TCP Performance Evaluation over Backpressure-based Routing Strategies for Wireless Mesh Backhaul in LTE Networks☆

Natale Patriciello[a], José Núñez-Martínez[b], Jorge Baranda[b], Maurizio Casoni[a], Josep Mangues-Bafalluy[b]

[a] *Department of Engineering Enzo Ferrari*
University of Modena and Reggio Emilia
via Vignolese 905, 41125 Modena, Italy.
[b] *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)*
Av. Carl Friedrich Gauss 7,
08860 Castelldefels, Barcelona, Spain

Abstract

Wireless redundant networks are expected to play a fundamental role to backhaul dense LTE networks. In these scenarios, backpressure-based routing strategies such as *BP-MR* can exploit the network redundancy. In this paper, we perform an exhaustive performance evaluation of different TCP variants over an LTE access network, backhauled by various routing protocols (including *per-packet* and *per-flow BP-MR* variants and a static alternative, *OLSR*) over two different wireless topologies: a regular mesh and an irregular ring-tree topology. We compare the performance of different TCP congestion control algorithms based on loss (NewReno, Cubic, Highspeed, Westwood, Hybla, and Scalable) and delay (Vegas) under different workloads. Our extensive analysis with ns-3 on throughput, fairness, scalability and latency reveals that the underlying backhaul routing scheme seems irrelevant for delay-based TCPs, whereas *per-flow* variant offers the best performance irrespective of any loss-based TCP

☆This work was partially funded by the EC under grant agreement no 645047 (H2020 SANSA project) and by the Spanish Ministry of Economy and Competitiveness under grant TEC2014-60491-R (5GNORM).

Email addresses: natale.patriciello@unimore.it (Natale Patriciello), jose.nunez@cttc.cat (José Núñez-Martínez), jorge.baranda@cttc.cat (Jorge Baranda), maurizio.casoni@unimore.it (Maurizio Casoni), josep.mangues@cttc.cat (Josep Mangues-Bafalluy)

congestion control, the most used in the current Internet. We show that *BP-MR per-flow* highly reduces the download finish time, if compared with *OLSR* and *BP-MR per-packet*, despite showing higher round-trip-time.
Keywords: TCP, Backpressure, ns-3, Routing, LTE

1. Introduction

Wireless mesh backhauls are expected to play a fundamental role in providing transport resources closer to the edge and the capillarity required by next generation mobile networks. In particular, for accommodating the rising demand for wireless connectivity, future Long Term Evolution (LTE) dense deployments formed by Small Cells (SC) are expected to be deployed, because reducing cell size to increase frequency re-use is the most simple and effective way to increase raw capacity [1]. Therefore, due to fiber unavailability in all the deployment scenarios, backhauling these dense LTE deployments can be addressed with wireless (e.g., mmWave) transport nodes associated to SCs. Each of the wireless transport nodes will have multiple backhaul interfaces, with the possibility to create a backhaul network through a redundant wireless mesh. In this way, LTE user and control plane traffic will be appropriately carried from/towards the Evolved Packet Core (EPC), with major benefits regarding cost, ease of deployment, coverage, and capacity.

The path redundancy of these potentially large wireless mesh networks should be exploited by the backhaul routing protocol since it determines the way data is transported to/from the UEs. In literature, we have congestion-agnostic strategies such as Multiprotocol Label Switching (MPLS, RFC 5921) and Optimized Link State Routing (OLSR, RFC 3626). There are also congestion-aware strategies, such as backpressure-based ones [2], which can exploit network redundancy because they dynamically map the trajectory followed by each packet to the less utilized path. However, these decisions may potentially make the path followed by consecutive packets of the same flow disjoint.

Today, one of the most appropriate protocols to transport traffic is TCP,

thanks also to the increasing popularity of on-line streaming services (e.g., YouTube, Netflix, and Spotify). Many TCP variants have been proposed over the years, with the declared objective of fixing inefficiencies of the standard version over peculiar environments (e.g., high bandwidth-delay product, different access channels, ...). While TCP performance over Radio Access Network (RAN) such as LTE, has been analyzed in the past, a fundamental question is whether backhaul routing protocols can also appropriately serve TCP traffic, and moreover if they penalize or reward different TCP congestion control algorithms in the mentioned wireless mesh backhaul deployments. The challenge for backhaul routing protocols is clearly to exploit the network redundancy without affecting the TCP performance.

The contribution of this paper is threefold: (1) to investigate the advantages and limitations of different backhaul routing strategies under a wireless mesh backhauls carrying LTE traffic; (2) to analyze the particular response of different TCP congestion control algorithms over these backhaul routing protocols; and (3) to infer the proper combination of TCP congestion control and underlying backhaul routing protocol within the aforementioned context.

Regarding backhaul routing protocols we compare well-known single-path *OLSR*, which in the absence of node mobility and failures is equivalent to MPLS for the purpose of the paper, with routing variants based on backpressure routing concept. In this sense, our starting point is Backpressure Multi-Radio (BP-MR) [3], which operates on a *per-packet* basis. To reduce the reordering at destination, we also employ a *per-flow* variant of BP-MR, proposed in [4] but tested only in an emergency scenario with a backhaul satellite link. We evaluate the response of these three backhaul routing variants using TCP Cubic between the end-points of the communication.

Afterwards, the paper broadens the analysis of the mentioned backhaul routing strategies considering the impact of installing different loss- and delay-based TCP congestion control algorithms (i.e. New Reno, Highspeed, Westwood, Hybla, Vegas, and Scalable) in the UEs and the remote server. Last but not least, we analyze in detail the different response of delay- and loss-based TCP con-

gestion control algorithms over the different routing strategies in the presence of an increasing UDP backhaul workload.

The conducted experiments with ns-3[1] on two different topologies analyzing throughput, fairness, scalability, and latency allow concluding that delay-based TCP variants, such as Vegas, are unable to stress the backhaul resources achieving similar performance over the three analyzed routing strategies, but they are outperformed by loss-based TCP variants, such as Cubic. However, these loss-based congestion control algorithms pose difficulties to routing protocols due to their aggressiveness in sending data. The use of different routing algorithms is then reflected by differences in TCP performance, irrespective of the TCP congestion control algorithm used. Specifically, using *OLSR* as the reference protocol, *BP-MR per-packet* exhibits lower fairness among flows and lower throughput due to required packet reordering at the destination. On the other hand, *BP-MR per-flow* offers the best TCP performance by reducing the download finish time despite its higher round-trip-time. The performance improvement is due to fewer losses and reordering events experienced while showing capabilities to circumvent congestion on a *per-flow* basis.

The remainder of this paper is organized as follows. Section 2 contains the necessary background on *BP-MR* (both *per-packet* and *per-flow*) and TCP protocol. Section 3 describes the methodology and the results gathered over a regular mesh backhaul topology. In Section 4 we present the results collected on an irregular topology, taken from a real eNodeB deployment in the city of Modena, Italy. Section 5 covers related work, and finally, Section 6 concludes the paper.

[1] https://www.nsnam.org

2. BP-MR and TCP Background

2.1. BP-MR (per-packet and per-flow)

Basically, if we define as *backlog* the queue size at nodes, the main idea of backpressure is to give priority to links and paths that have higher differential *backlog* between neighboring nodes, i.e. routing traffic by minimizing the sum of the queue *backlog*s in the network among time slots. The original *BP-MR per-packet* variant, proposed in [3], takes (for each packet) decentralized routing decisions in a two-stage process: firstly, it classifies the packets accordingly to their destination, and then employs queue informations (gathered from neighbors) to compute the best possible next-hop.

However, this per-packet strategy leads to a high degree of packet reordering at the destination node, as demonstrated in [4]. To overcome the packet reordering problem without losing intrinsic characteristic of *BP-MR* and the capability of circumventing congested paths, in that paper we applied the concept of *per-flow* path selection strategy (used in many fields, such as in Data Center Networks [5]) and proposed a new *per-flow* variant, testing it in an emergency scenario. The *per-flow* variant computes the best next hop for the first packet of a flow, and then remember the decision in a forwarding table. In this way, a new flow has the flexibility to route dynamically to any of the available paths, and so is able to circumvent congested routes, without causing packet reordering at the destination.

