Managing Background Traffic in Cellular Networks

Shanyu Zhou*, Muhammad Usama Chaudhry*, Vijay Gopalakrishnan†, Emir Halepovic†,
Balajee Vamanan*, and Hulya Seferoglu*
*University of Illinois at Chicago; {szhou45, mchaud30, bvamanan, hulya}@uic.edu
†AT&T Labs – Research; {gvijay, emir}@research.att.com

***Abstract*—A large variety of traffic — time-sensitive "foreground" traffic (e.g., web browsing) and time-insensitive "background" traffic (e.g., software updates) — compete for the scarce cellular bandwidth, especially on the downlink. While there is limited in-network support for traffic prioritization, existing end-to-end, "low priority transport protocols" exhibit sub-optimal performance in cellular networks. We propose *Sneaker*, which yields to time-sensitive foreground traffic during periods of congestion and enables time-insensitive background traffic to efficiently utilize any spare capacity. *Sneaker* achieves the desired goal by randomly dropping packets coming into the base station, based on traffic type and network conditions. Our *key* contribution is the derivation of the *optimal* dropping rate and a *practical* dropping rate, which performs close to optimal. Further, *Sneaker* co-exists and performs well with existing cellular schedulers and transport protocols.**

Keywords— Cellular networks; Proportional fair scheduler; LTE; Background traffic; Low priority transport

I. Introduction

In this era of ubiquitous cellular network connectivity, management of the scarce cellular bandwidth is important. A number of new applications (e.g., software updates, cloud sync) are starting to compete with existing applications (e.g., web browsing, video streaming) for cellular bandwidth. Therefore, it is critical to develop traffic management mechanisms that can handle large volumes of traffic with diverse characteristics.

One way to tackle the problem of managing such diverse traffic is to partition flows into two classes: *foreground* and *background* flows (although it is straightforward to extend the approach to more than two classes). In this paper, we broadly consider regular mobile traffic like human communication or even some time-sensitive machine-to-machine (M2M) flows (e.g., alarms, health monitoring) as foreground traffic, while treating large volume, time insensitive flows (e.g., software downloads) as background traffic. Our goal then is to design a traffic management approach for cellular networks that prioritizes foreground traffic during periods of congestion while scheduling background traffic to effectively utilize any spare capacity. We also focus on a solution for downlink because most large volume traffic flows in the downlink direction.

There are existing in-network and end-to-end approaches to manage foreground and background traffic. Today's cellular networks provide support for prioritization using QoS Class Identifier (QCI). However, QCI suffers from several limitations: (1) identifies only at the granularity of devices or radio bearers, (2) offers only a handful traffic classes, many of which are reserved for operator-provided services like voice and VoLTE, and (3) uses static weights for the classes independent of network load. As an alternative, end-to-end, low-priority transport protocols (e.g.,TCP-LP [1]) perform well in wired and Wi-Fi networks but suffer in cellular networks due to per-device queues and schedulers as demonstrated in Section VI. To summarize, existing in-network approaches do not prioritize at flow granularity and existing end-to-end approaches are not effective in cellular networks.

To address the challenge of managing large-volume background traffic, we propose *Sneaker*, which achieves high network utilization and high throughput for background flows, without affecting foreground flows. Flows are classified as foreground or background. This classification can be independent of applications running on- or off-screen at user devices, and can be performed by either end hosts, network or content providers. However, the mechanism for classification of traffic into foreground and background classes is beyond the scope of our work and not something we focus on. *Sneaker* ensures that background flows quickly yield to foreground flows when the network is congested and quickly recapture spare capacity when the network becomes lightly loaded. Further, *Sneaker* seamlessly co-exists with existing schedulers and end-to-end protocols. Our *key* insight is that we can achieve the appropriate prioritization by randomly dropping background flow packets based on network load. Although our design is inspired by Random Early Detection (RED) [2], our goals and mechanisms are different from those of RED; *Sneaker* drops packets from queues, not based on overall congestion, but the load of foreground flows.

