Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2007, Article ID 86510, 10 pages
doi:10.1155/2007/86510

Research Article

Evaluation of Cross-Layer Rate-Aware Routing in a Wireless Mesh Network Test Bed

L. Iannone, K. Kabassanov, and S. Fdida

Laboratoire d'Informatique de Paris 6 (LIP6), Centre National de la Recherche Scientifique (CNRS), Université Pierre et Marie Curie (Paris VI), 104 Avenue du President Kennedy, 75016 Paris, France

Received 29 June 2006; Revised 20 October 2006; Accepted 27 November 2006

Recommended by Marco Conti

Real deployments of wireless multihop networks, by Internet service providers (ISPs), have been slowed down by their poor performance and unreliability. The research community has already proved that efficient cross-layer routing, in particular rate-aware routing, can significantly improve performances. Nevertheless, this work has been done mainly by simulations, seldom being implemented in a real environment. We present in this paper the results we obtained by comparing the performances of the traditional routing approach based on the hop-count metric and the cross-layer routing approach based on the transmission rate metric. These measurements have been done on the `MeshDVNet` test bed we deployed in our laboratory. As a routing protocol, we used two versions (with and without cross-layer metric) of `MeshDV`, a simple routing protocol expressly designed for wireless mesh networks (WMNs). As our tests clearly show, cross-layer rate-aware metric gives important improvements, in both connectivity and throughput.

1. INTRODUCTION

Wireless mesh networks (WMNs) are an emerging two-tier architecture based on wireless multihop communications. A WMN is composed of wireless mesh routers (WMRs), which offer connectivity to clients by acting as APs, forming at the same time a self-organized wireless backbone. This backbone can be a self-standing network, simply offering interclient connectivity, or a local extension of the wired Internet, if a connection is available through one or more WMRs acting as gateways. In both cases, the WMN backbone is in charge of relaying all clients' traffic. Figure 1 shows an example of WMN, whose basic definition has been standardized by IETF [1]. Akyildiz et al. offer a comprehensive survey of WMNs and related issues [2].

Protocols and algorithms paradigms settled as consolidated solutions in the context of the wired world have shown heavy and unpredictable limits in the wireless networking context. As we will show with the results of the measurements we performed, classical routing approaches, based on the hop-count shortest path metric, are not sufficient anymore in this new context.

In the quest of finding routing metrics that correctly reflect the wireless link behavior, a myriad of routing protocols based on cross-layer metrics has been proposed. Although several works demonstrate that cross-layer routing, with well-designed metrics, may drastically improve capacity of multihop networks, they seldom have achieved a real implementation. This is essentially due to (i) the several lacks of current technology, which is not able to support some advanced features; (ii) the complexity of implementing a cross-layer solution into today's operating systems protocol stacks; (iii) the unrealistic assumptions of proposed theoretical solutions.

In this paper we show that cross-layer routing offers important improvements in real environments when it is really implemented. By some measurements of the delay and the throughput, we show that even a simple cross-layer metric like the physical transmission rate may offer good performances. Furthermore, we show that when connectivity is jeopardized using the traditional hop-count metric, cross-layer rate-aware routing still has a nonnegligible performance.

The measurements have been done on the `MeshDVNet` test bed, a WMN platform deployed in our laboratory. This

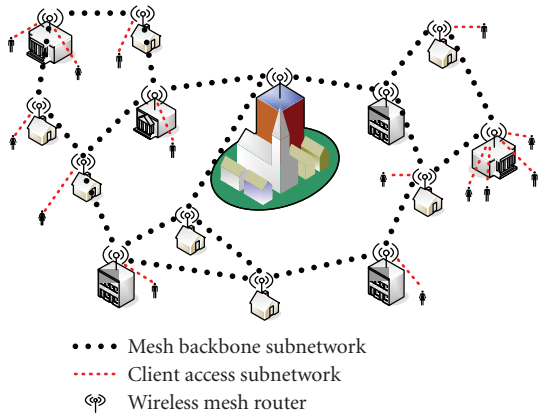


Figure 1: An example of wireless mesh network.

Figure 2: MeshDVBox: our wireless mesh router.

test bed is composed of MeshDVBoxes, custom WMRs built up from commercial components and open-source software. Since it is also totally IPv6-based [3], all measurements have been done using this protocol.

Our platform uses MeshDV, a routing protocol framework expressly designed for wireless mesh networks, composed of a fully modular architecture that eases its development. MeshDV combines proactive route computation for routers and on-demand path setup for clients. In particular, the proactive route computation on the mesh backbone can be done using the traditional hop-count metric as well as the physical transmission rate (cross-layer) metric.

The remainder of the paper is organized as follows. In Section 2, we describe MeshDVBox, our custom WMR, and give some details about the MeshDVNet test bed deployed in our laboratory. In Section 3, we present an architectural overview of MeshDV. Then in Section 4 we describe in detail all the tests we made and the results we obtained. In Section 5 we describe some related works, before concluding the paper in Section 6.

2. MESHDVBOX AND THE MESHDVNET PLATFORM

MeshDVBox, our WMR, is not a proprietary solution but is based on off-the-shelf components that can be easily acquired on the market. In particular, each of our WMRs consists in a Soekris net4521 box [4]. It is a PC-like architecture with a 133 MHz AMD ElanSC520 processor, 64 MByte SDRAM, and equipped with a 1 GByte Microdrive. It has also two PC-Card/Cardbus slots and a mini-PCI socket. Figure 2 shows a picture of a MeshDVBox.

Each MeshDVBox uses two wireless cards, both having their own external antennas. The first one is the client interface, working as an access point for client connections, using IEEE 802.11 b/g [5] technology. This interface is a Proxim 8470-WD b/g card [6] occupying one of the cardbus slots (cf. left-hand side of Figure 2). The second one is the mesh interface, working in ad hoc mode in order to form the mesh backbone with the other WMRs' peer interfaces, using IEEE 802.11a [7] technology. This interface is a NetGate 5354 MP Plus Aries2 4G a/b/g card [8] occupying the mini-PCI socket (cf. right-hand side of Figure 2). Since the two technologies we use (802.11a and 802.11 b/g) work in separate frequency bands, we have the nice property that the mesh backbone subnetwork and the client access subnetwork are physically independent.

