TCP-Aware Scheduling in LTE Networks

Narges Shojaedin
University of Calgary
nshojaed@ucalgary.ca

Majid Ghaderi
University of Calgary
mghaderi@ucalgary.ca

Ashwin Sridharan
AT&T
asridharan@research.att.com

***Abstract*—Designing scheduling algorithms that work in synergy with TCP is a challenging problem in wireless networks. Extensive research on scheduling algorithms has focused on inelastic traffic, where there is no correlation between traffic dynamics and scheduling decisions. In this work, we study the performance of several scheduling algorithms in LTE networks, where the scheduling decisions are intertwined with wireless channel fluctuations in order to improve the system throughput. We use *ns*-3 simulations to study the performance of several scheduling algorithms with specific focus on Max Weight (MW) schedulers with both UDP and TCP traffic, while considering detailed behavior of OFDMA-based resource allocation in LTE networks. We show that contrary to its performance with inelastic traffic, MW schedulers may not perform well in LTE networks in the presence of TCP traffic as they are agnostic to TCP congestion control mechanism. We then design a new scheduler called Q-MW which is tailored specifically to TCP dynamics by giving higher priority to TCP flows whose queue at the base station is very small in order to encourage them to send more data at a faster rate. We have implemented Q-MW in *ns*-3 and studied its performance in a wide range of network scenarios in terms of queue size at the base station and round-trip delay. Our simulation results show that Q-MW achieves peak and average throughput gains of 37% and 10% compared to MW schedulers if tuned properly.**

I. Introduction

To meet the growing demand for cellular services, cellular operators around the world are deploying 4G networks based on the Long Term Evolution (LTE) standard. The promise of LTE is to provide high data rate and low latency by providing a packet-optimized wireless access and core network. The high-level architecture of an LTE network is depicted in Fig. 1. The underlying physical layer technology in LTE is Orthogonal Frequency Division Multiple Access (OFDMA) in which the radio frequency is divided into many orthogonal subcarriers. A multi-user scheduler at the evolved NodeB (eNB) assigns subsets of subcarriers to individual users allowing for flexible bandwidth sharing in the system.

To efficiently utilize limited radio resources, wireless schedulers generally take into consideration varying user demands and fluctuating wireless channels and give higher priority to users with better channel conditions and/or higher bandwidth requirements. In an LTE network, users regularly report their channel quality indicator (CQI) for each subcarrier to the eNB. Moreover, each user has a dedicated queue at its corresponding eNB to buffer packets that arrive at the eNB for transmission to the user. An LTE scheduler may use the queue size as an indication of a user bandwidth requirement, in conjunction with CQIs, when making scheduling decisions.

There is a large body of work on developing wireless scheduling algorithms that can leverage wireless channel fluctuations [1]. An important class of such algorithms is the Max Weight (MW) scheduling algorithms [2]. In its general form, this class of algorithms prioritizes users with the largest weight for scheduling where the weight is a function of the user queue and channel conditions. There are different approaches for assigning scheduling weights to users [3]. A user's weight can be simply set to its queue size, its maximum achievable rate, or the product of its queue size and maximum achievable rate. A desirable property of such algorithms is that they have been shown to be throughput-optimal for a diverse set of traffic conditions. Specifically, they can stabilize user queues if any other algorithm can. However, a key premise for this property is that the traffic source rate is de-coupled from the wireless channel and scheduler decisions. This is reasonable for UDP traffic, but TCP traffic source rate is strongly correlated to the channel. Another popular scheduler that is widely deployed in modern day networks is the proportional fair (PF) scheduler which seeks to schedule users so as to maximize the proportional fairness metric [4]. However, the PF scheduler also de-couples the traffic source rate from the channel.

There is extensive work on generalizing MW (*e.g.*, to multi-carrier systems [5] and characterizing its performance in terms of delay, throughput and fairness (*e.g.*, see [6]). However, most studies consider only inelastic traffic (*e.g.*, UDP). In contrast, a majority of today's traffic is carried via HTTP which in turn utilizes TCP. The main reason for this is that HTTP traffic is well understood and can pass through middleboxes which may block other traffic (*e.g.*, UDP). TCP, unlike UDP dynamically adjusts its source rate in response to the perceived throughput, which in turn is decided by the scheduler. For example, in conditions where throughput reduces, the TCP source could time-out and drastically reduce its rate, which is undesirable.

While it has been shown that MW schedulers in conjunction with idealized rate-based congestion controllers can indeed perform optimally [7], in practice, TCP congestion controllers are loss-based and involve several artifacts such as time-outs, which are not accounted for in these works. As such, our goal in this paper is two-fold: first, we study the performance of the MW/PF class of schedulers for TCP with simulations that closely mimic real world constraints as well as channel traces to identify any potential shortcomings, and secondly, develop solutions that can address such issues.

Moreover, while designing techniques to improve TCP performance in 3G/4G wireless networks has been extensively studied in the literature [8], our focus in this work is specially on the impact of radio resource schedulers on TCP performance. A closely related work is presented in [9], where the authors propose a TCP-aware scheduling algorithm to reduce the latency of short TCP flows. Their solution requires detail knowledge of TCP flows while our proposed solution requires only the knowledge of user queues which is readily available in LTE networks.

To achieve our goal, we have implemented several variations of MW (see Section III) in *ns*-3. We use the LTE model in *ns*-3 to create an end-to-end LTE network which has all the major elements of a real LTE system including the Evolved Packet Core (EPC). Moreover, we use real channel traces in our evaluation. The LTE model in *ns*-3 provides a detailed implementation of various aspects of the LTE standard [10] such as OFDMA, hybrid ARQ, adaptive modulation and coding, and handoff management.