Our key design goals are to co-exist with end-to-end protocols (i.e., TCP), without requiring changes to existing schedulers. To achieve these goals, we first study the interaction of TCP with common cellular schedulers. Then, we formulate the problem as a Network Utility Maximization (NUM) problem to determine the *optimal* transmission rate of background flows. Using this optimal transmission rate, we derive an optimal dropping rate for background flows. Because the optimal dropping rate is hard to realize in practice, we identify a close approximation to the optimal rate, which is easy to implement and works in harmony with end-to-end protocols. We show that our *practical* dropping rate avoids TCP timeouts for background flows and achieves the intended prioritization of foreground flows. Extensive ns-3 simulations confirm our analysis and show that *Sneaker* outperforms an aggressive baseline that gives strict priority to foreground

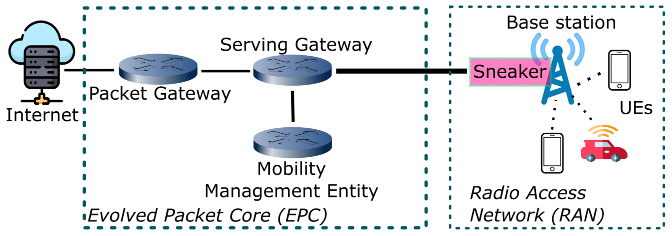


Figure 1: The service architecture in LTE network

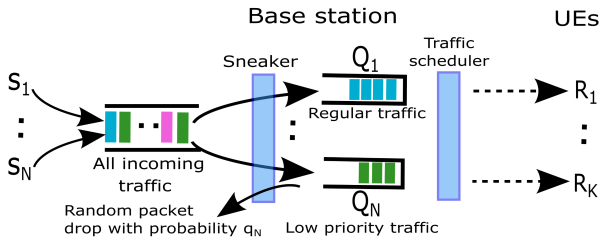


Figure 2: Example cellular system setup with *Sneaker*

traffic. Further, we also show that *Sneaker* performs better than existing low priority transport protocols. In summary, we make the following contributions:

- We analyze the complex interaction between TCP and cellular schedulers and characterize the average throughput of TCP for common cellular schedulers.
- We derive a dropping rate that satisfies our network objective of maximizing the performance of foreground traffic and **prove that the rate is optimal**.
- We develop a **practical dropping rate** that achieves prioritization and fairness among flows.
- We design *Sneaker* using the practical dropping rate and evaluate it via simulations in ns-3. We show that *Sneaker* significantly improves over in-network and end-to-end baselines in terms of prioritizing foreground flows over background flows and efficiently utilizing any spare capacity for background flows.

The rest of the paper is organized as follows. We start with a brief background and problem statement (Section II). We then analyze the interaction between TCP and common base station schedulers (Section III) and derive the optimal and practical dropping rates (Section IV). Next, we discuss our implementation (Section V) and present evaluation results (Section VI). We finally conclude with closing remarks (Section VIII).

II. System Model

System Overview. We consider the general architecture of LTE cellular networks as shown in Fig. 1, with two parts; evolved packet core (EPC) and radio access networks (RAN). The evolved packet core is a high-speed wired network that comprises the Mobility Management Entity (MME), the Serving Gateway (SGW), and the Packet Data Network Gateway (PGW). The LTE RAN includes the eNobeB (base station) and User Equipment (UE), which could be cellphones, tablets, connected vehicles, etc. Traffic from remote hosts in the Internet traverses through the packet core, arrives at the base station and eventually reaches the end users through wireless channels. In this setup, we develop *Sneaker* at base stations that works with existing TCP and cellular scheduling protocols and prioritizes foreground traffic over background.

Flows. We consider a system model depicted in Figure 2, which represents the RAN from Figure 1. There are N flows destined to K users within the same radio cell. M of these flows are foreground, while L of them are background flows (i.e., $N = M + L$). The set of all flows is $\mathcal{S} = \{S_1, \ldots, S_N\}$.