All wireless interfaces are configured exclusively with IPv6. Clients can use the autoconfiguration features of IPv6 in order to obtain a valid global address. This allows avoiding the deployment of complex solutions like DHCP or NAT mechanisms on each WMR. Moreover, the routing demon MeshDV can leverage on the Neighbor Discovery Protocol mechanisms to manage mobility.

Each MeshDVBox runs NetBSD 3.99 current, which is the development branch of the NetBSD Project [9]. It is regularly updated in order to take advantage of the latest developments of the wireless cards driver.

In Figure 3 the test bed we deployed in our lab is depicted, where each MeshDVBox is named Sxx, with xx ranging from 01 to 12. The device named C01S is an access point used to connect the test bed to the wired LAN through the S04 MeshDVBox that acts as a gateway. On the wired LAN we have also an IPv6 DNS server and an IPv6-to-IPv4-proxy server, thus allowing to reach IPv4-only sites on the Internet. In the same picture the routes that are established by the routing demon are shown. Figure 3 is a simplified version of a snapshot of our supervision web page, used to monitor if WMRs are running and reachable. This page is publicly available (IPv6 only) at http://www.infradio-jussieu.lip6.fr/supervision/supervision-mesh-scott.html.

3. MESHDV OVERVIEW

MeshDV, the routing demon running on each MeshDVBox, has also been developed in our laboratory [10]. In MeshDV, the backbone becomes totally transparent to the clients, who do not need to embed any new feature (e.g., an ad hoc routing protocol). This means that if two clients associated to different WMRs wish to communicate, the set of WMRs

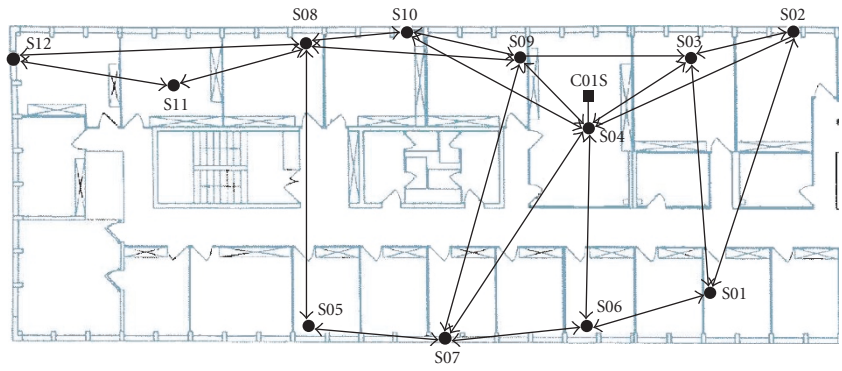


Figure 3: The MeshDVNet wireless mesh network deployed at the LIP6 laboratory.

will forward the traffic at the IP level. In particular, on the client subnetwork interface, the WMR acts in such a way to let local clients think that remote clients are in the local WLAN. Then, it is up to the WMRs to find out to which WMR the client is associated and to route the packets accordingly.

MeshDV has been expressly designed in order to take advantage of the two-tier architecture of WMNs. It performs proactive route computation on the mesh backbone, while path setup between clients is done on demand. This hybrid approach aims to ease the mobility management task, reducing the overhead introduced on the network in order to maintain active all connections of clients moving and associating to different MeshDVBoxes. Hereafter we give an overview of the MeshDV architecture including, for completeness, modules that manage wireless clients. Even if mobility measurements are out of the scope of this paper, a complete overview will help in the understanding of the results we present.

MeshDV is composed of the following four main modules (cf. Figure 4):

(i) client manager module;
(ii) NDP proxy module;
(iii) IPv6 forwarder module;
(iv) enhanced DV module.

The client manager module implements all the on demand mechanisms necessary to discover which MeshDVBox a client is associated to and to manage its mobility. Basically a multicast request is issued each time a local client wishes to communicate with a client associated to a different MeshDVBox.[1] The MeshDVBox, where the searched client is associated to, will reply to the query with a unicast message. This query reply is then stored in an appropriate data structure, for further use. The client manager module readily knows clients associated locally through system calls to the kernel.

[1] Recall that in IPv6 broadcast addresses do not exist anymore, multicast is used to send packets to multiple receivers.

MeshDV has a specific module that allows it to act as Neighbor Discovery Protocol proxy. This module manages ICMPv6 [11] requests issued by local clients in order to set up a communication to remote clients. The advantage is that clients do not have to perform any particular operation, since only standard ICMPv6 messages are exchanged between clients and MeshDVBoxes. In this way MeshDV behaves totally transparent for the users.

The NDP proxy module and the client manager module collaborate to set up a communication between clients associated to different MeshDVBoxes by providing a lookup mechanism and a standard interface toward clients. Once the communication is set up, data packets are transported on the mesh backbone using an IPv6-in-IPv6 tunneling approach. This task is performed by the IPv6 forwarder module. Such a kind of approach introduces a certain amount of overhead, due to the large IPv6 header, resulting in a smaller MTU (maximum transmission unit). Nevertheless, it has the nonnegligible advantage of ridding MeshDVBoxes along the path between the two clients from keeping state information about the ongoing communication. Only MeshDVBoxes at the edges of the communication path, more specifically their client manager module, have to be aware of the ongoing communication. In Figure 4 dotted lines show the way data traffic transits inside a MeshDVBox.

The last module (but not the least important) is the enhanced DV module, whose task is to maintain proactively a mesh of routes between all the MeshDVBoxes in the network, thus building the mesh backbone. This is a fundamental task, since the above-mentioned modules and mechanisms rely on the existence of these routes. The core of the enhanced DV module is an IPv6 implementation of the DSDV [12] routing protocol, improved in order to collaborate with the other modules of MeshDV. DSDV (destination-sequenced distance vector) is a simple distance vector routing protocol based on the Bellman-Ford algorithm. As the name suggests, for each destination, a sequence number is added to the update messages, in order to have loop-free and fresh paths.