2.1.1. BP-MR per-burst

Note that in this paper we evaluate *BP-MR per-flow* and *BP-MR per-packet* considering (i) short-lived flows and (ii) null network dynamics during the duration of such short-lived flows. However, in the case of long-lived flows and/or high dynamics the path selection granularity detailed by *BP-MR per-flow* can be inefficient. A solution could be to increase the frequency of weight computations for long-lived traffic flows. In this way, the path could potentially change for long-lived flows on a per-burst basis. An open question here is how

to define the weight computation frequency. The weight computation frequency could operate pro-actively by periodically recomputing the weights within the lifetime of a long-lived flow, reactively based on abrupt changes in congestion conditions or using some combination of the previous ones (hybrid). Such an evaluation is, however, out of the scope of this paper.

2.2. TCP and Congestion Control Algorithms

Since TCP is a connection-oriented protocol, it tries to setup a connection with the other peer before any data exchange, and at the end of the transmission, it peacefully closes it. The objective is to deliver an ordered stream of bytes between two end points. That flow is bi-directional (data can be exchanged both ways), and is governed by a state machine outlined in RFC 793.

Since the data stream must be delivered in order to the application, without holes, if some segments are lost they must be retransmitted. End hosts exchange control packets (that can be piggy-backed with application data) between themselves, to regulate the data flow: the main is the acknowledgment (ACK), which acknowledge one (or more) successful receipt(s). There are different strategies to sense a packet loss, through timer expiration (Retransmission TimeOut, RTO) or by inferring such loss by receiving multiple duplicated ACK segments (Fast Retransmission).

At this point, to characterize the network data load in the presence of TCP flows, it is worth to note that the amount of data injected into the network is not following a constant-rate pattern, but is instead defined by an internal state variable referred to as congestion window ($cWnd$) that changes over time. It is decreased after each congestion event, and it is increased, depending on the connection history, by either slow start or congestion avoidance phases. The slow start phase is commonly used unaltered by all TCP implementations (please note that, contrarily to its name, the $cWnd$ growth in this phase is exponential but depends on the connection RTT). Congestion avoidance algorithm enters the game when the $cWnd$ exceeds the slow start threshold; historically, the Reno and Tahoe algorithms were the first deployed. Right now, the algorithm

on the Standard Track (RFC 5681 and RFC 6582) is New Reno (loss-based), but many variants have been proposed to overcome general and environment-specific problems and performance issues, and some of them even define the *cWnd* reduction strategy after a congestion event. The TCP congestion avoidance variants can be divided following the main parameter that signal congestion: *loss-based* variants use the packet loss information (guessed through duplicated ACK or timeout) as the primary congestion signal, while *delay-based* variants use the queuing delay as the primary congestion signal, increasing window size if delay is small and decreasing it if delay is large.

For sake of completeness, we report in the following the main characteristics of the loss-based congestion control algorithms used in this paper:

- *New Reno* is the evolution of Reno algorithm, where the window growth depends on the RTT;
- *Cubic* is an algorithm for high-speed network environment. The window growth function is governed by a cubic function in terms of the elapsed time since the last loss event, and thus is independent from the RTT;
- *HighSpeed* is designed for TCP connections with large congestion windows, and its algorithm kicks in when the *cWnd* increases beyond a predefined threshold, allowing faster growths and accelerating the recovery from losses;
- *Westwood* is based on New Reno but uses the AIAD (Additive Increase/Adaptive Decrease) approach: when a congestion episode happens, instead of halving the *cWnd*, it tries to estimate the network bandwidth and use the estimated value to adjust the cWnd;
- *Hybla* is born for satellite connections, and the key idea is to obtain, for these connections, the same instantaneous transmission rate of a low-RTT reference connection;
- *Scalable* improves the bandwidth utilization by employing a more aggressive adjustment algorithm, with respect to New Reno.

For the delay-based family, we employed *Vegas*, a pure delay-based congestion control algorithm. It implements a proactive scheme which tries to prevent packet drops by maintaining a small backlog at the bottleneck queue.

3. Analyzing regular backhaul topologies

3.1. Methodology

We modeled a 10 MB file transfer from a remote server outside the Mobile Network to the UEs, which represents the download of a data chunk from a remote streaming service. In all the simulations, there is always one transfer per UE attached to the Mobile Network. Therefore the number of TCP file transfers is equivalent to the number of UEs. To complete the picture, the UEs are uniformly distributed over the eNodeBs.

We conducted experiments for different TCP variants; we compared loss-based flavors TCP Cubic, NewReno, HighSpeed, Westwood, and Hybla, with Vegas, a TCP flavor based on delay to calculate congestion window. Furthermore, for each TCP flavor, we compared the performance of different underlying backhaul routing protocols: single-path based on *OLSR*, and backpressure *per-packet* and *per-flow* based on *BP-MR*. All the simulations are conducted with ns-3 using latest version of LTE model[2], *BP-MR* routing protocol implemented in [3], and the different TCP variants presented in [6].

We measure two parameters to quantify the performance of the TCP connections: the download finish time and the average RTT. The download finish time quantifies the average throughput of the end-to-end connection (i.e., from the remote server to the UEs). The RTT abstracts network latency and is defined as the time taken by a TCP segment to reach the UE plus the time taken by the server to receive the correspondent ACK from the UE. This parameter is gathered from the TCP data sender each time it receives a non-duplicated ACK segment: therefore, retransmitted or out-of-order data packets are not taken into

[2] http://networks.cttc.es/mobile-networks/software-tools/lena/

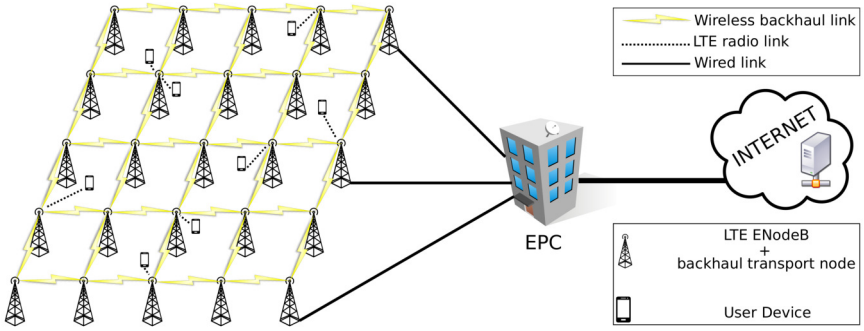

Figure 1: Full mesh topology (UE number can vary).

account in the RTT measurement. Thus, the RTT should be weighted with the download finish time to properly characterize the performance experienced by TCP in combination with each of the routing protocols under evaluation. The combinations of download finish time and RTT will give insights on throughput, fairness, and latency.

The reported values of download time and RTT are represented in candlesticks, where the boxes stretch from the 20th to the 80th percentiles, and the whiskers represent the maximum and minimum values, with the average value represented by a black horizontal line. In the following, we analyze the routing protocol impact, the TCP congestion control impact, and finally the different response of the routing protocols to loss-based and delay-based TCP.

Our reference scenario depicted in Figure 1 is a dense SC network deployment covering $2Km^2$ with an inter-SC distance of two hundred meters. Note that every SC is composed of an LTE eNodeB and a wireless transport node. In particular, twenty-five LTE eNodeBs are deployed with peak downlink throughput of 350 Mb/s that corresponds to an areal capacity of $8.8\frac{Gbps}{Km^2}$ [7]. The wireless transport node endows several 500 Mb/s point-to-point (PTP) interfaces that form wireless links of 4 ms of propagation delay and are connected among them to form a plain grid, as illustrated in Figure 1. The mesh is in turn connected to the LTE Evolved Packet Core (EPC) through three PTP wired links, with 1 Gb/s of available bandwidth and 0.5 ms of propagation delay. The

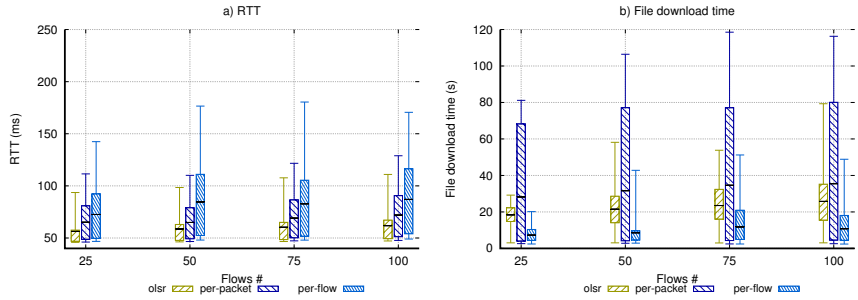

Figure 2: RTT and file download time using TCP Cubic, while increasing the number of LTE UEs downloading a 10 MB file.