Queuing Model. At the base station, packets are queued in per-flow queues; $\{Q_1, \ldots, Q_N\}$. Packets from flow S_n are stored in queue Q_n. These queues operate according to the First-Come First-Serve (FCFS) rule.

Channel Model. We consider that base station back-haul links are high speed and lossless. The bottleneck of the system is the last hop radio channel; a radio frequency carrier shared by a set of cellular devices. We consider that $c_n(t)$ denotes the downlink channel capacity of device n at time t, which is the maximum achievable data rate based on the channel characteristics, as determined by the base station.

Scheduler. At the base station, time is divided into Transmission Time Intervals (TTIs), and each TTI is usually 1 ms in LTE [3]. The scheduler determines which packets should be transmitted from the base station at a given TTI. Proportional Fair scheduler (PFS) is one of the widely used schedulers in today's cellular systems [4]. The other popular schedulers are Maximum Throughput (MT), Round Robin (RR), Blind Equal Throughput (BET), Throughput to Average (TTA), etc [5].

Problem Statement. Our goal is to develop a method that works with existing transport layer protocols and scheduling algorithms to prioritize foreground traffic over background. Our approach is to achieve the desired goal by randomly dropping packets coming into the base station, based on traffic type – foreground or background. In this context, the fundamental problem is to determine the optimal dropping rate. In this paper, we determine the optimal dropping rate that forces background traffic to quickly yield to foreground traffic when the network is congested, but allows it to quickly recapture spare capacity when network load subsides.

III. Interaction of TCP with Cellular Networks

In this section, we characterize the average TCP sending rate in cellular networks.

Let the congestion window size of flow destined to user n at time slot t be $W_n(t)$. We assume that round-trip time (RTT) of each packet is constant, and equals to T_n. This is a common assumption in classical TCP analysis [6, 7]. Let $q_n(t)$ be the probability that packets from flow n are dropped from buffers due to overflow in the core network as well as the base station. Let $\rho_n(t)$ be the probability that packets from flow n are scheduled to be transmitted from the base station according to the underlying scheduling algorithm.

In this analysis, we ignore the slow start and time-out phases and only focus on the congestion avoidance phase of TCP, since the duration of congestion avoidance phase takes most of the TCP flow's lifetime. In congestion avoidance phase, at time $t - T_n$, $W_n(t - T_n)$ packets are transmitted from the source of TCP flow n. The ACKs corresponding to these packets, received between t and $t+T_n$, determine the window size update. In particular, for each transmitted and ACKed packet, window size is increased by $1/W_n$. For each packet dropped due to buffer overflow or delayed at the base station due to congestion, window size is reduced by $W_n(t)\beta$, where $0 < \beta < 1$. Thus, the window size evolves as follows: $W_n(t + T_n) = W_n(t) + I_n(t) - D_n(t)$, where $I_n(t) = W_n(t - T_n)\frac{1}{W_n(t)}(1 - q_n(t))\rho_n(t)$ is the increase in window size, and $D_n(t) = W_n(t - T_n)\beta W_n(t)(1 - (1 - q_n(t))\rho_n(t))$ is the decrease in window size.

The differential of the window size at time slot t is $\dot{W}_n(t) = (W_n(t + T_n) - W_n(t))/T_n$. The steady-state window size that satisfies $\dot{W}_n = 0$ becomes $W_n = \sqrt{\frac{(1-q_n)\rho_n}{\beta(1-(1-q_n)\rho_n)}}$. Thus, the steady-state TCP rate is formulated as

$$x_n^{TCP} = \frac{W_n B}{T_n} = \frac{B}{T_n}\sqrt{\frac{(1 - q_n)\rho_n}{\beta(1 - (1 - q_n)\rho_n)}}, \qquad (1)$$

where B is the typical TCP packet size. TCP rate x_n^{TCP} depends on RTT, the scheduling probability at the base station, and the packet dropping probability in end-to-end path. As the bottleneck is usually the radio interface, we assume that packet drops (with probability q_n) only happen at the base station.