In order to perform the comparison between the hop-count metric and the rate-aware metric we implemented two different versions of the enhanced DV module.

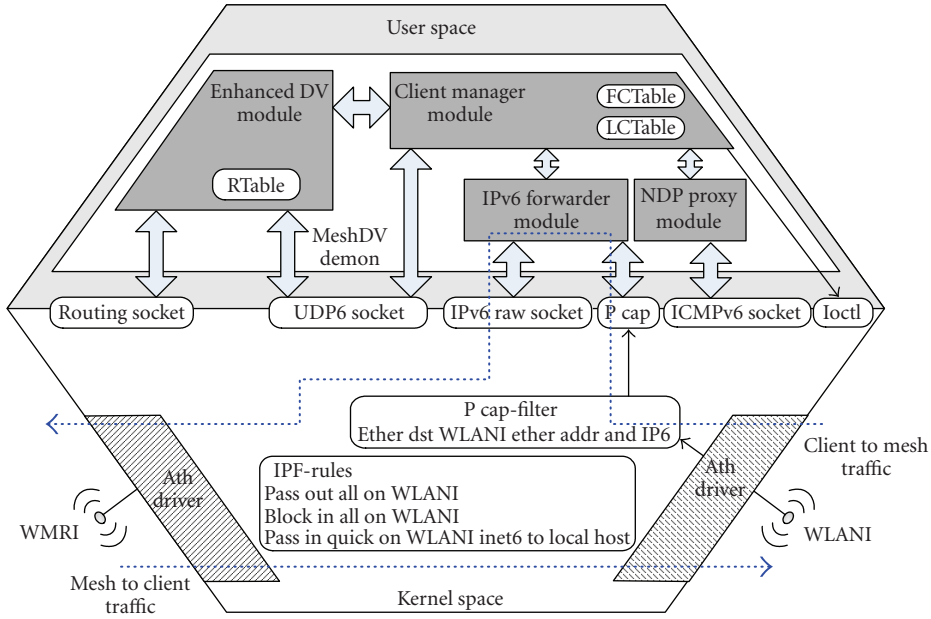


Figure 4: `MeshDV` architecture.

The version using the hop-count shortest path metric builds routes with the same method used for intradomain routing in the wired Internet. To each link (hop) of the network it is assigned a cost equal to one, thus routes are chosen by minimizing the following cost function:

$$C_{\mathrm{HC}}(\mathrm{Path}) = \sum_{\forall (i,j) \in \mathrm{Path}} 1, \tag{1}$$

where Path is the set of all links from the source to the destination. We call this version `MeshDV-HC`.

The second version uses the raw transmission rate as a metric and is the result of our efforts to implement the cross-layer metrics we proposed in [13]. We call this version `MeshDV-CL`. In [13], physical transmission rate, interference, and packet error rate are coupled with transmission power control in order to use them as routing metrics and improve the transport capacity of the WMNs' backbone.

The raw transmission rate metric allows to select links offering higher transmission rate, compared to the hop-count metric. In IEEE 802.11a, which is the technology we use on the mesh backbone, the possible values are 54 Mbps, 48 Mbps, 36 Mbps, 24 Mbps, 18 Mbps, 12 Mbps, 9 Mbps, and 6 Mbps. The rate used by a WMR to talk to each direct neighbor is obtained by `MeshDV-CL` through a particular system call. The cost that can be associated to a link in terms of transmission rate R is the inverse of the rate itself, consequently, routes are chosen by minimizing the following cost function:

$$C_{\mathrm{CL}}(\mathrm{Path}) = \max_{\forall (i,j) \in \mathrm{Path}} \left[\frac{1}{R_{i,j}} \right], \tag{2}$$

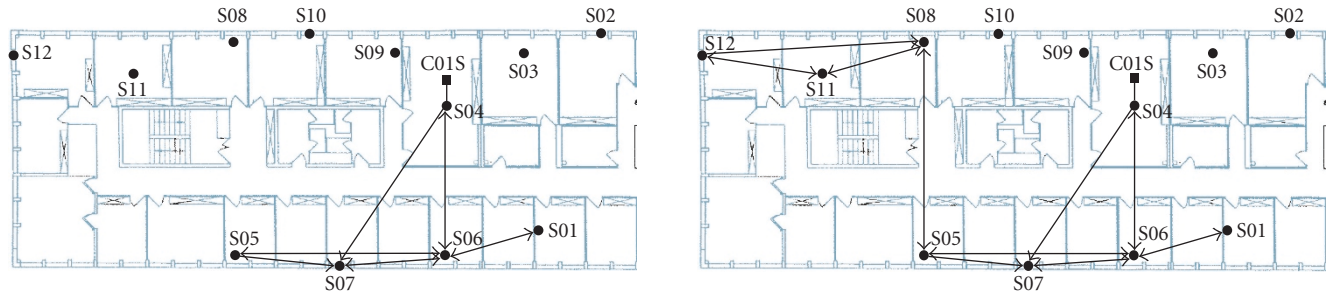
where $R_{i,j}$ is the raw data transmission rate on the link from i to j and Path is the set of all links from the source to the destination. The composition rule of the rate metric (i.e., the max function) is designed to avoid paths having bottleneck links that offer low transmission rate. Links with low data rate increase congestion, interference, and delay. Furthermore, as our measure proves, links with low transmission rates are usually more error-prone. Traditional layered approach relegates rate adaptation mechanism to the MAC layer, without any interaction with the routing layer. This layered design choice, while elegant from an architectural perspective, leads to under-exploit the wireless medium. Instead, rate-aware routing that chooses links offering high transmission rates is able to increase the average throughput, as we showed by simulation in [14] and confirmed by real measurements in the present work.

