cost
20:             H_{d,f} ← 1
21:             S_d ← n
22:          end if
23:       else
24:          for all channels f in F do
25:             new_cost ← C_{n,f}+ ett((n, d)) + sc((n, d))
26:             if new_cost < C_{d,f} then
27:                C_{d,f} ← new_cost
28:                H_{d,f} ← H_{n,f} + 1
29:                S_d ← S_{n,f}
30:             end if
31:          end for
32:       end if
33:    end for
34: end while
```

This algorithm is run independently at every node. Lines 1-10 initialize data structures S, C, H, U and R. S denotes the set of all next-hop candidates, C the set of all path costs for each node and channel, H the set of all hop counts for each node and channel, U the set of all unvisited nodes and R is the set of all visited nodes. Lines 11-32 visit each node, starting with the one having the smallest cost (line 12), and check all its neighbors for a better path. For the first hop (lines 16-21), we do not consider the switching cost $sc()$ as it will be considered later in the forwarding algorithm. This is because in the routing algorithm we just have averaged information available, which might not reflect the actual state of the channel switch operation. For all other links, we consider both the switching cost $sc()$ and $ett()$ in the calculation of the best metric path (lines 24-31). If a better path is found on a channel f to a neighbor d (lines 18 and 26), the new_cost is stored as $C_{d,f}$ (lines 19 and 27), the number of hops is stored as $H_{d,f}$ (lines 20 and 28) and the next hop is added to the candidate set S_d (lines 21 and 29). The outcome of this algorithm is then for each destination a candidate set S_d. Each member in the candidate set is reachable via a different channel and has for this given channel the minimum cost path to the destination.

C. Next Hop Candidate Selection and Forwarding Algorithm

The outcome of the routing algorithm informs our layer 2.5 forwarding module with a candidate set of next hop neighbors which can reach the destination via a different channel. Therefore, we need to design an algorithm which selects at any given time for each packet the ”best” neighbor and channel to forward the packet on. As our intention is to minimize the total end-to-end delay, which is composed of accumulated switching cost and ETT values, we need to consider the local switching cost, which is required for the channel switch module to tune the interface to the given channel. This depends on the channel scheduling mechanism. In our case, the channel scheduler serves channels that have a packet enqueued in round-robin order [4]. Once the interface is tuned to a given channel, it stays there for minimum 20 ms. If after 20 ms there are still packets to be sent on the given channel, the time can be increased to max 60 ms.

In addition, we want the forwarding module to exploit path diversity in order to effectively reduce the switching cost. Here, the idea is that we can tolerate a longer path if that path at the given time results in lower total cost. As this cost is determined during runtime due to the channel switching state, we allow a packet to deviate from the best metric path by a certain number of hops. Therefore, we introduce a path diversity factor $\alpha \geq 1$, which determines the maximum length path that a packet is allowed to follow. The source node then calculates the maximum number of hops H that a given packet can travel by multiplying the required number of hops for the best metric path with α. Clearly, there is a tradeoff. A small α leads to low path diversity and will not provide many opportunities for packets to minimize the switching time. A large α value may lead to routes which are either very long or may contain loops. Therefore, the source node inserts into each packet the maximum allowed number of hops H and every forwarding node decreases the value by one.

Given a set of next hop candidates S, where each candidate is reachable via a different channel, and a remaining number of allowed hops H, the layer 2.5 forwarding algorithm selects the next hop N for any given packet as follows:

Line 1 initializes C, which stores the lowest cost (line 6). Lines 2-10 loop through each candidate $i \in S$ and check if we can reach the destination via this neighbor and channel within H hops (line 3). If so, we calculate the total cost (line 4), which is composed of the path cost as given by the routing algorithm and the *actual* switching delay. If the cost C_i is lower than C (line 5), the candidate i is stored as N (line 7). The outcome of this algorithm is then the best candidate N as

Algorithm 2 Next_Hop_Candidate_Selection(S, H)

```
1: C ← ∞
2: for all candidates i in S do
3:    if required_hops(i) ≤ H then
4:       C_i ←switch_delay(i) + path_cost(i)
5:       if C_i < C then
6:          C ← C_i
7:          N ← i
8:       end if
9:    end if
10: end for
```

perceived by the local node. The packet will then be enqueued into the channel queue which is associated with this neighbor node N.