Our goal in this work is to determine the dropping rate q_n and actively drop packets from the queues (according to q_n) to prioritize foreground traffic over background. We characterize the optimal value of q_n in the next section.

IV. Design of *Sneaker*

In this section, we derive the dropping probability for background traffic so that in the case of congestion, it does not compete with foreground traffic while avoiding costly timeouts, which hurt throughput.

A. NUM Formulation when Foreground and Background Flows Coexist

First, we formulate a network utility maximization (NUM) problem when foreground and background flows coexist. Let $\mathcal{L}$ and $\mathcal{M}$ denote the sets of background and foreground flows, respectively. We assume there are L and M background and foreground flows in the network. Let x_l and x_m denote the rate of the background and foreground flows, respectively. And let c_l, c_m denote the channel capacity of background flow user l and foreground flow user m, respectively. We can formulate the following NUM problem,

$$\begin{aligned} \max_{[x_1,\ldots,x_L]} \quad & \sum_{l=1}^{L} U_l(x_l) \\ \text{s.t.} \quad & \sum_{m\in\mathcal{M}} \frac{x_m}{c_m} + \sum_{l\in\mathcal{L}} \frac{x_l}{c_l} \leq 1 \\ & x_l \geq 0, \forall l \in \mathcal{L} \end{aligned} \qquad (2)$$

where $U_l(\cdot)$ is the utility function associated with background flow l. The NUM formulation in Eq. (2) optimizes the total utility of background flows assuming that there exist foreground flows. The first constraint is the time sharing constraint across foreground and background traffic flows. In this problem, we do not control the data rate x_m of regular flows. That is to say, x_m is given (i.e., not an optimization parameter), which is controlled by end-to-end TCP congestion control mechanism. Note that we assume that $\sum_{m\in\mathcal{M}} \frac{x_m}{c_m} \leq 1$ since the data rate controlled by TCP will not exceed the capacity on average.

Theorem 1: Assuming that we use log-based utility function (i.e., $U_l(x_l) = \log(x_l)$), which is widely used to provide proportional rate fairness, the optimal solution to the NUM problem in Eq. (2) is expressed as

$$x_l^{OPT} = \frac{c_l}{L}[1 - \sum_{m\in\mathcal{M}} \frac{x_m}{c_m}], \forall l \in \mathcal{L} \qquad (3)$$

where x_l^{OPT} depends on its channel capacity c_l, the number of background flows L, and the occupancy of the air interface by foreground traffic $\sum_{m\in\mathcal{M}}(x_m/c_m)$.

Proof: The proof directly follows from the KKT conditions [8]. We omit the detailed proof due to space constraints. □

B. Optimal Dropping Rate

Now that we characterized the optimal and TCP rates of background flows (i.e., x_l^{OPT} in Eq. (3) and x_n^{TCP} in Eq. (1)), we can design *Sneaker* by pushing the TCP rate to the optimal rate. The optimal dropping probability q_l^{OPT} that satisfies $x_l^{TCP} = x_l^{OPT}$ is expressed as

$$q_l^{OPT} = 1 - \frac{\gamma(c_l T_l(1 - \tau_F))^2}{\rho_l(1 + \gamma(c_l T_l(1 - \tau_F))^2)}, \forall l \in \mathcal{L}, \qquad (4)$$

where $\tau_F = \sum_{m\in\mathcal{M}}(x_m/c_m)$, and $\gamma = \beta/(L^2B^2)$. Therefore, q_l^{OPT} depends on channel capacity c_l, RTT T_l, the amount of foreground traffic $\tau_F = \sum_{m\in\mathcal{M}} x_m/c_m$, and the number of background flows L. Eq. (4) also depends on ρ_l, the packet scheduling probability at the base station.

When foreground traffic congests the network, i.e., τ_F approaches 1, and q_l^{OPT} approaches 1. This means that every packet from background flows would be dropped at the base station, which would cause TCP timeouts, and eventually *stop* the transmission. Therefore, we develop a practical dropping probability in the next section, which yields to foreground traffic, yet prevents background flows from timing out.