We study the performances of our implementations of MW in a wide range of network conditions and compare them with other schedulers built-in *ns*-3, such as PF, Round Robin (RR) and Frequency-Domain Maximum Throughput (FDMT). Our results show that MW may not perform well in some network scenarios, and in fact, achieve throughput that is lower than that of other schedulers such as PF. Particularly, we observe that when there are only a few users in a cell (which is common with increasingly smaller cell sizes of LTE [11]), the throughput of MW drops considerably compared to when there are many users in the cell. To remedy this problem, we design a modified version of MW called Q-MW, in which the users whose packet queue at the eNB node is below a threshold are given higher priority regardless of their channel and queue conditions. The goal is to speedup the ACK generation rate from the user and hence encourage TCP to increase its sending rate (following the additive increase mechanism of TCP) in order to prevent the user queue at the eNB from being depleted. When there are only a few users in a cell, if some of them have no data in their queues then MW has limited choices in selecting users for transmission. This will negatively affect the throughput achieved by MW. Notice that this problem is specific to elastic TCP traffic and does not happen with inelastic UDP traffic. With inelastic traffic, in our simulations, user queues remain always saturated. Q-MW has a queue threshold parameter that needs to be tuned properly for a given network scenario. Our simulation results across a wide range of network configurations in terms of queue size at the base station and round-trip delay show that the proposed Q-MW can outperform the traditional MW and PF in terms of the achieved throughput as summarized below:

- With *dynamic* tuning, by 10% and 11% on average and up to 37% and 64%, respectively.
- With *static* tuning, by 7% and 6% on average and up to 29% and 64%, respectively.

The rest of the paper is organized as follows. Section II provides a brief introduction to LTE with specific focus on downlink scheduling. In Section III, we describe in detail the implementation of the new scheduling algorithms in *ns*-3. Our simulation results are discussed in Section IV, while Section VI concludes the paper.

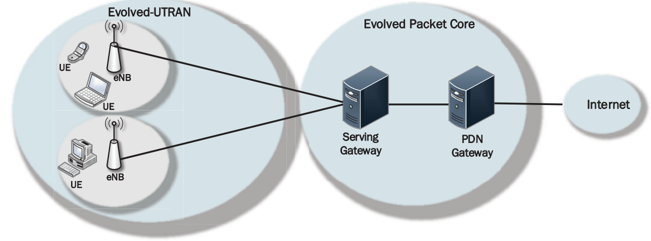


Fig. 1: High-level LTE network architecture.

II. LTE Primer

A. LTE Architecture

LTE is a fourth generation high-speed wireless network that evolved from the Universal Mobile Telecommunication System (UMTS), which in turn evolved from the Global System for Mobile Communications (GSM). The main goals of LTE are spectral efficiency, high data rate for different services (such as VOIP, streaming multimedia and video conferencing), flexible carrier bandwidth, and QoS support.

The high-level network architecture of LTE is depicted in Fig. 1. The network has three main components [12]:

- User Equipment (UE). *e.g.*, a smartphone or tablet which communicates to the LTE network over an OFDMA radio interface.
- Evolved UMTS Terrestrial Radio Access Network (E-UTRAN). The E-UTRAN handles the radio communications between the mobile and the evolved packet core and has one component, the evolved base station, called eNodeB or eNB. Each eNB is a base station that controls the mobiles in one or more cells and is capable of fast resource allocation over time slots of 0.5 milli-seconds. The base station that is communicating with a mobile is known as its serving eNB.
- Evolved Packet Core (EPC). The EPC acts as the intermediate gateway between the radio network (eNB) and the Internet and performs functions such as QoS control, charging, anchor point etc.

A key feature of the LTE network is that the radio interface uses OFDMA to significantly enhance speeds above that of UMTS or EV-DO. OFDMA has features such as high robustness against frequency selective fading and high spectral efficiency, and allows flexible bandwidth sharing which we elaborate on further below.

B. LTE Resource Allocation

In LTE, radio resources are allocated in both time and frequency domains (see Fig. 2). The smallest radio resource unit that can be assigned to a UE is called a physical Resource Block (RB). OFDMA assigns subsets of RBs (not necessarily adjacent ones) to individual users. In the frequency domain, the system bandwidth is divided into multiple sub-channels of

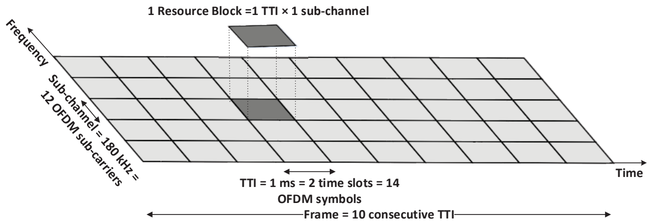


Fig. 2: LTE resource blocks in time and frequency domain.

180 kHz (*i.e.*, groups of 12 OFDM sub-carriers) where each RB spans over one sub-channel. An RB in the time domain is one Transmission Time Interval (TTI) which lasts for two time slots each of length 0.5 ms. Each time slot corresponds to 7 OFDM symbols. The time is divided into frames, each frame is made of 10 consecutive TTIs (or sub-frames). In each TTI, UE measures the pilot signal from the serving eNB and periodically reports the CQI to the eNB. The scheduler uses this indicator along with queue information to determine how many RBs each user is allocated.

C. ns-*3 Simulator and LTE*

In this work, we use the LTE component of the *ns*-3 simulator (LENA) [1] to accurately simulate dynamics of the LTE radio interface [10]. The LENA module closely emulates 3GPP standards for the data plane, faithfully reproducing interactions of the EPC as well as the various stacks of the LTE radio layer such as the PDCP, RRC, MAC and PHY. It incorporates the Femto Cell scheduling framework that reproduces how scheduling takes place according to the 3G framework as well as accurate models for emulating transmission, fading and decoding on the radio interface. As such, the simulator provides a fairly accurate proxy for the actual network.

III. System Model

In this section, we describe the network configuration and parameters that have been considered in our work. We also describe three variations of the Max Weight scheduling policy that we have implemented in *ns*-3.