This first cross-layer version of `MeshDV` lets us make the comparison between hop-count and rate-aware metrics. The results of measurements, which we present in the next section, show interesting improvements that prove the importance of having cross-layer metrics.

4. MEASUREMENTS

The `MeshDVNet` test bed deployed in the LIP6 laboratory offers IPv6 connectivity on all the building floors and can be used without any particular setup on PCs or laptops running any modern operating system.[2] Usual Internet usage including basic services like web browsing, e-mailing, ssh, and so

[2] Actually, WindowsXP is not able to make DNS queries using IPv6 packets, but only using IPv4 packets, thus we developed a small software translating DNS queries from IPv4 to IPv6.



(a) Topology T1.

(b) Topology T2.

Figure 5: Snapshot of the two main topologies used during the tests.

on can be performed without any particular problem when using `MeshDV-CL`.

Compared to traditional WLANs, with wired LAN behind each AP, some performance deterioration can be observed while performing heavy data transfers (massive downloads, FTP connections). This deterioration is due to the different performance of the multihop mesh backbone, compared to a wired LAN. In the following paragraphs, we describe and discuss the measurements we performed in order to evaluate the behavior of the deployed mesh backbone.

In a previous work [14] we showed by simulations that rate-aware routing outperforms hop-count routing in terms of traffic volume and is even able to reduce the end-to-end delay. Here we present the results obtained in a real environment, which partly confirm our previous findings. Indeed, as we show in the following, while real measurements confirm that the rate-aware metric increases the throughput, in contradiction to simulations, the delay is slightly increased as well. In contrast with simulation, performing real measurements has the drawback of being not repeatable. Even repeating them at the same time of day, with exactly the same setup, leads to different results. There is no way to avoid such a behavior, due to the intrinsic characteristics of the wireless medium. Nevertheless, we repeated all measurements several times, in order to ensure that what we present hereafter is the general behavior of the system and that numeric results are always in the same order of magnitude at each run. Therefore, for each set of tests we present a single measurement that is representative for the general behavior of the test bed. The purpose of our tests was to compare the cross-layer rate-aware metric versus the hop-count metric. In such context we do not focus on the routing protocol or algorithm, thus comparisons with protocols other than MeshDV, like DSR or AODV, are out of scope.

The presented performances are not absolute, they depend not only on the environment, but also on the hardware and software we used. Thus changing one of those parameters will change the results, in terms of achievable throughput or delay. Nevertheless, we argue that a cross-layer routing approach using raw transmission rate as a metric will always behave better than the one using the simple hop-count metric.

4.1. Tests setup

In order to highlight some *pathological* behaviors, we did not use all the `MeshDVBoxes` of our test bed when we performed our measurements. We reduced the number of running nodes to two different topologies.

T1: this topology is composed of four `MeshDVBoxes` shown at the bottom of Figure 3, namely, S05, S07, S06, and S01. A snapshot of the obtained topology is shown in Figure 5(a).

T2: this topology is composed of the same four nodes as in T1, and the nodes placed on the left of S05 in Figure 3, namely, S08, S11, and S12. A snapshot of the obtained topology is shown in Figure 5(b).

These two topologies allow to make different tests between nodes separated by 1 up to 6 hops, putting in evidence some interesting performance. Measurements concerned the delay, more specifically the round trip time (RTT), and the TCP throughput. We chose to perform measurements using TCP traffic instead of UDP traffic because it gives a better idea of the kind of connection available. Indeed, most of the traffic in the Internet is TCP, furthermore, the CBR traffic usually used for UDP does not model at all the reality. The choice of measuring the RTT is consequent, since it can be interpreted like the delay to send a TCP segment and receive the corresponding ACK.

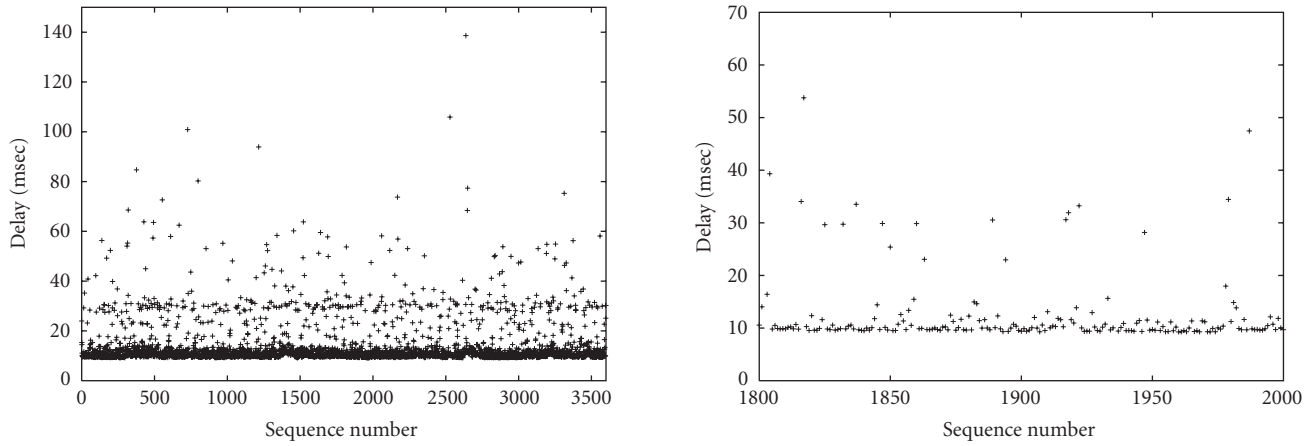
4.2. Delay measurements

The first kind of tests concerned the delay experienced by traffic when traversing the mesh backbone. For this purpose we used the `ping6` utility that allows measuring the RTT between two nodes. We performed this measure between S05 and S01 with the first topology T1. Table 1 shows the detailed command line parameters applied to ping6 for this test.

In the first set of tests, we turned off all nodes except those of T1 and we used `MeshDV-CL` as routing protocol. In Figure 6 there are the results obtained for 3600 packets (around one-hour measurement). Figure 6(a) shows the RTT of all packets, ordered by sequence number. Figure 6(b)

Table 1: Commands used to start tests.