C. Practical Dropping Rate

We develop a practical background flow rate based on the structure of the optimal rate in Eq. (3) as $x_l^{PR} = \frac{c_l}{L}\max\{1 - \tau_F, \epsilon\}$, where ϵ $(0 < \epsilon < 1)$ is the minimum rate that should be allocated to background flows to keep them *alive*.

Similar to the optimal packet dropping rate, we set $x_l^{TCP} = x_l^{PR}$ and determine the practical drop rate q_l^{PR} as

$$q_l^{PR} = 1 - \frac{\gamma(c_l T_l \max\{1 - \tau_F, \epsilon\})^2}{\rho_l(1 + \gamma(c_l T_l \max\{1 - \tau_F, \epsilon\})^2)}, \forall l \in \mathcal{L} \qquad (5)$$

From Eq. (5), the largest dropping rate for background flow l is $q_l^{PR} = 1 - \frac{\gamma(c_l T_l \epsilon)^2}{\rho_l(1+\gamma(c_l T_l \epsilon)^2)} \leq 1$, which happens when $1 - \tau_F \leq \epsilon$. Thus, even when foreground traffic is high, the background flows still get some resources for transmitting their packets. By tuning ϵ, we can adjust how much resources should be allocated to background traffic, hence aggressiveness of the background flows.

V. Implementation of *Sneaker*

A. *Design parameters and signalling for Sneaker*

1) Scheduling probability: The packet dropping probability in Eq. (5) is a function of ρ_l, which is the packet scheduling probability at the base station and depends on the scheduling algorithm. The packet scheduling probability can be measured by the scheduler and passed to *Sneaker* to calculate the packet dropping probability. Furthermore, we can characterize the scheduling probability analytically for common schedulers. For example, the scheduling probability is expressed as

$$\rho_l = 1 - (1 - \frac{1}{N})^{\mathrm{RTT/TTI}} \tag{6}$$

for PFS. (Note that we omit mathematical calculations due to space constraints.) Similar analysis can be performed for other schedulers including MT, RR, BET, and TTA.

2) Local signalling: The packet dropping rate in Eq. (5) depends on scheduling parameters such as the time share of foreground traffic τ_F, capacity of background user c_l, and the number of background flows L.

We approximate the time share of foreground traffic as $\tau_F \approx \sum_{m\in\mathcal{M}}(Q_m(t)/R_m(t))$, where $Q_m(t)$ is the queue size of foreground user m in the base station at TTI t, and $R_m(t)$ is the maximum number of packets that can be transmitted from foreground flow user m at TTI t. The main idea behind this approximation is that both x_m/c_m and $Q_m(t)/R_m(t)$ are time shares, and we conjecture that the time average of $Q_m(t)/R_m(t)$ approaches x_m/c_m. $Q_m(t)$ and $R_m(t)$ are collected by existing scheduling algorithms (such as PFS) and passed to *Sneaker*. Similarly, the scheduler passes $R_l(t)$ information to *Sneaker*, and we make $c_l \approx R_l(t)$ approximation.

3) End-to-end signalling: As for the access of other parameters, in our implementation, sender adds one bit as a tag in its TCP header to mark its classification (foreground or background). Sender also piggybacks the RTT information into its TCP header. At the base station, *Sneaker* extracts TCP header to obtain RTT and classification of the flow. In this way, *Sneaker* obtains T_l and learns if a flow is a background flow (hence the total number of background flows L).

B. *Implementation of Sneaker on ns-3*

We implemented and evaluated *Sneaker* using the ns-3 simulator [9]. We build on top of existing LTE protocol stack shown in Fig. 3, where data packets are buffered in Radio Link Control (RLC) layer after passing through Packet Data Convergence Protocol (PDCP) layer. MAC layer reads packets in RLC buffers depending on the scheduling algorithm used.