EPC is connected to the Internet through another PTP wired link (characterized by a bandwidth of 10 Gb/s and a propagation delay of 5 ms); the queue sizes of the backhaul network are set to the 100 % of each link bandwidth-delay product, making an effort to reduce default buffer size, as suggested in [8]. Regarding the Radio Access Network (RAN), we used European frequencies and a Okumura-Hata propagation model [9]: the LTE connection between UE and the eNodeB is modeled inside an urban environment of a medium city.

3.2. Backhaul Routing Protocol Impact

Here we analyze the backhaul routing protocols response when using TCP Cubic (the default congestion control algorithm of Linux, and hence of Android-based devices and the majority of servers on the Internet) as default transport protocol for the transfers. Briefly, we remind that *OLSR* always chooses the shortest path in terms of hops, *BP-MR per-packet* evaluates for each packet the best trade-off between proximity and congestion, while *BP-MR per-flow* performs the same proximity-congestion evaluation but for each flow.

Throughput. Figure 2 shows the TCP download time performance over four different cases, namely with 25, 50, 75, and 100 UEs concurrently downloading a remote file. Results reveal that by using *BP-MR per-flow* the flows experience the smallest file download times. Moreover, *BP-MR per-packet* experiences a significant throughput degradation because it has the higher times

of the entire set of protocols, whereas *OLSR* is between the two. The bigger size of the boxplots in per-packet indicates a frequent use of unconstrained path diversity, causing excessive reordering at UEs. The number of potential trajectories that a packet can take with *BP-MR per-packet* is unconstrained since (i) independent decisions are taken in every hop and (ii) routing decisions are independent for consecutive segments in each hop. Maximum values obtained with *per-flow* are around the average obtained with *BP-MR per-packet* indicating a frequent use of redundant paths in *BP-MR per-packet*.

Fairness. By looking at the size of the download time boxplots in Figure 2 we can reveal the degree of fairness between concurrent TCP flows (more tight they are, more fair are the flows). *BP-MR per-packet* exhibits a high degree of variability, indicating a poor fairness between parallel TCP file transfers. In this case, fairness is sacrificed in an attempt to make the most out of the network resources blindly by taking routing decisions on a per-packet basis without caring about segment ordering at the destination. *OLSR* also exhibits a lower fairness degree compared to *BP-MR per-flow* that increases with the number of concurrent file transfers, given its trend to prioritize file downloads launched by UEs closer in terms of hops to the EPC.

Response to an increasing load. Increasing the number of concurrent downloads, we can see that the download time increases for all the variants. The degree of such increase, however, is different. Moreover we can see for *OLSR* a decreased fairness, with the candlesticks that dramatically widen from 25 to 100 UE. For what regards *BP-MR per-packet*, that degree of wideness is reduced, while with *BP-MR per-flow* practically we have the same performance between 75 and 100 UE, but the fairness decreases from the 25 and 50 UE cases.

Latency. Looking to the RTT values of Figure 2, we can see that *BP-MR per-flow* experiences the highest RTT values. This can be explained by the fact that these flows have a lower probability to experience RTOs and reordering, and then more samples can be collected (as we described in the Methodology subsection). For *OLSR* and *BP-MR per-packet* we have lower RTT values, with *OLSR* performing slightly better, but higher finish time, as we said before. With

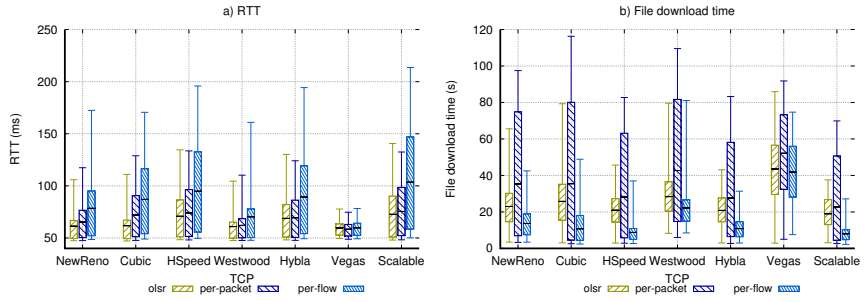

Figure 3: File download time and RTT for different TCP, with 100 UEs performing a 10 MB download

more time to conduct a file transfer, both routing protocols are more likely to exhibit lower RTT samples, thanks to the slow probing phase that follows a congestion event, since these packets will encounter less congestion along the way.

Insights. *OLSR* suffers when increasing the load, and generally has no defense against congestion but the fixed path selection gives in-order delivery and practically constant delays. On the other hand, *BP-MR per-packet* dynamically avoid congestion, but leads to a high degree of reordering, worsening the TCP performance with respect to *OLSR*. The best results are obtained with *BP-MR per-flow*, that blends the best from the other two strategies: fixed paths and congestion avoidance. In the following, we investigate if these insights are also valid when employing different TCP congestion control algorithms.

3.3. TCP Congestion Control Impact

To analyze the impact of TCP congestion control algorithm over the overall performance obtained through the routing protocol strategies, we will refer to Figure 3, which contains results achieved with 100 user devices connected.

Throughput. The smallest average finish time, considering all the routing protocols, is achieved with Scalable, followed by Highspeed and Hybla variants. Contrarily to what we could have expected giving its predominance on the Internet, Cubic is not performing well: in this particular case, the average value

indicates that its performance is comparable with the New Reno variant, and moreover Cubic presents the highest download time for one, unlucky, flow. In average, however, the worst throughput performance is achieved by Vegas.

Fairness. Analyzing the wideness of the boxes, the most fair TCP congestion control algorithm is Scalable, that presents very tight boxes for all the routing protocols. Other algorithms seem to suffer more, with Westwood that presents the worst fairness indicator. In general, the fairness is increased through the use of the *BP-MR per-flow* routing protocol.

Latency. TCP Vegas is clearly offering a practically constant RTT, regardless of the routing protocol. This value is also the smallest registered, due also to the low throughput, emphasizing the delay-based mechanism of Vegas: in the next subsection we will analyze in depth the Vegas behavior. On the other hand, Scalable and Highspeed present the higher average values, near the 100 ms mark. We would like to remark the fact that this value indicates the time that, in average, is required from the moment that a segment leaves the TCP buffer to the time that its ACK is received by the sender node.

Congestion control characteristics. In the following, we briefly recap the most important result for each congestion algorithm:

NewReno does not suffer the competition with other loss-based variants, in this scenario. For instance, the average RTT and finish time average values are slightly better than Cubic for all the routing protocols;

Cubic puts a lot of stress on the routing protocol, and in fact, its average RTT value is higher than Westwood, Hybla, NewReno, and Vegas. Neither in the finish time it presents better results, performing worse than New Reno with all the routing protocols;

Highspeed presents slower finish time than other versions (except for Scalable), but presents a really different performance giving different routing protocols, achieving the best finish time with *per-flow*;

Westwood thanks to its bandwidth estimation approach is able to maintain

lower RTT, but this reflects on the worsening of the performance related to the finish time;

Hybla obtains similar performance as Highspeed, giving that the low RTT of the network is not permitting to exploit its characteristics;

Vegas is more conservative throughput-wise, but achieves the best results in terms of RTT. This conservativeness implies smaller congestion window;

Scalable does not show a clear improvement in terms of RTT from NewReno, but it shows tight boxes for the finish time, and so demonstrating itself as the fairest.