3600 requests IPv6 ping toward S01, using packets of 1100 bytes: `ping6 -n -s 1100 -c 3600 soekris-01.infradio-jussieu.lip6.fr`
3600 seconds (1 hour) IPv6 TCP flow toward S01 using packets of 1100 bytes and reporting results each 20 seconds: `iperf -V -c soekris-01.infradio-jussieu.lip6.fr -l 1100 -t 3600 -i 20`



(a) The complete test.

(b) Zoom on packets with a sequence number between 1800 and 2000.

Figure 6: RTT measurements using topology T1 with `MeshDV-CL` and all other nodes turned off.

presents a zoom on the interval concerning sequence numbers ranging from 1800 to 2000, that is, in the middle of the test. Both figures highlight the regular RTT value (around 13.5 msec) and the low loss ratio on a 3-hop topology using `MeshDV-CL`. The first line of Table 2 summarizes the numeric output.

In the second set of tests, we turned on all the remaining nodes of the test bed and restarted the 3600 packets RTT measurement between the same nodes, still using `MeshDV-CL`. Results of this second test are depicted in Figure 7. With this setup nodes that are not part of T1 do not generate data traffic; they only generate and exchange some `MeshDV-CL` routing packets. Even if they do not participate actively to the measurement, the nodes now present in the network, even with their small amount of routing exchange, have an impact on the RTT between S05 and S01. Indeed, the average RTT has an increase of 50% and is more unstable. This can be clearly seen comparing Figure 6(a) to Figure 7(a). Instead by comparing Figure 6(b) to Figure 7(b) one can see how the packet loss has increased. The third line of Table 2 summarizes the numeric output.

Our third set of tests is similar to the first one, but using `MeshDV-HC`, that is, without cross-layer routing. Results are depicted in Figure 8. The first time we performed this test, we were expecting that `MeshDV-HC` would perform worse compared to `MeshDV-CL`. However, we were not expecting such a large gap. The RTT values remain almost unchanged or even reduced compared to the first test, but the packet loss ratio rises to more than 80%. Numeric results are summarized in the second line of Table 2. A deeper analysis reveals that when using `MeshDV-CL` the route between S05 and S01 goes through both S07 and S06. In the case of `MeshDV-HC` instead, S05 and S06 see each other directly, but unreliably, thus while reducing the number of hops, the transmission rate is decreased (since the hop is longer) and lots of packets get lost on this long hop. This also explains why the RTT experienced may be slightly lower. Indeed, when a packet and its corresponding reply do not get dropped, they go only through 2 hops (not 3 hops as in the case of `MeshDV-CL`). There is no use to show the performance on T1 with `MeshDV-HC` when all other nodes are turned on, since performance gets worse with a higher packet loss ratio.

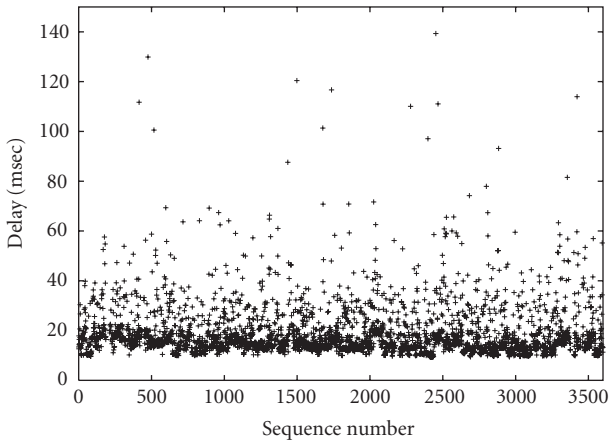
Beside the fact that the above tests give an idea of the RTT experienced on a 3-hop mesh backbone, they also show how, by using a simple cross-layer approach, based on the raw transmission rate, it is possible to improve performance in terms of reliability. In the next section we will show the impact of the cross-layer solution on the TCP throughput.

4.3. Throughput measurements

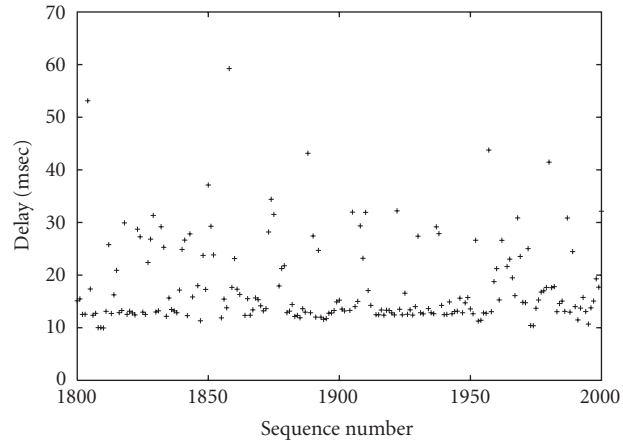
The second kind of tests performed concerned the throughput experienced by TCP connections on the mesh backbone. For this purpose, we used the Iperf utility [15] which by a

Table 2: Delay measurements.

Cross-layer	Number of nodes	Minimum	Average	Maximum	Standard deviation	Packet loss
Yes	4 nodes only	9.226	13.611	138.669	9.953	5.8%
No	4 nodes only	9.234	12.382	68.949	6.302	86.4%
Yes	All nodes	9.292	19.848	1020.911	22.196	14.7%



(a) The complete test.



(b) Zoom on packets with a sequence number between 1800 and 2000.

Figure 7: RTT measurements using topology T1 with MeshDV-CL and all other nodes turned on.

simple client/server architecture allows measuring the TCP throughput between two nodes. We installed the Iperf server on S01 MeshDVBox, while the client was installed on several other MeshDVBoxes, in order to perform different tests. The second line of Table 1 shows the detailed command line parameters applied to Iperf for this test.