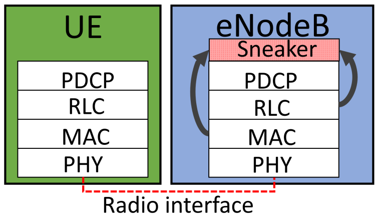


Figure 3: LTE protocol stack

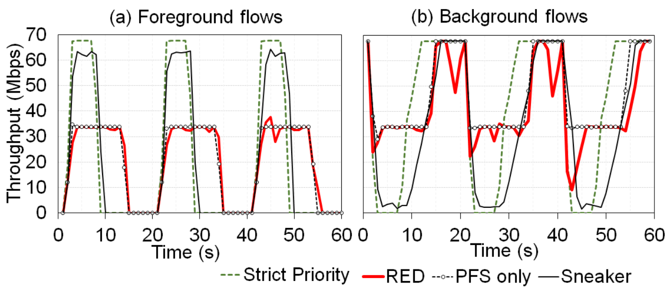


Figure 4: Comparison of different approaches to prioritization

Sneaker, implemented at eNodeB as a slim layer on top of PDCP, inspects every incoming packet, and extracts end-to-end information. It also gets local signalling data from RLC and MAC layers to calculate the packet dropping probability using Eq. (5). Packets are dropped by *Sneaker* according to the calculated packet drop probability before arriving to PDCP. Note that we also make minimal updates to TCP so that packets carry end-to-end information.

VI. Evaluation of *Sneaker*

We evaluate the performance of *Sneaker* through ns-3 simulations. Our simulation topology is shown in Fig. 1. It consists of multi-hop wired links that connect remote servers to Packet Gateway using 1 Gbps, 10 ms delay links. The link speed between the base station and packet and service gateways is 300 Mbps. The base station is configured with 751 MHz downlink band with 10 MHz bandwidth and 50 resource blocks, MIMO transmission, transmission power of 47.78 dBm, and RLC buffer size is 512 KB. The path loss model is log distance propagation model with loss exponential parameter of 3.52.

Prioritization. In this experiment, we verify if *Sneaker* achieves the correct prioritization between background and foreground flows. We use two remote senders to send traffic to two different end users. While the first sender continuously sends background traffic to a user, the second sender sends foreground traffic in an on-off pattern every 10 seconds. All senders share a cellular link with 70 Mbps and use TCP Reno. We compare *Sneaker* with the following baselines; `PFS only`, `RED`, `Strict Priority`. All these mechanisms (including *Sneaker*) use TCP-Reno and Proportional Fair Scheduler (PFS). *Sneaker* drops packets according to Eq. (5), `PFS only` does not drop packets actively, `RED` drops packets according to the policy in [10], and `Strict Priority` drops every packet from background flow if there exists foreground traffic.

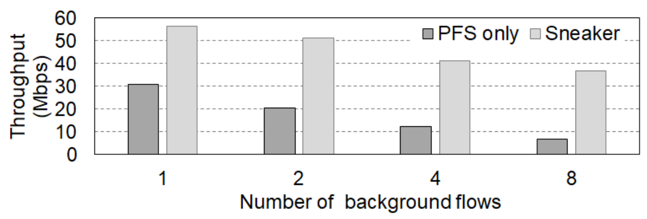
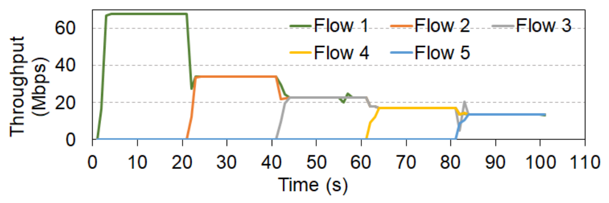
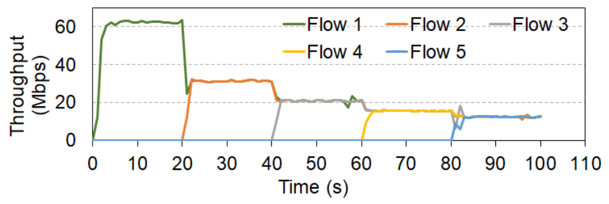