A. *Network Topology*

We have implemented an LTE-EPC program in *ns*-3 where a remote host sends packets to a number of UEs. Since we are not interested in the effect of handoffs in this work, only one cell is simulated where all UEs are connected to a single eNB node. The UEs are distributed randomly in the cell within a distance of 500 m to 5000 m from the eNB. We use channel fading traces for pedestrian mobility (3 km/h speed) that come with the LENA package [13]. The remote host is connected to the gateway node of the LTE network with a high-speed link (100 Gb/s) in order to avoid any bottleneck effects outside the LTE network. All UEs are connected to the remote host, while multiple TCP/UDP servers run on the remote host, each server is dedicated to one UE. In fact, there is one TCP/UDP flow from the remote host to each UE. Each flow of traffic is generated by remote host and passes through the gateway to the eNB. The eNB maintains a queue for each flow where traffic flow awaits transmission to associated UE. A scheduler in eNB allocates radio resources to flows by following a specific priority metric (*i.e.*, scheduling policy).

[1]We use ns-3.18 which at the time of writing is the latest release.

TABLE I: System Parameters

Parameter	Value
AMC mode	PiroEW2010 [13]
Fading model	trace-based
RLC mode	UM mode
Tx power of eNB	30 dBm
Tx power of UE	23 dBm
Noise figure	5 dB
Internet delay	10 ms, 30 ms, 50 ms, 100 ms
Number of UEs	5, 10, 20
UE locations	random distance from 500 m to 5000 m
Packet size	512 and 1024 bytes
Buffer size at eNB	10, 50, 100, 1000 packet
Simulation time	15 seconds

B. *System Parameters*

Various system parameters are summarized in Table I. Those parameters that are not listed in the table are used with their *ns*-3 default values. As can be seen, some of the parameters that deal with the LTE network setup are fixed during the simulations while those that represent network properties, *e.g.*, Internet delay, vary over a wide range of different values. In this work, we are specifically interested in studying the impact of the following parameters on the performance of TCP/UDP under various scheduling algorithms:

- *Internet delay:* This is the one-way propagation delay between the remote host and the LTE gateway. Using real Internet measurements, it has been reported that a one-way delay of around 10 ms is representative of Internet delays [14]. Thus, we set the default Internet delay in our experiments to this value, *i.e.*, 10 ms, but will run our experiments with a range of Internet delays as well.
- *eNB buffer size:* Some schedulers take into consideration the user queue size when making scheduling decisions. Moreover, packet loss and delay that affect TCP throughput are highly dependent on the queue size. We consider a range of queue sizes covering small and large queues.
- *Packet size:* This is the maximum segment size of TCP/UDP, which is set to 512 Bytes or 1024 Bytes. The default value is 1024 Bytes.

C. *Fairness Metrics*

To compare the schedulers in terms of fairness, two commonly used metrics, namely α-fairness and Jain's fairness, are used. In α-fairness, α is a trade-off factor between fairness and throughput where $\alpha = 0$ gives complete weight to throughput and $\alpha = \infty$ gives complete weight to fairness. A popular choice is $\alpha = 1$, which results in proportional fairness [15]. A scheduler that is proportionally fair, such as PF, should yield the highest metric under α-fairness with $\alpha = 1$. A scheduler is proportionally fair if it maximizes the following metric:

$$\sum_{1 \le i \le N} \log(x_i),$$

where x_i is the throughput achieved by UE i.

Jain's fairness index is measured by the following metric [16]:

$$\text{Jain's fairness index} = \frac{(\sum_{1 \le i \le N} x_i)^2}{N \cdot \sum_{1 \le i \le N} x_i^2}.$$

The maximum value for Jain's index is 1, which is attained when all UEs achieve equal throughput.

D. Max Weight Schedulers

We have implemented three variations of the Max Weight scheduling policy, as described below:

1) **MR:** The Max Rate (MR) scheduler aims at maximizing the overall throughput of the eNB. It allocates each RBG to the UE that can achieve the maximum expected data rate over that RBG in the current TTI. Unlike the built-in FdMt scheduler, MR updates the queue length of the UE that gets allocated a new RBG during a TTI, and also does not allocate any RBG to a UE with nothing to send. Let $R_j(k,t)$ denote the maximum rate achievable by UE j on RBG k at TTI t. Then, the scheduling decision is made as follows:
$$\hat{i}_k(t) = \arg \max_{1 \le j \le N} R_j(k,t),$$
where, N denotes the number of UEs in the cell and $\hat{i}_k(t)$ is the UE chosen for transmission on RBG k at TTI t.
2) **MW:** This version considers both the queue length and channel quality of each UE. The priority metric for this scheduler is the product of queue length and achievable data rate. As for MR, the queue length of the UE allocated a new RBG is updated and no RBG is allocated to a UE with an empty queue. The scheduling decision for MW is performed as follows:
$$\hat{i}_k(t) = \arg \max_{1 \le j \le N} Q_j(t) \cdot R_j(k,t),$$
where, $Q_j(t)$ denotes the queue size of UE j before allocating RBG k at TTI t.
3) **MQ:** The Max Queue (MQ) scheduler uses UE queue length at eNB as its priority metric. MQ simply sorts UEs in decreasing order of their queue lengths. Therefore, the UE which has more data to send has the highest priority for resource allocation.

Our implemented schedulers are compared against the traditional PF scheduler, as well as FdMt (built-in *ns*-3 scheduler aiming to maximize throughput) and RR schedulers. The PF scheduler uses the following policy to schedule users:

$$\hat{i}_k(t) = \arg \max_{1 \le j \le N} (R_j(k,t)/T_j(t)),$$

where, $T_j(t)$ is the past throughput achieved by UE j until TTI t. To calculate $T_j(t)$, a moving average is used as expressed below:

$$T_j(t) = \beta \cdot T_j(t-1) + (1-\beta) \cdot R_j(k,t),$$

where, β $(0 \le \beta \le 1)$ is the weight factor for the moving average.