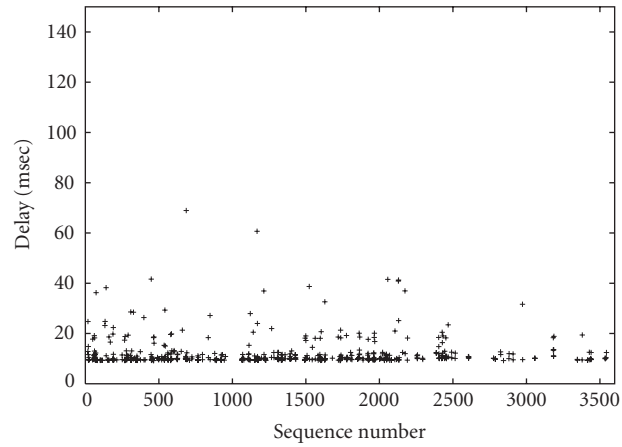
The first test was performed using the topology T1 and a single TCP flow between S05 and S01. This test is the counterpart of the first RTT test. Indeed, all nodes were turned off except S05, S07, S06, and S01 on which MeshDV-CL was running in order to obtain the same conditions. Figure 9 shows the results of the one-hour test, with throughput measurements repeated every 20 seconds. The average throughput is around 1 Mbit/s and obviously it is highly variable due to the wireless environment. The same test performed using MeshDV-HC leads to an average throughput lower than 100 Kbit/s, more than 10 times smaller, which is not worth to be shown here. This first set of tests already shows that even on a 3-hop topology cross-layer routing offers good results in terms of throughput. The big amount of packet loss when using MeshDV-HC prevents TCP from increasing the congestion window through the slow start mechanism.

The second set of tests was performed on the same topology (T1), but starting 3 different TCP flows with a 20-minute interval between each one. The first flow was started at time 0 between S06 and S01 and triggered to last 1 hour. The second flow, between S07 and S01, was started after 20 minutes (time 0 + 1200 seconds). The third flow, between S05 and S01, was started 20 minutes after the second one (time 0 + 2400 seconds). Results for both MeshDV-CL and MeshDV-HC are shown in Figure 10. In both cases, flows between S07→S01 and S06→S01 behave the same, since there is no route change whether MeshDV-CL or MeshDV-HC is used. The third flow instead performs differently. In the case of MeshDV-CL it achieves a throughput of 1 Mbit/s when both the other two flows stop, while in the case of MeshDV-HC it performs very poorly. When the third flow remains the only one, we fall in the same situation as in the previous throughput test (single flow case), obtaining indeed the same results for both MeshDV-CL and MeshDV-HC.

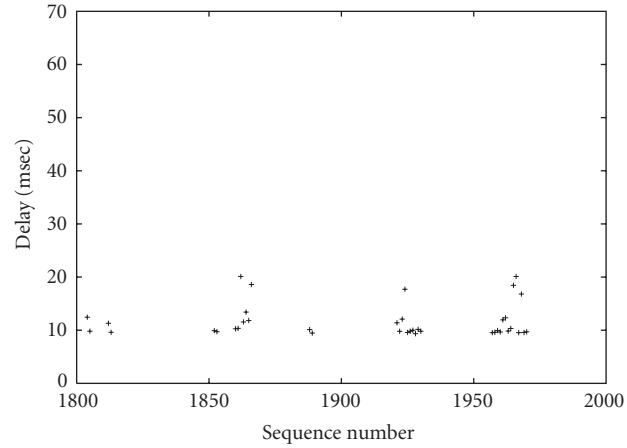
The fact that the third flow is not able to send traffic in the presence of the other two flows is not a new result, but a known issue. Several works, which can be found in the literature, deal with this kind of unfairness.

For completeness we report in Figure 11 the result when running the same test using MeshDV-CL, but with inversed starting order for the three TCP flows. The result does not change, proving that it is not a matter of start order.

The last kind of tests we performed concerned the throughput on the topology T2. The purpose of this test was to look at the throughput performance when the TCP flow traverses more than 4 nodes (3 hops). To do this, we set up a flow from S12 to S01, thus passing up to 6 hops, depending on MeshDV version used. In particular, using MeshDV-HC, the



(a) The complete test.



(b) Zoom on packets with a sequence number between 1800 and 2000.

Figure 8: RTT measurements using topology T1 with `MeshDV-HC` and all other nodes turned off.

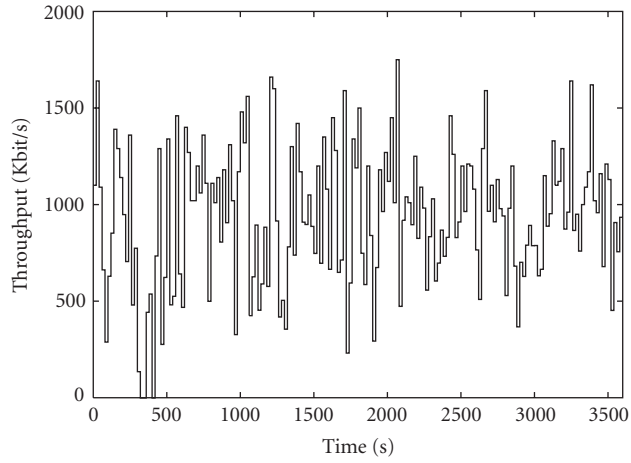


Figure 9: Iperf measurement of a single TCP flow between S05 and S01 on the T1 topology using `MeshDV-CL` and all other nodes turned off.

number of hops was continuously fluctuating between 4 and 6. This behavior was due to the presence of long unreliable hops that, from time to time, were considered as broken, thus changing the route using shorter links and increasing the number of hops. This continuous fluctuation led to a throughput that was almost always zero with only few packets getting through. In the case of `MeshDV-CL`, the path was stable on 6 hops, using shorter links offering higher raw transmission rates. The throughput obtained has an average of almost 600 Kbit/s and is shown in Figure 12.