Routing protocol trend remarks. We can observe that the trend outlined in the previous subsection for the different routing variants is also valid for other loss-based congestion control algorithms than Cubic. Regardless of the employed TCP: *OLSR* provides constant delays but is not fair (wide boxes) and not optimal throughput-wise; *per-packet* has really high finish times, and it is highly unfair between different parallel TCP file transfers, while *per-flow* exhibits lower file finish times at the expenses of an increased RTT. From this information, merged with the results presented in this subsection for different TCPs, we can conclude that for loss-based TCP *BP-MR per-flow* significantly improves the TCP performance, more than the particular TCP variant under evaluation. A different conclusion should be drawn for Vegas, which is delay-based. The performance for RTT and throughput are practically the same, regardless of the routing protocols.

Insights. Overall, loss-based TCP variants finish their download earlier than delay-based TCP variants (Vegas), or variants that use a bandwidth estimation (such as Westwood). This higher amount of bytes in-flight reflects on the RTT, which is generally higher in more aggressive variants, regardless of the routing protocol. The broad literature on the subject supports the obtained result. By contrast with wired networks in redundant topologies the routing protocol matters, as we have seen. This suggests that TCP protocols evalua-

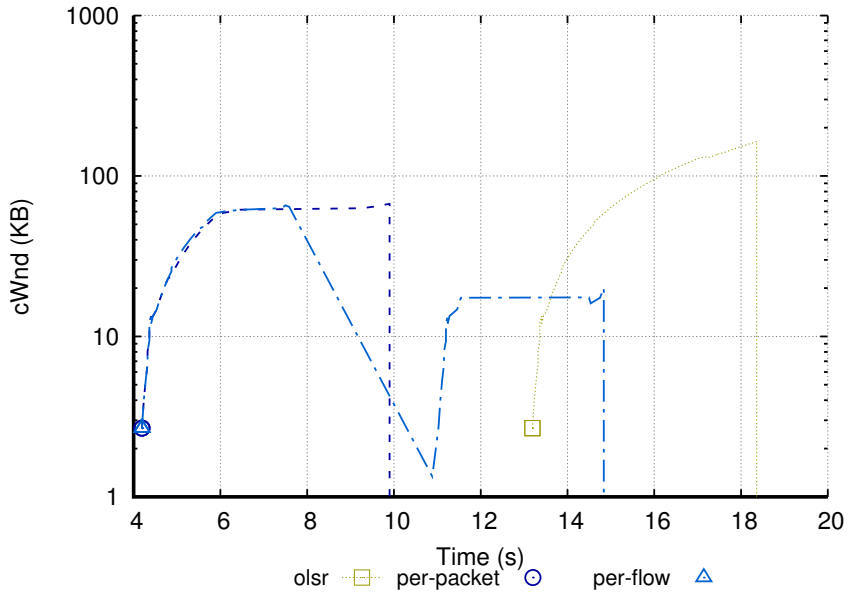

Figure 4: Vegas cWnd evolution for one UE

tion should always be done by keeping in mind the underlying routing protocol. At this point, it is interesting to analyze in detail the Vegas and the Highspeed variants, representative of delay-based and loss-based categories, respectively, to see the differences in the congestion window evolution and if (and how) different routing strategies are influencing their internal mechanism.

3.4. Delay-based versus loss-based TCP variants

As stated before, to better understand the differences between delay-based and loss-based TCP variants on the different routing strategies, we choose one representative congestion window evolution for two TCP variants: Vegas (delay-based) in Figure 4 and HighSpeed (loss-based) in Figure 5.

TCP Vegas, delay-based. For what regards Vegas, its delay-based mechanism for increasing the congestion window maintains the cWnd value under the 100 KB threshold except for *OLSR* (that approaches the 200 KB threshold), and its growth is very slow (please note that the y-axis is logarithmic). This does not create congestion, and the different routing algorithms are not stressed; hence,

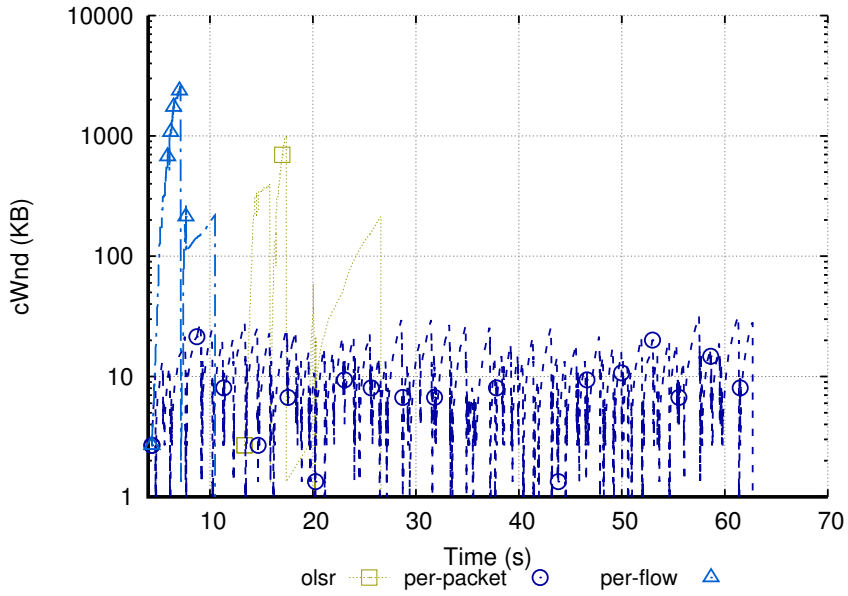

Figure 5: HighSpeed cWnd evolution for one UE

the TCP behavior on top of them is practically the same. Unfortunately, the slow growth and the limited values are reflected also in the download time (and so in the throughput), which is higher than the other variants. An interesting property to analyze is if the routing protocol influences the delay-based mecha-
365 nism at the root of Vegas. Indeed, Vegas depends on the measured RTT value, and so theoretically a *per-packet* strategy (in which each packet has potentially a very different RTT) should penalize the algorithm itself. However, without any other congestion source, the slow growth of Vegas does not introduce any congestion, even with 100 devices, and the *per-packet* RTT is quite similar to
370 the values achieved with other strategies. For sure, if the RTT values start to be very different (e.g. already present congestion) all the mechanisms based on the RTT will be very penalized. In general, it is widely known that the throughput of delay-based methods is worse than the throughput of more aggressive variants, but adding multi-path routing protocols on the backhaul network is
375 worsening the situation, penalizing them more than loss-based strategies.

Highspeed, loss-based. On the other hand, loss-based HighSpeed variant produces some stress on the routing protocols, and each one is responding differently, producing different shapes of the congestion window. Congestion events with *OLSR* are due to packet losses, clearly visible from the drastic decrease of the window that is happening near the same y-value. In fact, each hop is using a drop-tail queue that, after being filled by the amount of in-flight data, produces losses, actually limiting the sender. *BP-MR per-packet* experiences the lowest congestion window absolute value, and it drops randomly during the simulation. This is due to high degree of reordering introduced by the strategy itself. The reordering is happening for the aggressiveness of the TCP variant, seen by the backpressure strategy as congestion. As a consequence, we can see that the transmission needs longer time with respect to *OLSR* to complete. Finally, with *BP-MR per-flow* variant, the reordering is not happening anymore, but still, the flow can avoid congested paths; the congestion window can grow to a more appropriate value, before being limited by the losses due to a buffer overflow, in the same way as *OLSR*. Overall, this allows *BP-MR per-flow* to reduce the transmission time with respect to both *OLSR* (thanks to the congestion avoidance strategy of the routing protocol itself) and *BP-MR per-packet* (thanks to avoiding reordering at the destination, by keeping the flow path fixed).

3.5. Workload Impact

At this point, it is interesting to know if the behavior of the delay-based or loss-based variants remains the same when increasing the workload on the mesh. To do so, we modeled existing LTE traffic as a constant-rate UDP traffic, composed by different flows directed among random SC source-destination pairs. In the following, we will refer to the aggregated UDP traffic in the entire mesh network, instead of characterizing throughput performance of every single UDP flow. We will not consider latency of these flows because an in-depth analysis of UDP traffic within the *BP-MR* framework was already conducted in [3].