TABLE II: Default Configuration.

Parameter	Default Value
Packet size	1024 Bytes
Buffer size	100 Packets
Internet delay	10 ms

IV. Simulation Results

In this section, we study the performance of different scheduling algorithms in terms of their achieved throughput and fairness. While our focus is on Max Weight scheduling algorithms, we provide simulation results for other schedulers for comparison purposes.

Most plots in this section represent the total throughput achieved in the cell by all UEs. For these set of plots, the 95% confidence intervals are also plotted as error bars for each plot. Each individual experiment result is the average of 10 independent simulation runs, each lasting for 15 seconds. The default values for simulation parameters are summarized in Table II.

A. Experimental Design

Two sets of experiments are performed to understand the throughput and delay performance of different scheduling algorithms:

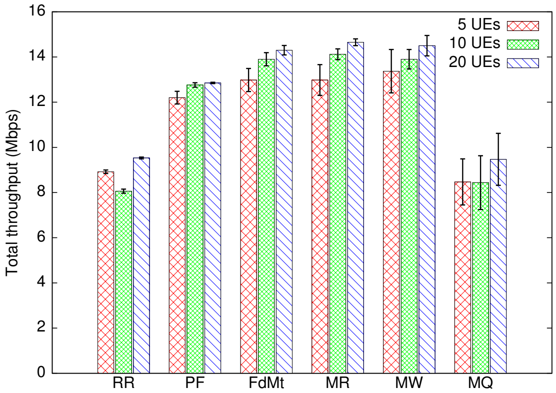
1) *Long-lived flows:* Bulk data transfers of unlimited size are simulated with both TCP and UDP. The goal is to measure the performance of scheduling algorithms in terms of the total throughput and fairness achieved. For the UDP traffic, we allow the remote host to saturate the Internet connection by sending packets back-to-back to the receiving UE.
2) *Short-lived flows:* Motivated by the popularity of web traffic on mobile devices, short TCP flows lasting for only a few round-trip times are also simulated. The goal is to measure the performance of different scheduling algorithms in terms of the time it takes to download objects of various sizes from a remote host.

B. Long-Lived Saturated Flows

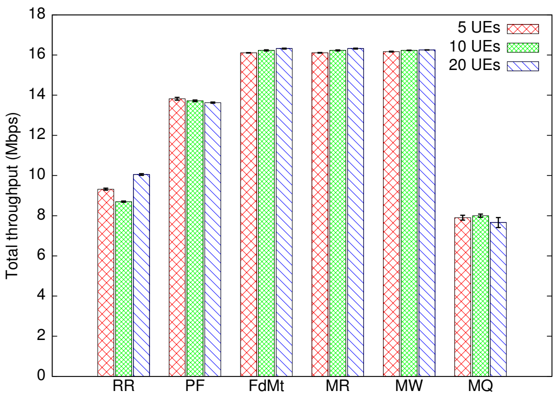
In this scenario, the remote host sends packets to UEs in a back-to-back fashion for the entire duration of the simulation. We are interested in understanding the effect of the number of users, buffer size, packet size and Internet delay on TCP and UDP performance under different schedulers.

1) Effect of Number of Users: In this subsection, we investigate how changing the number of UEs affect the performance of schedulers. Figs. 3(a) and 3(b) represent the total throughput for saturated TCP and UDP flows, respectively. In each figure, we compare the total throughput of different schedulers for 5, 10 and 20 UEs. As it can be seen in Fig. 3(a), MR and MW, have the highest total throughput among the simulated schedulers for TCP traffic.

An interesting observation is that by increasing the number of UEs, the total TCP throughput increases for all the schedulers. However, for UDP traffic (Fig. 3(b)) increasing the number of UEs has almost no effect on the total throughput of



(a) TCP flows. Two schedulers MR and MW have the highest total throughput. By increasing the number of UEs, the total throughput increases for all schedulers.

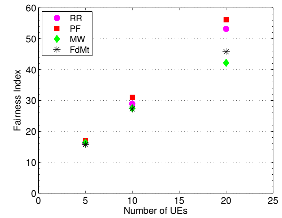


(b) UDP flows. FdMt, MR, and MW achieve the highest throughputs. Increasing the number of UEs has almost no effect on the total throughput for all schedulers.

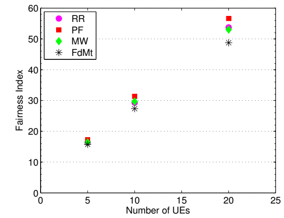
Fig. 3: Effect of the number of UEs on the total throughput of different schedulers for long-lived TCP and UDP flows.

the schedulers. The reason for this behavior is that with TCP, if there are only a few UEs in the cell, sometimes the scheduler may not be able to find a user that has both a good channel and data for transmission. With UDP, on the other hand, UEs always have data for transmission and the scheduler is able to find a set of UEs with good channel conditions in every time slot. We will come back to this problem when discussing the effect of buffer size on TCP throughput in subsection IV-B4.

In Figs. 4(a) and 4(b), proportional fairness indices of different schedulers are compared. As expected, for both TCP and UDP, PF yields the highest metric. As it can be seen, for small number of UEs, the indices are very close to each other, however, for higher number of UEs, their difference increases. In addition, generally, MW achieves a poor fairness performance compared to other schedulers. In Fig 5(a) and 5(b), Jain's fairness indices of the schedulers are compared. We observe that fairness indices of both RR and PF schedulers are very close to each other, while the other two schedulers yield much lower metrics. Again, MW has a low fairness performance, and its performance deteriorates as the number of UEs increases.

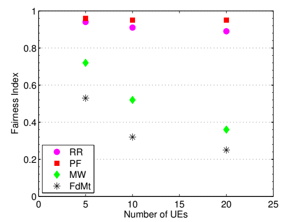


(a) TCP flows.