Figure 5: Foreground flows benefit from *Sneaker*



(a) Fairness among background flows



(b) Fairness among foreground flows

Figure 6: Fairness among background and foreground flows

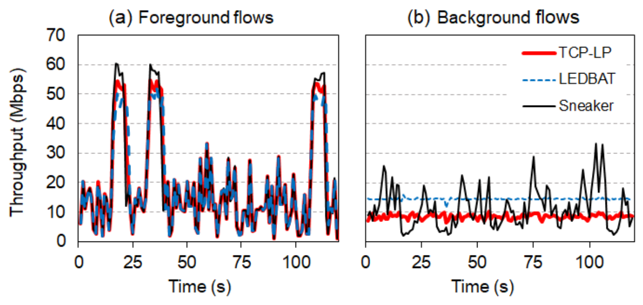


Figure 7: Large-scale results at moderate load (50%)

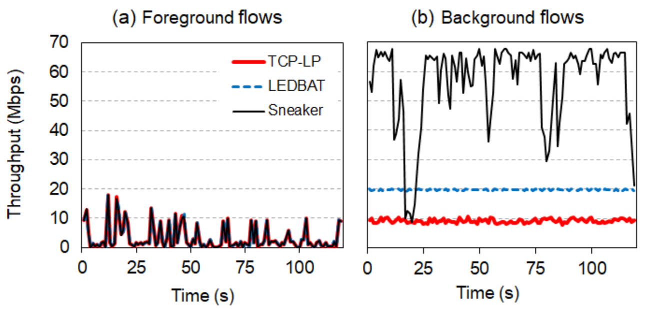


Figure 8: Large-scale results at low load (20%)

Fig. 4 shows the throughput achieved by both foreground and background flows. *Sneaker* and `Strict Priority` work as expected, i.e., yield to foreground traffic, and serve background traffic if there is no foreground traffic. On the other hand, `PFS only` and `RED` do not provide such ability. We prefer *Sneaker* over `Strict Priority` as `Strict Priority` completely blocks background traffic until foreground traffic finishes and causes timeouts and connection disconnects, which is not ideal.

We also study the impact of increasing number of background flows on foreground traffic rate. In this experiment, we send a foreground flow to one user. Simultaneously, we have a different number of background flows to another user connected to the same base station. We show the average throughput achieved by the foreground flow with `PFS only` and *Sneaker* in Fig. 5. As shown, the throughput of foreground flows degrades as the number of background flows increases, as expected. However, the degradation is significantly smaller with *Sneaker* than it is with `PFS only`, which shows that *Sneaker* is able to better isolate foreground flows. With *Sneaker*, the foreground flow achieves 1.8 times higher throughput than with `PFS only`; with *five-fold* throughput gain as the number of background flows increases to 8.

Fairness. We want to verify: *fair sharing of spare bandwidth among background flows*, and *fair sharing of available bandwidth among foreground flows*. We consider two scenarios: (i) five background flows without any foreground flow, and (ii) five foreground flows with one background flow. In both scenarios, the new flows join every 20 seconds. Figure 6(a) shows the throughput achieved by background flows using *Sneaker* in the first scenario. We see that the background flows quickly *converge* to their fair-share throughput as new flows are added. Figure 6(b) shows that rate is fairly shared by foreground flows in the second scenario using *Sneaker*.

*Large-scale simulations.*We consider realistic workloads and create a large topology with 100 users that connect to a base station. We generate a mixed traffic with short (64 KB), medium (1 MB), and long flows (32 MB). Short flows generate 10% of the overall foreground traffic, whereas medium flows generate 40% of the load; the rest (i.e., 50% of load) comes from long flows. While we generate foreground traffic mix, we send a continuous background (long) flow. We compare *Sneaker* (with background flow using TCP-Reno) with `TCP-LP` and `LEDBAT`, the known end-to-end congestion control mechanisms designed to yield to foreground traffic. We simulate traffic for 120 seconds and compare the throughput achieved by foreground and background flows in all the schemes, with `PFS` as the default scheduler.