D. Implementation Details

We implemented the proposed routing and forwarding algorithms as extensions to [4]. We modified Net-X to use OLSR as basic routing protocol [12] instead of using AODV. The OLSR routing protocol was extended to support multiple routing metrics in order to carry the switching cost information and ETT values [13]. Also, we used the ETT implementation from [6]. The anypath and forwarding extensions (AP-OLSR) consist of two components, a user space application and a kernel space module. The user space application is responsible to find multiple next hop candidate neighbors for all destinations, where each neighbor is reachable via a different channel. It forwards for each destination the next hop candidate set along with the switching cost, ETT values and hop count information to the kernel module using ioctl messages. The kernel module is an extension of the Net-X bonding module [4] and handles the next hop selection on a per packet basis. It also calculates the maximum allowed number of hops H for each packet created and inserts this value into a special IP-Options field within the IP header.

E. Interaction between bonding and WLAN driver

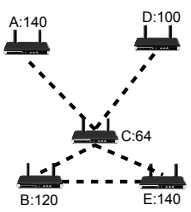
A packet arriving at the bonding module to be sent on the network is placed in a packet queue, where it is buffered until the required channel is tuned on the sending network interface card. The bonding module has multiple packet queues, one for each channel. The network driver has one packet queue, which is flushed when the channel is changed. Therefore, before a channel switch is requested the bonding module stops the flow of packets to the driver and waits until the driver's queue is drained. During our evaluation we discovered that the madwifi driver was not able to give an accurate value of number of packets in the queue. This led to packet loss when the driver incorrectly reported that the queue was empty.

The packet loss problem was solved by exchanging the madwifi driver to a newer driver named ath5k (developed within the same project). Also the bonding module was exchanged to match the ath5k driver. The new bonding module splits the queues into two groups, prio and normal, and has a task runner that determines which queue to be served. When the task runner is called, it starts by iterating through a list containing queues. The next step is to check if there exist packets in the current queue type (prio or normal). For a packet queue type switch to take place, the current queue type cannot contain packet and the specified minimum time, if any, must have been reached. A channel switch is requested by the task runner only if the packet queue type is not changed, and is performed using an iw handler to make an ioctl call to the driver. The information sent to the driver from the bonding module is the concerning network device and the desired channel.

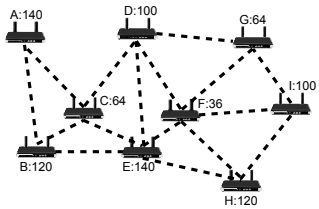
The ath5k driver has a packet queue size of 200 packets, which must be drained before a packet switch can be performed in order to not lose any packets. The function that is called to switch channel, is modified to check whether the queue is empty or not. The channel switch is performed if the queue is empty, otherwise an error message EAGAIN is returned to the bonding module.

III. Evaluation

Our routing and forwarding approach has been evaluated in the KAUMesh test bed [9], which is a multi-radio multi-channel network deployed at the Karlstad University Campus. Each node consists of Cambria GW2358-4 Network Computer, equipped with three Atheros 802.11a/b/g mini-PCI cards (WLM54AG with 5212 chipset). The tests were performed using 9 nodes, located as in figure 2(a). The bandwidth of each link was statically set to 6 Mbps. We used both UDP (using mgen [14], packet size 1024 bytes) and TCP (using iperf [15], packet size 536 bytes). The packet rate for UDP varies through all tests and is described for each experiment separately.