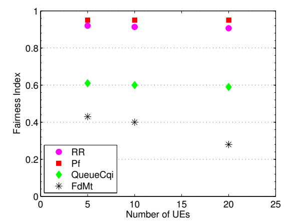


(b) UDP flows.

Fig. 4: Proportional fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF achieves the highest fairness. For small number of UEs, the indices are very close. In all experiments, MW achieves the lowest fairness index.



(a) TCP flows.



(b) UDP flows.

Fig. 5: Jain's fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF and RR achieve the highest fairness. In all experiments, MW achieves the lowest fairness index and its performance deteriorates as the number of UEs increases.

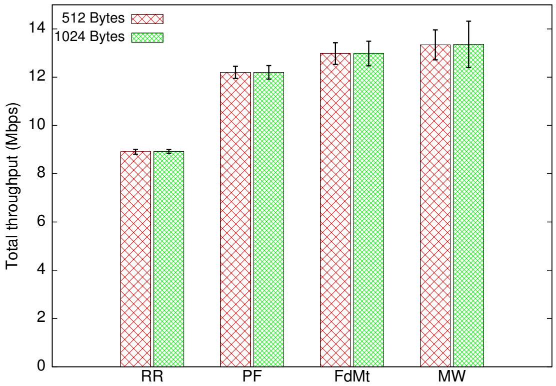
2) Effect of Packet Size: In this subsection, we investigate how changing the packet size affects the performance of different schedulers. The default packet size in our experiments is 1024 bytes. Figs. 6(a) and 6(b) compare the total throughput for packet sizes 512 bytes and 1024 bytes for TCP and UDP long-lived saturated flows, respectively. We observe that, regardless of the packet size, MW achieves the highest throughput. Moreover, changing the packet size has no noticeable effect on the TCP total throughput. On the other hand, in the case of UDP, the total throughput for 512 byte packets is slightly less than that of 1024 byte packets. This is expected as smaller packet size means higher header overhead and thus less throughput. However, in the case of TCP, larger packet size may lead to a higher packet loss probability which would negatively affect the TCP throughput. Let p_b and p denote the bit-error-rate (BER) and packet error probability. We have,

$$p = 1 - (1 - p_b)^L \approx 1 - Lp_b,$$

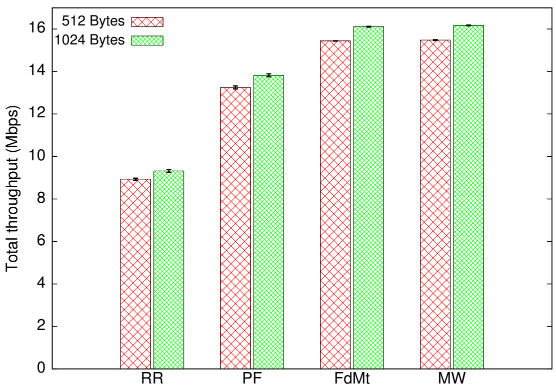
where L denotes the size of the packet. As can be seen, increasing the packet size leads to higher packet error probability. We *conjecture*[2] that in our experiments the throughput loss due to higher packet error probability cancels out the throughput gain due to a larger packet size.

3) Effect of Internet Delay: Internet delay has a significant effect on TCP performance as it affects the round-trip delay of TCP packets. We change the Internet delay from the default

[2]Further investigation is required to make a concrete conclusion. There could be other factors at the lower layers such as packetization, coding, etc that would affect this behavior.

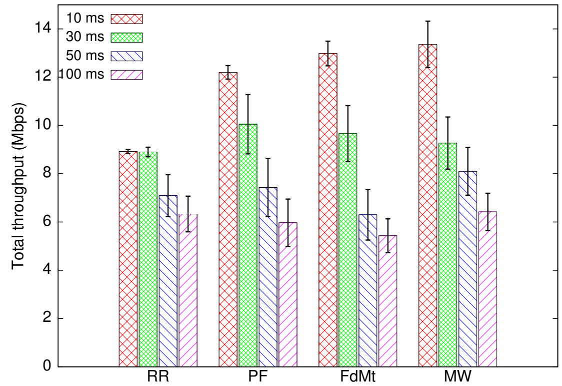


(a) TCP flows. Changing the packet size has no effect on the total throughput.

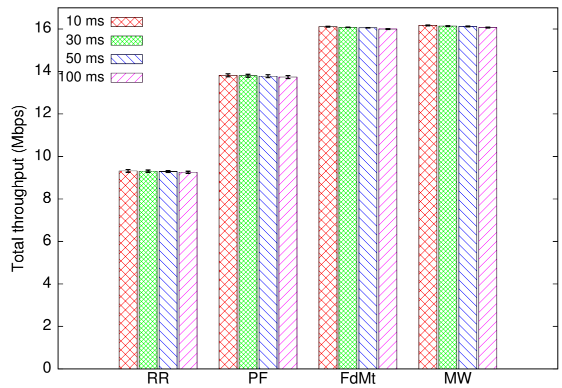


(b) UDP flows. The total throughput for 512 byte packets is slightly less than that for 1024 byte packets.

Fig. 6: Effect of packet size on the performance of different schedulers for long-lived TCP and UDP flows. Regardless of the packet size, MW achieves the highest throughput.



(a) TCP flows. Increasing network delay causes dramatic drop in total throughput for all of the schedulers.



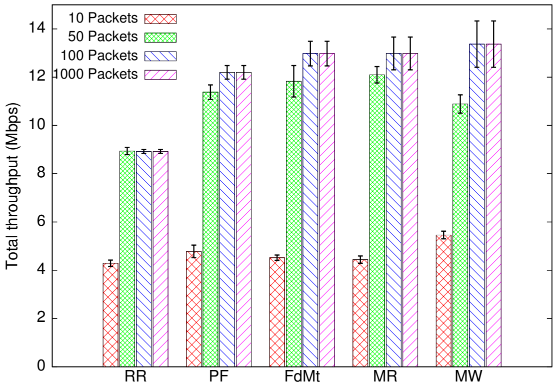
(b) UDP flows. Increasing network delay has no effect on total throughputs.