This last test highlights even more the performance improvements that a cross-layer routing may induce to a mesh backbone. Indeed, even if 600 Kbit/s does not seem to be an impressive performance, compared to the almost zero traffic of the hop-count approach, it assumes a nonnegligible value. Furthermore, in a more general context, 600 Kbit/s traffic on a 6-hop topology is anyway a good result.

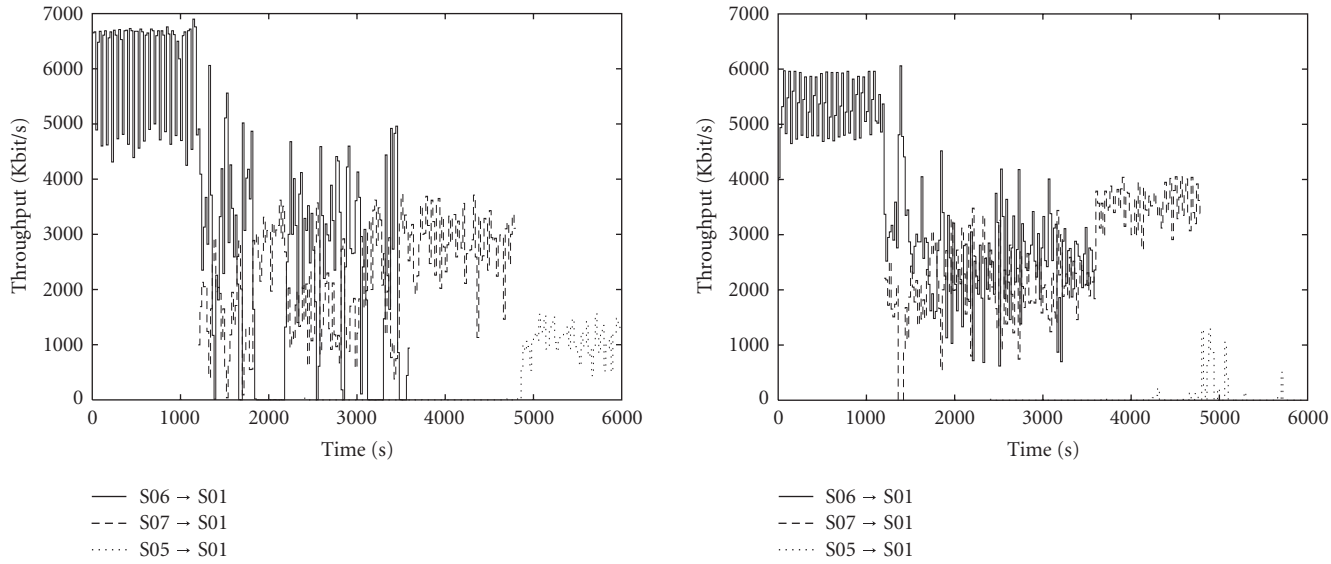
5. RELATED WORKS

In last years, several test beds for wireless ad hoc and mesh networks have been built in various research labs. Some typical examples are MIT Roofnet [16], Carleton University [17], Politecnico di Milano MobiMESH [18], APE test bed in Uppsala University [17], Microsoft Research [19], and UCSB MeshNet [20]. Among these projects, only the MIT, Politecnico di Milano, and Carleton proposals are able to offer connectivity to wireless clients without the necessity for the latter to embed any kind of software.

In particular for MIT, their test bed is IPv4-only and uses NAT [21] on each router in order to hide any client associated to it. Such a solution is not robust at all to mobility and is limited by issues related to the NAT protocol. This project uses a particular metric, the ETX metric, which estimates the reliability of used links. This solution is not a cross-layer approach, since it is based on beacons sent from the network layer.

The Carleton University Project uses IPv6 in its test bed like in our case, however, the routing protocol used is OLSR with the simple hop-count metric. The approach proposed by Carleton University is very similar to the one proposed by Politecnico di Milano. The main differences seem to be the IP version (IPv4 for Politecnico di Milano) and the hardware used.

The others projects are basically ad hoc test beds using hop-count-based routing protocols, usually AODV [22] or OLSR [23], and not implementing any cross-layer metric. Only the Microsoft proposal uses a cross-layer metric, since it implements a variation of the ETX metric which takes into account the raw transmission rate like in our solution.



(a) Throughput when using `MeshDV-CL`.

(b) Throughput when using `MeshDV-HC`.

FIGURE 10: Iperf measurements using 3 TCP flows started with a 20-minute interval.

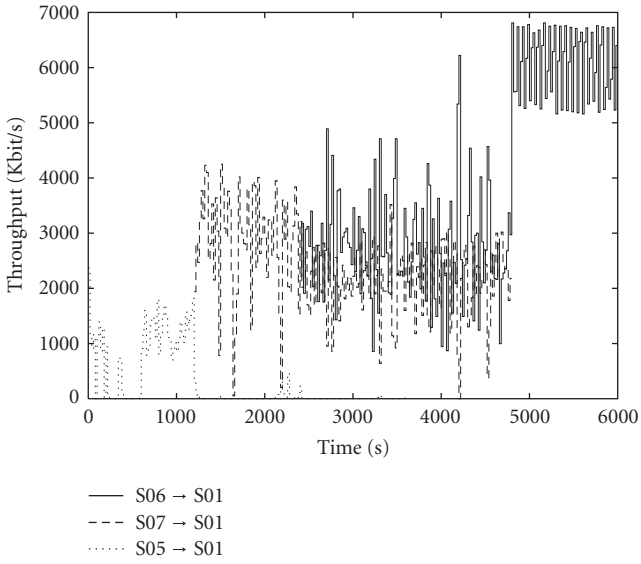


FIGURE 11: Iperf measurements using 3 TCP flows started with a 20-minute interval and inverting the starting order.