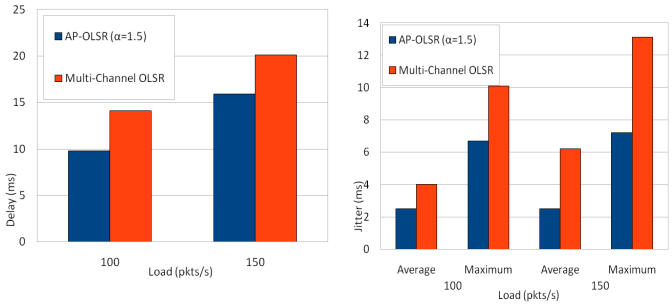


(a) *Topology for Scenario I*



(b) *Topology for Scenario II*

We compare the performance using three approaches. In the first approach, we use standard single-channel configuration, where we configure channel 36 for all links. Here, we are using Olsrd [16] as routing protocol which we extended using ETT routing metric according to [6]. By using only one channel, packets will not experience any switching delay but only one node within a collision domain can transmit data at the time leading to low throughput. We also compare against our *Multi-Channel OLSR* implementation, where we modified the hybrid channel assignment approach Net-X[4] to use OLSR instead of AODV. All other components have been used as in the



(c) Delay (d) Jitter

Fig. 2. Delay and jitter for different loads (*Scenario I*)

original implementation. The routing metric considers ETT and switching cost. The channels for the receiving interface are shown in figure 2(a). Finally, our approach *AP-OLSR* is based on Section II as described above, where we modified the OLSR and Bonding module accordingly. For fair comparison, we used the same channels for the receiving interface as for the *Multi-Channel OLSR*. The main differences are the ability of *AP-OLSR* to select different next hops considering the channel currently tuned on the sending interface and the possibility of using path diversity when beneficial ($\alpha > 1$).

A. Scenario I: Small topology with extra hop

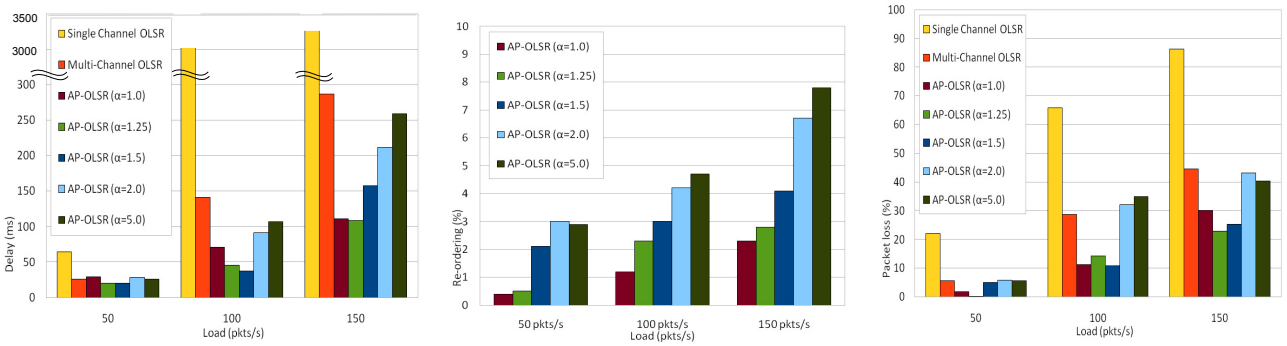
For this scenario, five out of nine nodes are used and figure 2(b) indicates the topology and channels on the receiving interface for the multichannel cases. We compare Multi-Channel OLSR with our AP-OLSR (using $\alpha = 1.5$). We use two UDP flows with equal packet rate (Flow 1 from node A to node E and Flow 2 from node D to node B). We vary the packet rate from 100 to 150 packets per second. Figure 2 shows delay and jitter averaged across both flows. Clearly, AP-OLSR reduces average delay from around 15 ms to 10 ms (for the 100 pkts/sec case) while at the same time reducing both maximum and average jitter significantly. Under higher load, the reduction of jitter is even more visible. Having smaller jitter is beneficial for VOIP traffic as it reduces the loss at the application due to jitter buffer overflow.