Fig. 7: Effect of Internet Delay on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput.

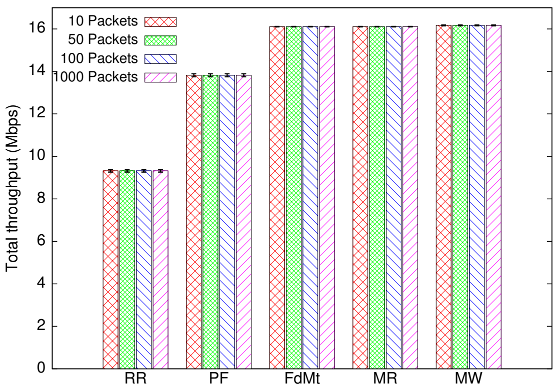
value of 10 ms to 30, 50, 100 ms and measure the total throughput. The results are depicted in Figs. 7(a) and 7(b). As shown in the figures, while increasing the Internet delay dramatically reduces TCP throughput, it has no effect on UDP throughput. The reason is that when a packet loss happens (*e.g.*, due to buffer overflow), it takes some time for TCP throughput to recover to its highest level, where the recovery time is directly proportional to TCP round-trip time. With UDP, however, the throughput is insensitive to packet losses and Internet delay as it continuously sends packets at a constant rate. Overall, MW achieves the highest throughput for the range of Internet delays considered in this experiment (MW and FdMt results are similar for UDP flows).

4) Effect of Buffer Size: The default buffer size in our experiments is set to 100 packets (*i.e.*, 100 KByte). Figs. 8(a) and 8(b) show the performance of different schedulers for a range of buffer sizes. As shown in the figures, increasing the buffer size has no effect on the UDP total throughput. The reason is that the links are in saturation state and thus the buffers are always full regardless of their size. On the other hand, in the case of TCP, as the buffer size increases so does the TCP throughput. The reason is that if there is a drop in TCP rate (*e.g.*, due to a packet loss), with a small buffer at eNB, there is not enough data at the buffer to saturate the wireless link. It is well-known that in order to maximize TCP throughput, a buffer of size equal to the delay-bandwidth product of the link is required. Increasing the buffer beyond the delay-bandwidth product does not result in considerable increase in throughput. As can be seen in Fig. 8(a), increasing the buffer size from 100 packets to 1000 packets does not result in any throughput improvement. Interestingly, MW achieves the highest throughput in all experiments.

To see how large a buffer is maintained, we have also computed the average buffer occupancy at the eNB using a moving average estimator. A very large queue would be indicative of TCP gaining throughput at the expense of large delays to the user. On the other hand, if the large buffer is used only for absorbing random fluctuations then the buffer occupancy should be low. Fig. 9 shows the moving averages for buffer occupancy (under TCP traffic) for UE 1, 2, 3 and 4 with MW for 100 packets buffer size and 50 ms Internet delay. As shown in the figures, the buffer occupancy of UE 1 suddenly drops and stays very low for a sustained period of time. Note that the plots show the moving averages and

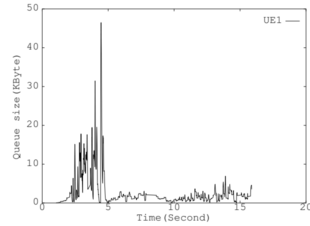


(a) TCP flows. TCP throughput increases by increasing the buffer size up to 100 packets. Increasing the buffer size to 1000 packets does not result in any improvement.

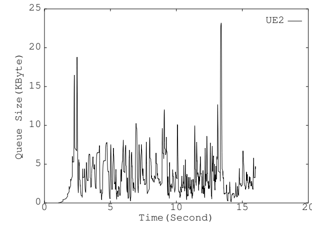


(b) UDP flows. Increasing the buffer size has no effect on the throughput as the links are in saturation state regardless of the buffer size.

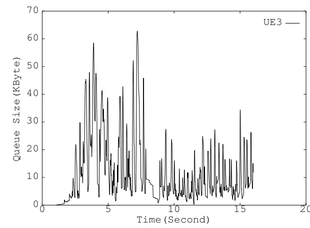
Fig. 8: Effect of Internet Delay on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput in all experiments.



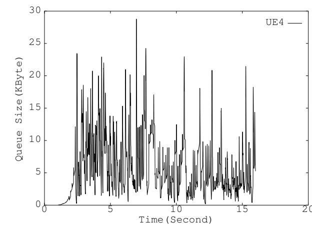
(a) UE 1



(b) UE 2



(c) UE 3



(d) UE 4

Fig. 9: Moving averages of buffer occupancy for UE 1, 2, 3 and 4 with MW for 100 packets buffer size and 50 ms Internet delay. UE 1 has a very low queue occupancy for a sustained period of time.

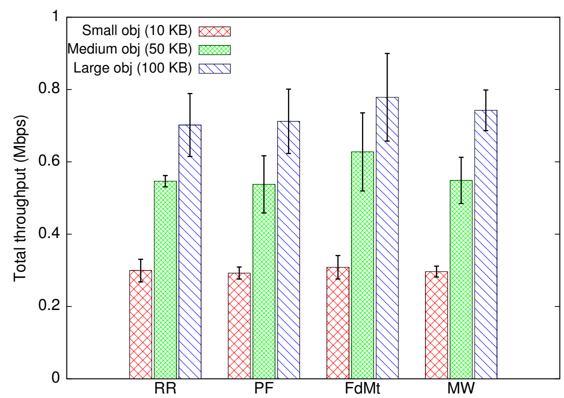


Fig. 10: Response time comparison for short-lived TCP flows. Overall, there is no significant difference between the response times of different schedulers.

not the instantaneous measurements. Having very low queue occupancy for some UEs can result in a throughput loss with TCP, as described earlier in subsection IV-B1. The throughput loss is more significant when there are only a few UEs in the system as some UEs with good channel may not have any data for transmission in the buffer. In this experiment, there were only 5 UEs in the system. Interestingly, a similar behavior was observed for other schedulers as well. This could explain the increase in throughput by increasing the number of UEs that was observed in Fig. 3(a).