TCP Throughput. In Figure 6 it is reported the download finish time

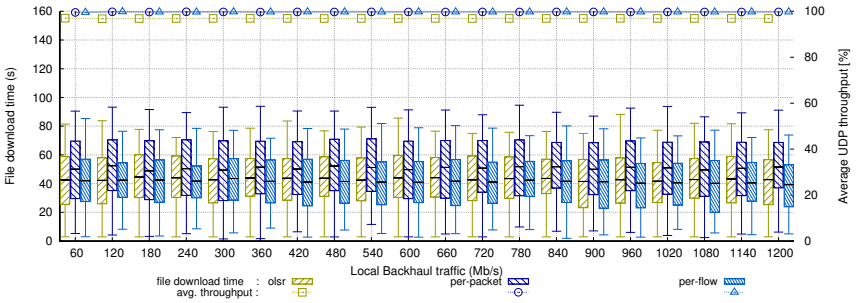

Figure 6: Time required to perform a 10 MB download from a remote node using TCP Vegas under different backhaul workload conditions

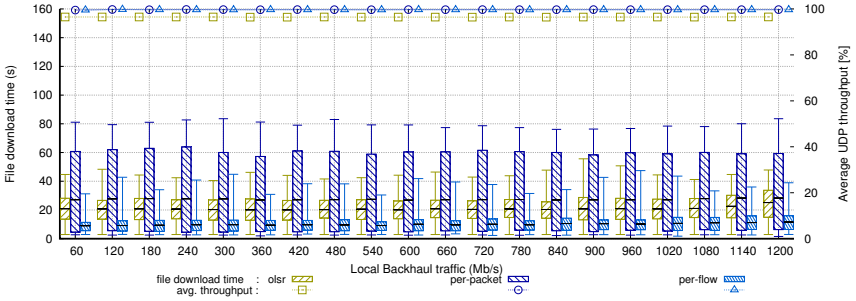

Figure 7: Time required to perform a 10 MB download from a remote node using TCP Highspeed under different backhaul workload conditions

statistics (over a 10 MB file) of 100 UEs employing TCP Vegas, increasing the aggregated UDP traffic from 60 Mb/s to 1200 Mb/s. The TCP performance on top of *OLSR* and *BP-MR per-flow* are practically the same, with some small differences between the two on the higher whisker. *BP-MR per-packet* is generally performing worst, with wide boxes and generally an higher average values. By comparing the 1200 Mb/s case with the result for Vegas reported in Figure 3, we can state that the general trend is the same, or in other words the increased load on the backhaul is not reflected in the behavior of Vegas on top of the analyzed routing strategies.

Even for loss-based TCP, the situation does not change. From Figure 7, where we plot the download finish time statistics employing TCP Highspeed, the trend does not change while increasing the backhaul load. However, a small difference between this result and the previous with Vegas is that, even if very slowly, the average download time increases with the increase of the load. This is better visible by looking at the higher whisker of each case, that represent the worst download time: for OLSR and *BP-MR per-flow* it increases with the load, except for some situations in which the randomness of the simulation is coming into play (even if simulations are repeatable, changing the backhaul load implies different moments on which drops occur, changing the outcome by a small delta, but without changing the overall picture). On the other hand, we have practically a constant trend for *BP-MR per-packet*, where the reordering problem has a bigger impact on the performance rather than queue drops. By comparing the 1200 Mb/s case with the results obtained for Highspeed in Figure 3, is clearly visible how the routing protocol affects the performance of the TCP flows in the same way, regardless of the backhaul load, like in the delay-based case.

UDP Throughput. For both Figures 6 and 7, we plot on the right side the average UDP throughput. The value is reported as a percentage of the measured traffic at the endpoints over the injected traffic. The routing protocols can deliver more than the 95% of the traffic (with *per-flow* and *per-packet* up to 100%). The reason of the lower result of *OLSR* is in the time it requires to populate the routing tables of each backhauling node, and so zeroing the

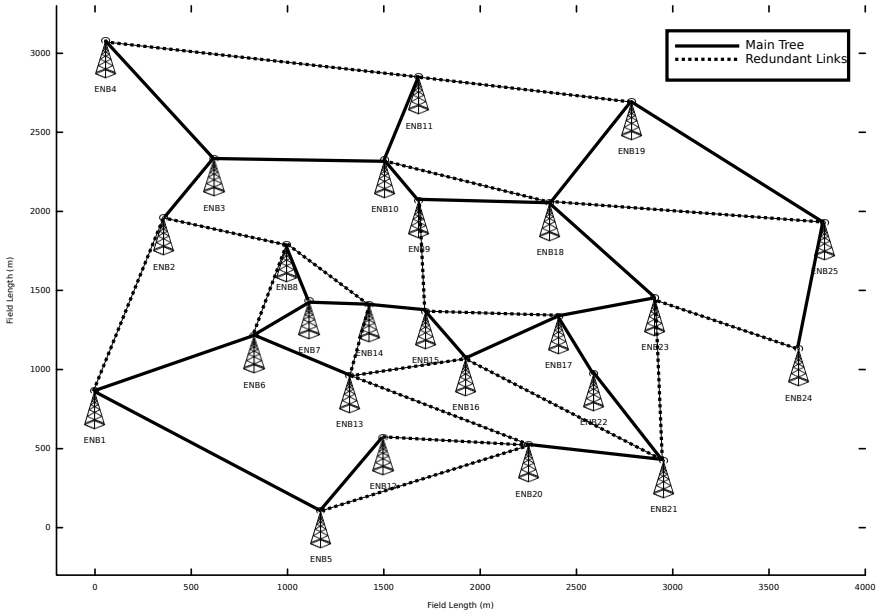

Figure 8: Modena real topology.

throughput for the very first seconds of the simulation. The result is derived from the high competition that is going on between the TCP and UDP flows. The 100 TCP flows are increasing their congestion window slowly, and since UDP constant rate is unresponsive (i.e. it does not reduce the traffic after congestion event) it is not overall penalized by the routing strategies, even after the saturation point (in which, without any scheduling, TCP flows will starve).

4. Analyzing irregular backhaul topologies

To conduct experiments in an irregular topology, we used the eNodeB placement locations available through the service of ARPA (Agenzia Regionale Protezione Ambiente) Emilia-Romagna [3]. We extracted the geographic coordinates of the eNodeBs in the city center of Modena for a single telco company. The resulting topology modeled in ns-3 covers a field of 12 Km^2. Regarding back-

[3] https://www.arpae.it/cem/webcem/modena

haul links between eNodeBs, we have placed them following the most recent works in the field [10, 11, 12], given the unavailability of them to the public. As a result, we conducted two different set of experiments: the first on top of a non-redundant tree topology, while the other on a redundant ring-tree topology. The plain tree is built by connecting closer eNodeBs, with each eNodeB that has a maximum of 2 connections towards the neighbors. The redundant topology starts from the plain tree, but then others links are added to reach a maximum of 4 towards the neighbors. Both topologies are represented in Figure 8, in which the plain tree is composed by the eNodeBs connected by continuous lines, while the redundant is formed by the eNodeBs connected by continuous and dotted black lines.

UEs are specifically placed to create a congestion situation in the very center of the deployment (near the Duomo, in the central Modena piazza). In particular, we placed 7 UEs for each eNodeB in the center (namely ENB7, ENB8, ENB14, ENB15, ENB16), and 2 for every other eNodeB. Since the field is covered by 25 eNodeB, there is a total of 75 UEs. All the rest is unchanged from the previous simulation set (data chunk size, LTE frequencies and error models, backhaul link bitrate and delays).

The reason for simulating these topologies and heterogeneous traffic patterns is two-fold. First, to validate the aforementioned findings obtained with the grid mesh topology in a more realistic and irregular backhaul deployments. Second, to provide a study of fairness performance amongst TCP flows under non-homogeneous traffic loads and deployments.

4.1. Throughput and delay analysis

Plain Tree topology. We start our analysis from the plain tree topology. The objective is to demonstrate that even without using a redundant topology, the performance improvements previously observed with *per-flow* still hold. We analyzed the RTT and finish time of 75 TCP Cubic flows in the tree topology, and reported the results in Figure 9 and Figure 10. Combining these results we can draw a very similar picture with respect to that obtained with the regular

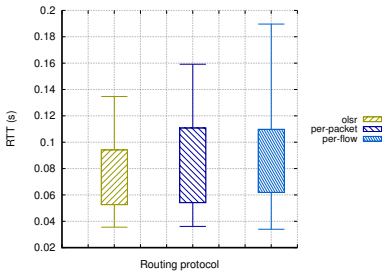

Figure 9: Aggregate RTT of 75 TCP Cubic flows over different routing protocols, plain tree Modena scenario.