Using AP-OLSR with $\alpha = 1.5$ allows path diversity and the forwarding module can effectively minimize the switching delay. For example, in multi-channel OLSR packets for Flow 1 are sent via node C directly to node E. This leads to large switching delay as packets for Flow 1 are enqueued in node C to be sent on channel 140 to the destination E while channel 120 serves packets from Flow 2. In contrast, AP-OLSR gives the packets of Flow 1 the opportunity to be delivered at node C via an extra hop B. This is because the best metric path has two hops and $\alpha = 1.5$ allows packets to travel at most 3 hops. Therefore, when node C tunes the switchable interface to channel 120, it can deliver not only packets for Flow 2 to their destination B, but at the same time can forward packets of Flow 1 to node B which can immediately forward them to the destination E. Consequently, while some packets may travel an extra hop, the overall switching delay and thus end-to-end delay is reduced significantly. Using TCP instead of UDP, AP-OLSR sustained a throughput 50% higher than when using Multi-Channel OLSR.

B. Scenario II: Impact of Path Diversity

In this scenario, all nodes in figure 2(a) are used. We created one UDP flow F1 (100 packets/sec, packet size 1024 bytes) from node A to I. In addition, we used two background flows with equal date rate: F2 from B to D and F3 from E to G. We vary the rate for F2 and F3 from 50, 100 to 150 packets/sec and compare the performance using Single Channel OLSR, Multi-Channel OLSR and AP-OLSR (with $\alpha = \{1.0, 1.25, 1.5, 2.0, 5.0\}$). Figure 3(a) shows the average delay for F1 under different offered load of the background flows F2 and F3. As we can see, with single channel operation the delay of F1 increases significantly with the offered load. This is because mesh nodes need to serialize reception and forwarding on the same interface/channel leading to high contention and packet drops due to collision (see also figure 3(c)). In contrast, for multi-channel OLSR the delay is significantly lower due to the multi-channel operation, which allows nodes to forward packets on the switchable interface while in parallel receive on the fixed interface. However, when background load increases from 50 to 100 or 150 packets/sec, the switching overhead also increases as more packets need to be served at nodes C (F) to forward packets of F2 (F3) on channel 100(64) to node D(G). This increases the waiting time in the bonding layer for packets belonging to F1 at intermediate nodes and consequently leads to high end-to-end delay.

When using AP-OLSR with no path-diversity ($\alpha = 1.0$) we get even smaller delay compared to multichannel OLSR. For example, when the background traffic is 150 pkts/s, AP-OLSR reduces average delay from around 280 ms to around 110 ms. This is because of the effective reduction of switching overhead. When we increase the path diversity for AP-OLSR we observe that the best performance is achieved with $\alpha = 1.5$, which allows paths for the F1 that traverse not more than 6 hops. If we allow more path diversity we observe that the



(a) Average delay (b) Re-ordering (c) Packet loss

Fig. 3. Results for different offered loads for F1 (*Scenario II*)

delay increases, which is due to some packets circulating some limited time in loops or too long paths.

Figure 3(b) shows the amount of re-ordering introduced by our approach. As expected, low path diversity leads to low re-ordering, which increases with larger α values and background traffic volume. While this might have negative impact on TCP performance, mechanisms such as [17] can be used to mitigate those effects. Another option could be to make the layer 2.5 forwarding flow aware, which would increase the complexity. Finally, Figure 3(c) plots the impact of background volume and path diversity on packet loss of Flow F1. Here, we can observe the same trend as for the delay. We can conclude that AP-OLSR shows the best performance, but also that the value of α is important and may depend on the scenario.