C. Short-Lived TCP Flows

In this subsection, we study the performance of different schedulers for TCP short flows. Web traffic on mobile devices is composed of TCP short flows, where each flow is responsible for fetching an object from a remote web server. Unlike long flows which mainly require high throughput, the major performance bottleneck for short flows is latency or response time, that is, the time it takes to complete the flow (*i.e.*, download the object). To compare different schedulers, the response time for every UE is computed. To compute response time, we measure the time interval from receiving the first packet of a flow until receiving the last packet of the flow at the UE. The average of response times of UEs is the response time for the corresponding scheduler.

To determine the size distribution of short flows, we use the statistics reported in [17] for the popular Internet mail service `gmail.com`. Based on the measurements in this report, the average size of an HTTP response is 8 KByte. Therefore, in our experiments, we consider 3 web object sizes: small objects (10 KByte), medium objects (50 KByte) and large objects (100 KByte). In each experiment one short TCP flow is simulated and the maximum amount of data to send is set to one of the web object sizes.

Fig. 10 shows the response time of different schedulers for different web object sizes. We observe that the response times of RR, PF and MW are very close to each other while FdMt results in slightly longer response time. Overall, there is no significant difference among the simulated schedulers in this experiment.

V. TCP-Aware Scheduler

The TCP throughput plot (see Fig. 3(a)) reveals that there is a gap of about 10% in the total throughput between 5 and 10 user scenarios. However, this gap does not exist for UDP traffic. Note that in both scenarios (TCP and UDP), we are using the same wireless channel trace meaning that the gap can not be attributed to channel fluctuations. Given that MW selects the UE that has the highest product of achievable channel rate (based on the received channel quality indicator) and queue size, we conjecture that the gap is due to some UEs having no data in their buffer for transmission. In this case, the set of UEs that can be selected for transmission is small and hence there is less chance of scheduling users with the highest achievable rates, leading to a smaller throughput. Indeed, looking at a snapshot of buffer occupancy at eNB for MW (see Fig. 9) shows that there are times when the buffer occupancy is extremely low. This situation does not happen with UDP traffic.

To remedy this problem, we need to somehow make sure the UE with small queue gets timely ACKs to increase its sending rate and consequently build up its queue quickly. Therefore, we propose a dynamic scheduler called Q-MW that gives higher priority to UEs with small buffers. Specifically, with Q-MW, the UEs whose queues are below a threshold have higher priority and are scheduled based on their channel quality only. But UEs whose queue is larger than the queue threshold are scheduled using the MW scheduling policy.

A. Scheduling Algorithm

Q-MW divides UEs into two groups: (a) UEs whose queue is smaller than a threshold q, and (b) those whose queue is larger than q. The UEs in the first group have higher priority for scheduling over UEs in the other group. The priority metric used for group (a) is as same as MR scheduler while UEs in group (b) are scheduled using MW algorithm. The scheduler assigns RB k at TTI t to UE $\hat{i}_k(t)$, which is selected as follows:

$$\hat{i}_k(t) = \begin{cases} \arg\max\limits_{1\le j\le N} \; R_j(k,t) & \text{if } Q_j(t) \le q \\ \arg\max\limits_{1\le j\le N} \; Q_j(t)\cdot R_j(k,t) & \text{if } Q_j(t) > q \end{cases}$$

where q denotes the threshold on the queue size. The details of the scheduling algorithm are presented in Algorithm 1. Note that the threshold parameter q is an input to the algorithm. In the next subsection, we discuss how q can be selected for a given network configuration in order to improve the throughput of Q-MW.

B. Performance Evaluation

Fig. 11 represents the throughput gain of MW over PF for different buffer and delay values. As it can be seen, depending on network conditions, sometimes PF performs better than MW. This is contrary to the optimality of MW for inelastic (*e.g.*, UDP) traffic. Overall, the range of gains varies from -19% to 38%. The throughput gain is computed as follows:

$$\text{gain} = 100\% \times \frac{\text{Throughput}_{\text{MV}} - \text{Throughput}_{\text{PF}}}{\text{Throughput}_{\text{PF}}}\,.$$

Algorithm 1 Q-MW(q) Scheduler

```
▷ allocate all RBGs to flows
for each RBG i do
    MW_a ← 0
    MW_b ← 0
    flag ← false
    ▷ find the flow that has max weight for this RBG
    for each flow j do
        rate = AMC(flow j, RBG i)
        queue = QueueSize(flow j)
        if (queue > 0 AND queue ≤ q) then
            weight = rate
            if (weight > MW_a) then
                MW_a = weight
                mwFlow_a = j
            end if
            flag = true
        else
            weight = rate * queue
            if (weight > MW_b) then
                MW_b = weight
                mwFlow_b = j
            end if
        end if
    end for
    ▷ allocate the RBG to the max weight flow
    if (flag = true) then
        mwFlow = mwFlow_a
    else
        mwFlow = mwFlow_b
    end if
    allocate RBG i to flow mwFlow
end for
```