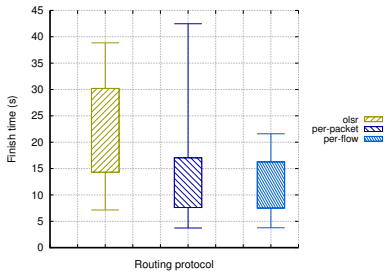

Figure 10: Aggregate finish time of 75 TCP Cubic flows over different routing protocols, plain tree Modena scenario.

mesh. Backpressure protocols still have bigger RTT than *OLSR*, but the time required to finish a download is considerably lower. An interesting result is that not only *per-flow* have better throughput performance, but also with *per-packet* the flows can complete the download earlier than *OLSR*. The lack of redundant paths is forcing the protocol to choose the same output interface for subsequent packets, leading to less reordering with respect to what is happening in a redundant topology, such as the theoretical mesh. Performance degradation of *OLSR* is because of the time that the protocol needs to build its view of the network, which inevitably reflects in the download finish time. Therefore, even in topologies with no (or low) redundancy the *per-flow* protocol can maintain good performance.

Redundant Modena topology. The redundant Modena topology is built on top of the tree topology and is characterized by the presence of redundant paths between the eNodeB. The objective of the following is to prove that the TCP performance improvements of *per-flow* continue to hold in a redundant and irregular topology. Figure 11 reports the RTT, whereas Figure 12 shows finish times of TCP flows. The effect of incrementing redundancy on the tree topology is clearly visible on the finish time results and confirms main findings observed with the regular mesh topology: *per-packet* is penalized by the abundance of available paths, while *per-flow* can exploit the increased redundancy, outperforming *OLSR*. The results for *per-flow* and *OLSR* are in line with the previous

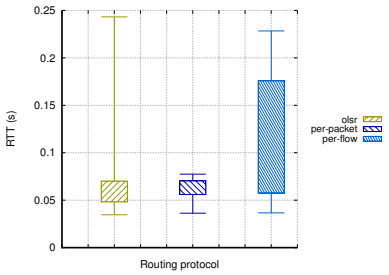

Figure 11: Aggregate RTT of 75 TCP Cubic flows over different routing protocols, redundant Modena scenario.

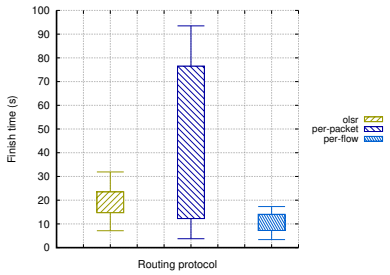

Figure 12: Aggregate finish time of 75 TCP Cubic flows over different routing protocols, redundant Modena scenario.

experiment with the plain tree topology, improving the finish time performance by a small factor. In fact, *OLSR* can find shorter paths to the destination, using in average less time to complete the download but is still limited by the impossibility to distribute the load to other (less congested) paths.

4.2. Fairness

Given the heterogeneous traffic demands in Modena topology, we evaluate the fairness performance amongst TCP flows. To define fairness, one of the most used metric is Jain's Fairness Index (JFI), a normalized metric bounded between 0 and 1, defined as (x_i is the average throughput of the flow i, with a total of n flows):

$$JI = \frac{(\sum_n^{i=1} x_i)^2}{n \cdot \sum_n^{i=1} x_i^2}$$

However, this metric is static and does not consider the different starting time of flows. Moreover, the Jain's fairness index is proposed assuming a simple model of n sources sharing the same bottleneck link, which is good to describe the static properties of competing flows, but the characteristics of new architectures (such as redundant ones) and new routing protocols can not be well captured in all aspects by that model. In the previous section, we have used the download finish time as fairness indicator: more close are these values together, more fair is the behavior of the network towards the users (giving a transmission

of 10 MB for each user towards the same remote node) because of different flow start times.

In the following, we provide a picture of the fairness between TCP flows through the use of JFI. We have adapted the simulations by using the same start time for every flow, but maintaining the same destination for each flow, and using one single bottleneck to reach the destination. We have used the two topologies based on Modena data, with the same UE distribution as before. Instead of the average throughput, we have used the average goodput of each flow, in order to not consider retransmissions, providing a useful end-to-end performance indicator. The average goodput in bit/s is calculated by dividing the fixed amount of data to transmit (10 MB) by the end-to-end finish time of each flow. We have then used the JFI to calculate the values of fairness: 1 is the ideal value in which all flows take the same time to complete.

We can look at the JFI for the plain tree topology in Figure 13. The value is close to 0.6 for all routing protocols, indicating that without redundancy there is not much difference between them. The result is also related to the TCP congestion control used, in this case Cubic. Probably by using a delay-based variant this value could have been bigger (as the broad literature on the subject suggests), but the interesting fact here is that in a plain tree topology, with no redundancy, the routing protocol does not impact the fairness, as we could expect from wired networks. In fact, non-redundant topologies are the closest to the plain wired networks, in which TCP was developed and, over the years, tested.

By switching to Figure 14, in which we depict the fairness results obtained in the redundant topology, we can see a different trend. *OLSR* is maintaining roughly the same value (i.e., near 0.6), while the *per-packet* situation is worse, decreasing the fairness index to 0.4. On the contrary, with *per-flow* the situation is clearly better, with an index of 0.8. By increasing the number of links, the JFI between flows has improved. Interestingly, with *per-flow* and in a scenario very different from a wired network, TCP is able to reach fairness value close the perfection. This means that the TCP protocol itself is good enough to couple

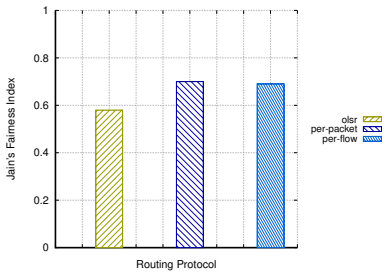

Figure 13: Jain's Fairness Index of 75 TCP Cubic flows over different routing protocols, plain tree Modena scenario.

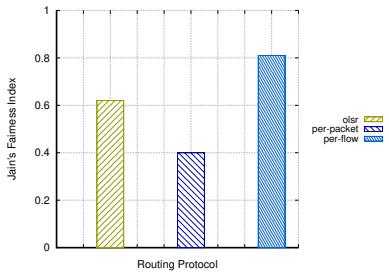

Figure 14: Jain's Fairness Index of 75 TCP Cubic flows over different routing protocols, redundant Modena scenario.

with the new architectures, in which redundancy exists and often exaggerated, but more investigation should be done considering the couple TCP and routing protocols as a block, and not investigating them separately.

5. Related work

This is the first work that extensively analyzes different TCP variants and different backhaul routing strategies to carry LTE traffic over wireless mesh networks. Previous work has been focused on adding multi-path TCP (MPTCP) [13] extensions at the transport layer. However, this would require modifications at the LTE mobile clients and Internet servers. Besides, MPTCP is end-to-end and can exploit the path diversity only at endpoints. Therefore, it can not help when such diversity is located and dynamically exploited inside the backhaul network, which is the primary focus of the analyzed routing algorithms. Many works, such as [14], use a wired network with no redundancy as environment for comparing the different congestion control algorithms. For instance, in many works the dumbbell topology is used; we aim to provide a comparison in redundant topologies, with the support of specialized routing protocols. For what regards mobile networks, a congestion control comparison is performed in [15], where the authors focus on three different congestion control (Cubic, NewReno, and Westwood+) and investigate the relation between the algorithms and the bufferbloat problem in commercial 3G and 4G networks. Interestingly, their

findings show how 3G and 3.5G networks highly suffer from bufferbloat, and how aggressive congestion controls (such as Cubic) significantly increase the response time to the users. In our paper, we also include the routing protocol in the analysis, using redundant topologies. Our objective is to demonstrate that the TCP congestion control algorithm is not the only component to define the expected performance, standing out from the previous literature. Moreover, using routing protocols and TCP congestion controls as variables allowed us to see a big performance improvements, due to the backpressure-based root of *BP-MR per-flow*, even without using custom layer approach (such as in [16] or [17]).