IV. Conclusions

Reducing the switching overhead in hybrid multi-channel multi-radio wireless mesh networks is crucial to improve performance. In this paper, we have developed a novel routing and forwarding algorithm based on anypath and multipath concepts. The key idea of our approach is that intermediate nodes maintain multiple next hop candidates for each destination, where each alternative candidate is reachable via a different channel. A layer 2.5 forwarding module decides then, based on local information, which channel and next hop to use for any packet taking into account the state of the channel switch module. This helps effectively in reducing the switching cost while at the same time having the benefits of the hybrid approach in terms of capacity and channel diversity. The proposed routing and forwarding mechanism has been implemented and evaluated in a multi-channel multi-radio wireless test bed KAUMesh. Our evaluation has shown that significant reduction in end-to-end delay and jitter is achievable due to the lower switching cost and exploitation of path diversity, even if path diversity may lead to longer routes.

References

[1] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee and R. Morris, "Capacity of ad hoc wireless networks," *Proc. of 7th ACM MOBICOM*, 2001.

[2] K. N. Ramachandran and E. M. Belding and K. C. Almeroth and M. M. Buddhikot, "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proc. of INFOCOM*, 2006.

[3] Mansoor Alicherry, Randeep Bhatia, Li (Erran) Li, "Joint Channel Assignment and Routing for Throughput. Optimization in Multi-radio Wireless Mesh Networks," *Proc. of ACM MOBICOM*, 2005.

[4] P. Kyasanur, C. Chereddi, and N. H. Vaidya, "Net-X: System eXtensions for Supporting Multiple Channels, Multiple Interfaces, and Other Interface Capabilities," *Tech. Rep., University of Illinois at Urbana-Champaign*, August 2006.

[5] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, v 11, n 4, pp. 419-34, 2005.

[6] P.M. Esposito, M. Campista, I.M. Moraes, L. Costa, O. Duarte, M.G. Rubinstein, "Implementing the expected transmission time metric for OLSR wireless mesh networks," in *2008 1st IFIP Wireless Days*, pp. 5, 2008.

[7] R. Draves, J. Padhye, B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MOBICOM*, 2004, pp. 114-128.

[8] R. Laufer, H. Dubois-Ferriere, L. Kleinrock, "Multirate anypath routing in wireless mesh networks," in *2009 Proceedings IEEE INFOCOM.*, pp. 37-45, 2009.

[9] KAUMesh http://www.kau.se/en/kaumesh/ [Accessed July, 2010].

[10] Marcel C. Castro, Peter Dely, Andreas J. Kassler, Nitin H. Vaidya, "QoS-Aware Channel Scheduling for Multi-Radio/Multi-Channel Wireless Mesh Networks," in *Proceedings of the 4th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH), co-located with ACM MobiCom 2009*, Beijing, China, September 21st 2009.

[11] P. Kyasanur, "Multichannel wireless networks: capacity and protocols," *Ph.D. dissertation, University of Illinois at Urbana-Champaign, IL*, 2006.

[12] Marcel C. Castro, Peter Dely, Andreas J. Kassler, Francesco Paolo DElia, Stefano Avallone, "OLSR and Net-X as a Framework for Channel Assignment Experiments," in *ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH), co-located with ACM MobiCom 2009*, Beijing, China, September 21st 2009.

[13] A. Lavén, P. Hjärtquist, "Multimetric OLSR and ETT," in *5th OLSR Interop & Workshop*, October 2-4, 2009 in Vienna, Austria.

[14] Multi-Generator, Naval Research Laboratory (NRL) http://cs.itd.nrl.navy.mil/work/mgen/ [Accessed July, 2010].

[15] Iperf, Network measurement tool developed by NLANR/DAST http://iperf.sourceforge.net/ [Accessed July, 2010].

[16] UniK OLSR daemon software, Available: http://www.olsr.org/ [Accessed July, 2010].

[17] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: Balancing tcp over multiple paths in wireless mesh network," in *In Proc. of ACM MOBICOM*, 2008.