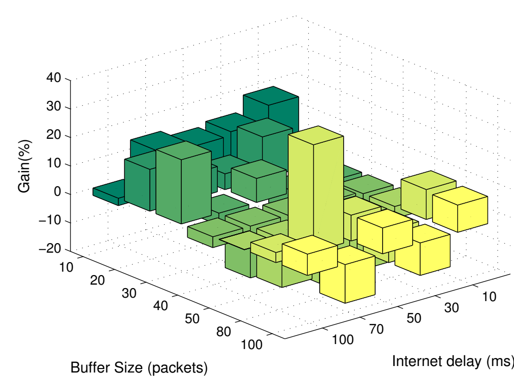
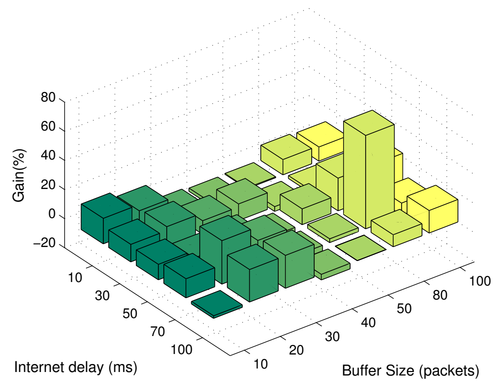


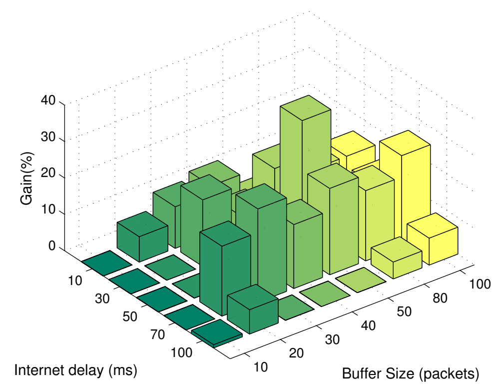
Fig. 11: Throughput gain of MW over PF for different combinations of buffer size and Internet delay. For some combinations, PF outperforms MW.

Next, we show the throughput gain of Q-MW over traditional MW and PF. We consider two scenarios where the parameter q is tuned dynamically and statically.

1) Dynamic Tuning: In this case, for each combination of buffer/delay, we chose the best threshold q that maximizes the throughput of Q-MW. Currently, we do not have a mechanism for automatically tuning Q-MW, and hence the tuning for these plots is performed manually using a small range of thresholds ($[1, 5]$). Fig. 12 shows the throughput gain of the



(a) The gain over PF ranges from -4% to 64%. The average gain is 11%.



(b) The gain over MW can be up to 37%. The average gain is 10%.

Fig. 12: Throughput gain of Q-MW over PF and MW for different combinations of buffer size and Internet delay.

new scheduler for various buffer and delay combinations with respect to traditional MW and PF. It is observed that Q-MW can achieve up to 64% and 37% throughput gain over PF and MW, respectively. On average, the throughput gains over PF and MW are 11% and 10%, respectively. Notice that our goal was to recover the 10% throughput loss of MW when the number of UEs is low.

2) Static Tuning: In this case, using off line experiments, we have chosen the thresholds $q = 2, 3, 4$ for eNB buffer sizes $100, 80, 50$ packets. These q values result in the highest average gain across the entire range of Internet delays. In practice, the eNB buffer size is a system parameter that is known in advance. Thus, the network operator can choose the best q value, for example, using a lookup table. However, once the q value is set for a buffer size then it does not change over time. In this case, we observed that that Q-MW can achieve up to 64% and 29% throughput gain over PF and MW, respectively. On average, the throughput gains over PF and MW are 6% and 7%, respectively. Thus, even a simple static tuning can recover most of the throughput loss of MW.

VI. Conclusion

In this work, we studied the performance of scheduling algorithms in LTE networks with TCP using *ns*-3 simulations and real channel traces. We particularly focused on MW scheduling as it has been shown in the literature to be throughput optimal for inelastic traffic. Our simulation results show that MW perform as well or better than other scheduling algorithms including Proportional Fair scheduling (PF). However, we observe that in small cells with only a few users, MW scheduling cannot achieve its full performance compared to larger cells with many users. Our experiments show that MW may result in small queues where some users do not have any data for transmission due to elastic TCP behavior. We then propose a new scheduler called Q-MW to remedy this problem. Simulation results show that Q-MW can achieve up to 64% and 37% throughput gain, respectively, over traditional PF and MW, if tuned properly. Currently, we do not have a mechanism for automatically tuning Q-MW. Designing a dynamic tuning algorithm for Q-MW is left for future work.

References

[1] F. Capozzi, G. Piro, and G. Boggia, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, 2013.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, 1992.

[3] M. Andrews *et al.*, "Providing quality of service over a shared wireless link," *IEEE Commun. Mag.*, vol. 39, no. 2, 2001.

[4] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.

[5] M. Andrews and L. Zhang, "Scheduling algorithms for multicarrier wireless data systems," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, 1992.

[6] L. B. Le, K. Jagannathan, and E. Modiano, "Delay analysis of maximum weight scheduling in wireless ad hoc networks," in *Proc. CISS*, Mar. 2009.

[7] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, 2007.

[8] B. Sardar and D. Saha, "A survey of TCP enhancements for last-hop wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 3, 2006.

[9] M. C. Chan and R. Ramjee, "Improving TCP/IP performance over third-generation wireless networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 4, 2008.

[10] G. Piro, N. Baldo, and M. Miozzo, "An LTE module for the ns-3 network simulator," in *Proc. SIMUTools*, Mar. 2011.

[11] Small Cell Forum. [Online]. Available: http://www.smallcellforum.org/

[12] tutorialspoint. [Online]. Available: http://www.tutorialspoint.com/lte/lte_quick_guide.htm

[13] C. T. de Telecomunicacions de Catalunya (CTTC), "LTE simulator documentation release M6," Apr. 2013.

[14] AT&T, "Network latency report," Oct. 2013. [Online]. Available: http://ipnetwork.bgtmo.ip.att.net/pws/global_network_avgs.html

[15] M. Uchida and J. Kurose, "An information-theoretic characterization of weighted α-proportional fairness," in *Proc. IEEE Infocom*, Apr. 2009.

[16] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.

[17] "Web metrics: Size and number of resources," Oct. 2013. [Online]. Available: https://developers.google.com/speed/articles/web-metrics