Wireless Mesh Networks topologies are not new in the academic world. When the network is dense, especially in client meshing (i.e. nodes collaborate in absence of a backhaul), the routing overhead can be very high. One technique to reduce the overload added by the routing management messages is to selectively enable or disable the routing functionality in nodes, actually implementing a topology control [18]. On the contrary, we assume that the topology is fixed in the network design phase, with our objective that is to maximize the usage of network resources. Therefore, turning off a routing node is a possibility not contemplated in this work.

Congestion algorithms comparison in an ad-hoc network has been performed in [19], but using only one routing protocol, Dynamic State Routing (DSR). The results presented in the paper clearly indicate that the DSR achieves maximum throughput, higher packet delivery ratio and a less average end-to-end delay with TCP Vegas, a delay-based congestion control. This is somehow against the main conviction that delay-based algorithms incur in low delay but also low throughput: this confirms once more that, in a wireless environment, is important to measure the performance by evaluating together routing protocols and congestion control algorithms: different combination provides always different results. In our case, the use of Vegas does not give the best combined delay/throughput results with any of our routing protocols.

TCP performance over the LTE RAN has been investigated in both simulated environments and over real data. For the simulated environment, many

works (such as [20, 21]) simulate the access network with simple point-to-point links, with different properties, and so without taking into account the complex dynamics of the Radio Resource Control (RRC) state machine and the TCP protocol, thus invalidating the obtained results. Nguyen et al. in [22] investigate the performance of TCP over the full RAN stack, concluding that increasing the load in a cell can significantly throttle the bandwidth available to a UE, thus increasing the experienced delay, especially when the eNodeB maintains a large per-UE queue. This can invalidate the estimated RTO value, causing unnecessary TCP timeouts even when no packets are lost. Also, they concluded that radio-link handover could cause significant performance degradation. Similar conclusions are drawn in [23], where the authors analyze a large-scale real LTE data set to study the impact of protocol and application behaviors on the LTE network performance. For instance, they concluded that some TCP behaviors (such as not updating the RTT estimation using the duplicate ACKs) could cause severe performance issues; also, the bandwidth utilization ratio is usually below 50% for large flows. The work provides valuable insights on the interaction between TCP and LTE. However, the details of the backhaul network are out of the analysis (e.g. the routing algorithm). We aim to fill this hole by adding the analysis of different backhaul routing protocols over many TCP variants assuming a constrained wireless mesh backhaul.

For analysis of TCP traffic over backpressure routing we refer to [24], that identify the packet reordering at the receiver as the main issue with backpressure routing, and then proposes a delayed reordering algorithm at the destination for keeping packet reordering to a minimum. Other proposals use the MAC layer to perform such scheduling, such as in [25]. While using the MAC layer ties the proposal to a particular technology (in [25] is used an IEEE 802.11-based wireless mesh network), we believe that avoiding reordering is more profitable than reordering packets in later stages. Under this light, we proposed the *BP-MR per-flow* variant. Furthermore, talking about backpressure-based algorithms, in [26] it is shown that TCP experiences incompatibilities with backpressure strategies that maintain per-flow queues, hence leading to unfairness between flows. In

contrast to this work, we analyze the performance of *BP-MR*, a distributed and scalable backpressure-based routing protocol that maintains per-interface queues, which does not require changes at the TCP layer.

The history of *BP-MR* started with *BP* [27], a self-organized backpressure routing protocol that is a decentralized flavor of the original centralized backpressure algorithm. For dealing with sparse networks, Backpressure for Sparse Deployments (*BS*) [28] included additional extensions to *BP*. In particular, *BS* added a penalty function able to overcome dead ends in a scalable and decentralized way. However, *BS* was designed to tackle sparse topologies where nodes are equipped with a single backhaul radio and presented huge inefficiencies in multi-radio deployments (i.e. with multiple backhaul interfaces). To mitigate such inefficiencies, we proposed in [3] *BP-MR*, used in this paper and detailed for the sake of completeness in Section 2.

In our work, each routing node of a multi-hop path from the source to the destination cooperates in sending information to its neighbor nodes. Under such cooperative view, we can compare *BP-MR* with other routing protocol proposals. For instance, one of the most practical environment to evaluate these proposals is a Wireless Sensor Networks (WSN) deployment. In [29] is presented a Distributed Adaptive Cooperative Routing (DACR) protocol, that starts from a path created by Ad hoc On-Demand Distance Vector (AODV) and then refined, hop by hop, by introducing delay and energy constraints gathered through a reinforcement learning method. The main difference with our work is that we use point-to-point links to model the wireless network, instead of the point-to-multipoint typical of WSN. In point-to-multipoint scenario, the quality of a link between two neighbors plays a crucial role in the routing decision, while that quality in our environment is fixed, and overtaken in importance by the queuing delays (not analyzed in [29]).

6. Conclusion

This paper extensively evaluates throughput, fairness, scalability, and latency of different TCP congestion control algorithms (TCP New Reno, Cubic, Highspeed, Westwood, Hybla, Scalable, and Vegas) in the context of current and future wireless mesh backhauls carrying LTE traffic. We analyzed fixed-path and different backpressure-based routing strategies : *OLSR*, *BP-MR per-packet*, and the *BP-MR per-flow* variant. Experiments conducted with ns-3 revealed some findings regardless of the backhaul topology under evaluation. First, simulations with delay-based TCP Vegas, being conservative, are incapable of stressing the backhaul yielding into similar performance over the three analyzed routing strategies.

Second, loss-based variants such as Cubic (which is used by default by the Linux operating system) pose difficulties to the routing protocols, due to their aggressiveness, and then differences in the performance are clearly visible independently from the variants themselves. With loss-based TCP, *BP-MR per-flow* decisions offer the best flow performance, regardless of the TCP variant, despite its higher round-trip-time irrespective of the backhaul topology under evaluation.

6.1. Lessons Learned

The evaluations presented let us state the following generic recommendations:

- Wireless backhaul deployments requires flexible routing protocols, in order to evenly exploit backhaul redundant resources;
- There is a trade-off between routing flexibility and TCP performance. We have shown how per-packet routing decisions (highest degree of load balancing for system efficiency) can be good for UDP traffic, but it experiences the worst results for TCP traffic (with reliability and ordering requirements);

- The per-flow protocol, being able to trade-off between even resource usage and traffic patterns requirements, is the one getting the best results in terms of throughput and fairness;
- The aforementioned results are generalizable to multiple topologies, spanning from current backhaul deployments of eNodeBs (e.g., Modena topology) to future dense SC mesh deployments;

6.2. Future Work: BP-MR per-burst

Future works include (but are not limited to) the investigation of another variant of *BP-MR*, which takes decision on a *per-burst* basis, to have some degree of flexibility (e.g., in case of long-lasting flows) without incurring in the reordering penalization typical of *BP-MR per-packet*, to be tested on different topologies with different degree of randomness.

References

[1] W. Webb, Wireless Communications: The Future, John Wiley & Sons, 2007.

[2] Z. Jiao, B. Zhang, C. Li, H. T. Mouftah, Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey, IEEE Wireless Communications 23 (2016) 102–110.

[3] J. Baranda, J. Núñez-Martínez, J. Mangues-Bafalluy, BP-MR: Backpressure Routing for the Heterogeneous Multi-radio Backhaul of Small Cells, in: 2015 8th IFIP Wireless and Mobile Networking Conference (WMNC), 2015, pp. 48–55. doi:10.1109/WMNC.2015.31.

[4] N. Patriciello, C. Grazia, J. Núñez-Martínez, J. Baranda, J. Mangues-Bafalluy, M. Casoni, Performance Evaluation of Backpressure Routing in Integrated Satellite-terrestrial Backhaul for PPDR Networks, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2016 IEEE 12th International Conference on, 2016, pp. 110–117.

[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, Hedera: Dynamic Flow Scheduling for Data Center Networks., in: NSDI, volume 10, 2010, pp. 19–19.

[6] M. Casoni, N. Patriciello, Next-generation TCP for ns-3 simulator, Simulation Modelling Practice and Theory 66 (2016) 81–93.