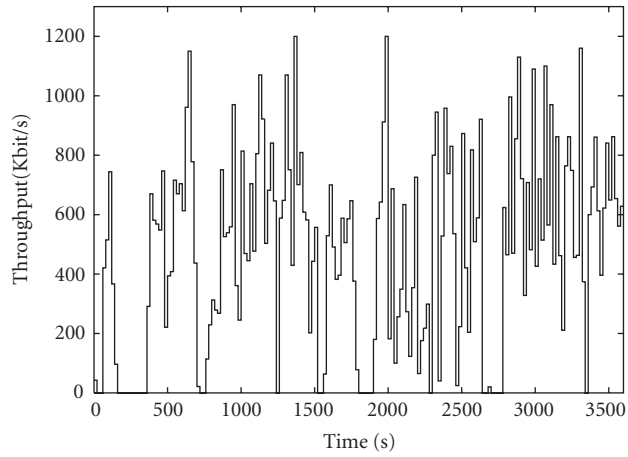


FIGURE 12: Iperf measurement of a single TCP flow on the T2 topology using `MeshDV-CL`.

Nevertheless, the proposed metric is a complex cost function that cannot be compared directly to the simple transmission rate metric.

Rate-aware metric has been studied by simulation also by Seok et al. [24], however, compared to our solution they have a totally different approach. Indeed, they first compute the hop-count shortest path and then they try to split long hops with low transmission rate in shorter hops with higher transmission rate. Such two-phase solution leads to suboptimal solutions since the starting point is always the not optimal route obtained with the hop-count metric. Nevertheless, the results are similar to what we proposed in [14], that is, an increased traffic volume and a reduction in the delay. The fact that both simulation studies observe a reduction of the delay, which is not observed in real measurements, leads us to suppose that in NS-2, the simulator used in both cases, the packet latency in a forwarding operation is not correctly modeled.

6. CONCLUSION

In this paper we showed by measurements made on a real test bed that cross-layer routing is able to increase the robustness and performances of a mesh backbone.

Despite the amount of research done on routing in wireless multihop networks, showing that the hop-count metric behaves poorly, and despite the myriads of works proposing new cross-layer routing approaches, few of them have known a real implementation allowing realistic feedback. We implemented a simple cross-layer routing approach (`MeshDV-CL`) and compared its performances to a more traditional hop-count routing approach (`MeshDV-HC`).

The tests performed on the MeshDVNet test bed, deployed in the LIP6 laboratory show clearly that cross-layer routing increases the robustness of the mesh backbone by choosing shorter links that are more reliable. Furthermore, these shorter links also offer higher raw transmission rate, increasing the transport capacity of the backbone.

A representative result we obtained shows how a TCP flow of 600 Kbit/s can be maintained when using `MeshDV-CL` on a 6-hop topology, where the throughput using `MeshDV-HC` (the traditional hop-count metric) is almost zero. This result is not absolute, it depends on the hardware, software, routing protocol we used. Nevertheless even changing one of those parameters using the raw transmission rate as a metric will always give better performances than using the simple hop-count metric.

ACKNOWLEDGMENT

This work was supported in part by the European Commission project WIP under contract 27402.

REFERENCES

[1] L. Yang, P. Zerfos, and E. Sadot, "Architecture taxonomy for control and provisioning of wireless access points (capwap)," RFC 4118, June 2005.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

[3] S. Deering and R. Hinden, "Internet protocol version 6 (ipv6) specification," RFC 2460, December 1998.

[4] Soekris Engineering, http://www.soekris.com/net4521.htm.

[5] Institute of Electrical Electronics Engineers (IEEE), "Supplement to 802.11-1999, wireless lan mac and phy specifications: higher speed physical layer (phy) extension in the 2.4 ghz band," IEEE Standard 802.11, 1999.

[6] Proxim wireless, http://www.proxim.com/products/wifi/client/11bgpccard/.

[7] Institute of Electrical and Electronics Engineers (IEEE), "802 standard working group, wireless lan medium access control (mac) and physical layer (phy) specifications: high-speed physical layer in the 5 ghz band," IEEE 802.11a Standard, 1999.

[8] Netgate, http://www.netgate.com.

[9] The NetBSD Project, http://www.netbsd.org.

[10] L. Iannone and S. Fdida, "MeshDV: a distance vector mobility-tolerant routing protocol for wireless mesh networks," in *Proceedings of the 1st IEEE ICPS Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality (REALMAN '05)*, Santorini, Greece, July 2005.

[11] A. Conta and S. Deering, "Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification," RFC 2463, December 1998.

[12] C. Perkins and P. Bhagwat, "Highly dynamic desination-sequenced distance-vector (dsdv) routing for mobile computers," in *Proceedings of the ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pp. 234–244, London, UK, August-September 1994.

[13] L. Iannone and S. Fdida, "Mrs: a simple cross-layer heuristic to improve throughput capacity in wireless mesh networks," in *Proceedings of the 1st International Conference on Emerging Networking Experiments and Technologies (CoNEXT '05)*, pp. 21–30, Toulouse, France, October 2005.

[14] L. Iannone and S. Fdida, "Can multi-rate radios reduce end-to-end delay in mesh networks? a simulation case study," in *MESH NETWORKING: Realizing the Wireless Internet (Mesh-nets '05)*, Budapest, Hungary, July 2005.

[15] DAST/NLANR Iperf Project, http://dast.nlanr.net/Projects/Iperf/.

[16] Mit roofnet project, http://pdos.csail.mit.edu/roofnet/doku.php.

[17] Wireless mesh networking at carleton university, http://kunz-pc.sce.carleton.ca/MESH/index.htm.

[18] Politecnico di Milano MobiMESH, http://www.elet.polimi.it/upload/antlab.

[19] Mesh Networking Academic Resource Toolkit 2005, http://research.microsoft.com/netres/kit/.

[20] Ucsb meshnet, http://moment.cs.ucsb.edu/meshnet/.

[21] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, January 2001.

[22] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on demand distance vector (aodv) routing," RFC 3561, July 2003.

[23] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," RFC 3626, October 2003.

[24] Y. Seok, J. Park, and Y. Choi, "Multi-rate aware routing protocol for mobile ad hoc networks," in *Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference (VTC '03)*, vol. 3, pp. 1749–1752, Jeju, Korea, April 2003.