[7] Small Cell millimeter wave mesh backhaul, Whitepaper, 2013. URL: http://www.interdigital.com/research_papers/2013_01_25_small_cell_millimeter_wave_mesh_backhaul.

[8] K. Jamshaid, B. Shihada, A. Showail, P. Levis, Deflating link buffers in a wireless mesh network, Ad Hoc Networks 16 (2014) 266–280.

[9] The European table of frequency allocations and applications in the frequency range 8.3 kHz to 3000 GHz (ECA table), European Conference of Postal and Telecommunications Administrations (CEPT), 2014. URL: http://www.grss-ieee.org/wp-content/uploads/2014/07/ECA_European_Table_of_Frequency_Allocations_May_2014.pdf.

[10] F. C. Kuo, F. A. Zdarsky, J. Lessmann, S. Schmid, Cost-efficient wireless mobile backhaul topologies: An analytical study, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, 2010, pp. 1–5. doi:10.1109/GLOCOM.2010.5683870.

[11] D. Bojic, E. Sasaki, N. Cvijetic, T. Wang, J. Kuno, J. Lessmann, S. Schmid, H. Ishii, S. Nakamura, Advanced wireless and optical technologies for small-cell mobile backhaul with dynamic software-defined management, IEEE Communications Magazine 51 (2013) 86–93.

[12] R. Nadiv, T. Naveh, Wireless backhaul topologies: Analyzing backhaul topology strategies, Ceragon White Paper (2010) 1–15.

[13] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar, et al., Architectural guidelines for multipath TCP development, IETF, Informational RFC 6182 (2011) 2070–1721.

[14] T. Lukaseder, L. Bradatsch, B. Erb, R. W. V. D. Heijden, F. Kargl, A Comparison of TCP Congestion Control Algorithms in 10G Networks, in: 2016 IEEE 41st Conference on Local Computer Networks (LCN), 2016, pp. 706–714. doi:10.1109/LCN.2016.121.

[15] S. Alfredsson, G. D. Giudice, J. Garcia, A. Brunstrom, L. D. Cicco, S. Mascolo, Impact of TCP congestion control on bufferbloat in cellular networks, in: 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2013, pp. 1–7. doi:10.1109/WoWMoM.2013.6583408.

[16] C. Y. Huang, P. Ramanathan, Network Layer Support for Gigabit TCP Flows in Wireless Mesh Networks, IEEE Transactions on Mobile Computing 14 (2015) 2073–2085.

[17] A. A. Al Islam, V. Raghunathan, iTCP: an intelligent TCP with neural network based end-to-end congestion control for ad-hoc multi-hop wireless mesh networks, Wireless Networks 21 (2015) 581–610.

[18] A. Vázquez-Rodas, J. Luis, A centrality-based topology control protocol for wireless mesh networks, Ad Hoc Networks 24 (2015) 34–54.

[19] M. K. Goyal, Y. K. Verma, P. Bassi, P. K. Misra, Performance evaluation of TCP congestion control variants using Dynamic State Routing in wireless AD-HOC network, in: Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing, Springer, 2013, pp. 445–449.

[20] G. A. Abed, M. Ismail, K. Jumari, Traffic Modeling of LTE Mobile Broadband Network Based on NS-2 Simulator, in: Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on, 2011, pp. 120–125. doi:10.1109/CICSyN.2011.36.

[21] G. A. Abed, M. Ismail, K. Jumari, Behavior of cwnd for TCP source

variants over parameters of LTE networks, Information Technology Journal 10 (2011) 663–668.

[22] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, J. Van der Merwe, Towards Understanding TCP Performance on LTE/EPC Mobile Networks, in: Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges, AllThingsCellular '14, ACM, New York, NY, USA, 2014, pp. 41–46. URL: http://doi.acm.org/10.1145/2627585.2627594. doi:10.1145/2627585.2627594.

[23] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, O. Spatscheck, An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance, in: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, ACM, New York, NY, USA, 2013, pp. 363–374. URL: http://doi.acm.org/10.1145/2486001.2486006. doi:10.1145/2486001.2486006.

[24] B. Radunović, C. Gkantsidis, D. Gunawardena, P. Key, Horizon: Balancing TCP over Multiple Paths in Wireless Mesh Network, in: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, MobiCom '08, ACM, New York, NY, USA, 2008, pp. 247–258. URL: http://doi.acm.org/10.1145/1409944.1409973. doi:10.1145/1409944.1409973.

[25] F. Nawab, K. Jamshaid, B. Shihada, P. H. Ho, Fair packet scheduling in wireless mesh networks, Ad Hoc Networks 13 (2014) 414–427.

[26] H. Seferoglu, E. Modiano, TCP-aware backpressure routing and scheduling, in: Information Theory and Applications Workshop (ITA), 2014, IEEE, 2014, pp. 1–9.

[27] J. Núñez-Martínez, J. Baranda, J. Mangues-Bafalluy, Experimental evaluation of self-organized backpressure routing in a wireless mesh backhaul of small cells, Ad Hoc Networks 24 (2015) 103–114.

[28] J. Núñez-Martínez, J. Baranda, J. Mangues-Bafalluy, A self-organized backpressure routing scheme for dynamic small cell deployments, Ad Hoc Networks 25 (2015) 130–140.

[29] M. A. Razzaque, M. H. U. Ahmed, C. S. Hong, S. Lee, Qos-aware distributed adaptive cooperative routing in wireless sensor networks, Ad Hoc Networks 19 (2014) 28–42.

***Author Biography**

Natale Patriciello received his MS in Computer Science from the University of Bologna in 2013. He is currently a PhD candidate in Information and Communication Technologies at Department of Engineering "Enzo Ferrari" of UNIMORE. He has been involved in the Italian PRIN 2009 project "SFINGI: SoFtware-routers to Improve Next-Generation Internet" and the EU FP7 Project "PPDR-TC". His research interests include distributed systems and computer networking.

José Núñez-Martínez is currently a Researcher in the Mobile Networks Department at CTTC in Barcelona (http://networks.cttc.es/mobile-networks/). He received a MSc (2005) degree in Computer Science Engineering from the Technical University of Catalonia (UPC) and the PhD (2014) from the Computer Architecture Department of the same university (http://www.upc.edu). From 2004 until September 2007, he worked as Network Engineer in the Advanced Broadband Communication Center (CCABA) of UPC. He joined CTTC in September 2007 as software network engineer. He has participated in several national (Cicyt), European (FP7, H2020), and industrial projects (Ditech, and AVIAT). He is author and co-author of more than 20 research papers. His current research interests include: wireless backhaul, small cells, network protocols, self- organization, network optimization, software-defined networking and network function virtualization.

Jorge Baranda Hortigüela is currently a Researcher in the Mobile Networks Department at CTTC in Barcelona. He received a Msc (2008) degree in Electrical Engineering from the Technical University of Catalonia. Before joining CTTC in November 2009, he did an internship at Philips Research (Eindhoven, The Netherlands). At CTTC, he has participated in several national (Cicyt), European (FP7, H2020), and industrial projects (AT4) developing different proof of concepts of novel wireless communication systems and efficient routing strategies for mobile network backhauling. His current research interests include: wireless communications, wireless backhaul, routing network protocols, network optimization, software-defined networking and network function virtualization.

Maurizio Casoni is Associate Professor of Telecommunications in the Department of Engineering "Enzo Ferrari" (DIEF) at University of Modena and Reggio Emilia (UNIMORE), Italy. He received his M.S. with honors and his Ph.D. in electrical engineering from University of Bologna, Italy, in 1991 and 1995, respectively. In 1995 he was with the Computer Science Department at Washington University in St. Louis, MO, as a Research Fellow. He was responsible at UNIMORE for the EU FP7 Projects E-SPONDER and PPDR-TC.

Josep Mangues-Bafalluy is Researcher and Head of the Communication Networks Division of the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). Prior to that, he coordinated the activities of the IP Technologies Area of the CTTC since June 2003. Projects in which he is currently or has been involved: XHAUL

(ICT-H2020), BeFEMTO (ICT-FP7), SCALE (Industrial-Aviat Networks), DESSERT (Industrial-Cisco), EXTREME Testbed (CTTC-internal) and he is leading the Spanish 5G-NORM research project.