We first evaluate the performance of the three schemes under moderate load of 50%. For this study, we set the overall network load due to foreground flows to 50%. Fig. 7 shows the throughput of foreground and background flows. We observe the bursty nature of foreground traffic under all schemes, as expected. For foreground flows, `TCP-LP` achieves better throughput than `LEDBAT`, and *Sneaker* achieves the best throughput. On the other hand, the throughputs achieved by background flows drastically differ between the three schemes. `LEDBAT` and `TCP-LP` does not yield to foreground flows, which is not desirable. In contrast, *Sneaker* modulates its throughput to allow foreground traffic to use most of the available capacity while effectively utilizing the spare capacity.

We consider the same experiment under low load (20%).

Fig. 8 shows that all the schemes achieve similar throughput for foreground flows, as expected. However, their background flow rates differ significantly. While `TCP-LP` and `LEDBAT` are unable to fully utilize the spare capacity at low loads (i.e., they achieve lower throughput), *Sneaker* efficiently utilizes the spare capacity and achieves much higher throughput.

VII. Related Work

Our work is closely related to the ideas from active queue management (AQM), interaction of TCP with cellular network and low priority data transmission.

Active queue management (AQM): AQM is a common approach to control data transmission rate in order to avoid congestion and improve network performance. One of the best known AQM schemes is Random Early Detection (RED) [11], which controls congestion by randomly dropping packets in the queue based on the average queue size information. This idea generated tremendous interest in congestion control and a lot of work has been done to improve the performance of RED based on local information such as queue dynamics and packet loss [12–15]. While our approach seems similar to RED, there are important differences. While the goal of RED is to manage congestion across all flows, ours is to achieve differentiation between classes of flows.

Interaction of TCP with cellular network: TCP is not designed to work in cellular networks [16]. Thus the interaction of TCP with cellular network needs to be explored to better understand the performance of TCP flows. Due to highly variable delays on wireless links, spurious timeouts also occur in cellular network, which causes unnecessary re-transmissions and decreases throughput [17–20]. The scheduling algorithm in the base station also affects performance of TCP [21]. Compared to these works, our focus is to characterize the data rate of TCP by taking into account the specific factors in cellular networks and formulate the desired dropping rate at the base station based on the desired TCP data rate.

Low priority transport protocols: Researchers have studied several low priority transport protocols such as LEDBAT [22], TCP-LP [1] and TCP-Nice [23]. LEDBAT [22] and TCP-LP [1] use one-way delay as the congestion indicator and adjust congestion control accordingly. TCP-Nice uses RTT-threshold-based mechanism to indicate congestion [23]. The key idea of the current low-priority protocols is to detect congestion earlier than regular TCP. Compared to these works, the focus of our work is not to design an end-to-end low priority transport protocol, but to design a packet dropping policy in queues to enable per-flow differentiation.

VIII. Conclusion

Existing in-network and end-to-end mechanisms for per-flow prioritization do not work well over scheduled links in cellular networks. We presented *Sneaker*, an in-network arbiter that enables differentiation between foreground and background flows, while providing high performance and maintaining fairness within each traffic class. We have formulated the problem of per-flow prioritization using NUM framework and shown that *Sneaker* achieves the desired optimality. Further, we have extensively evaluated our design using both targeted small-scale and realistic large-scale simulations. We plan to analyze mobility and signal quality variation in future work.

Acknowledgments

This work was supported in part by UIC seed grant, NSF grant CNS-1801708, and the ARL grant W911NF-1820181.

References

[1] A. Kuzmanovic and E. W. Knightly, "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM ToN*, vol. 14, 2006.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM ToN*, no. 4, pp. 397–413, 1993.

[3] B. Holfeld, D. Wieruch, T. Wirth, L. Thiele, S. A. Ashraf, J. Huschke, I. Aktas, and J. Ansari, "Wireless communication for factory automation: an opportunity for LTE and 5G systems," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 36–43, 2016.

[4] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks." in *NSDI*, 2013, pp. 459–471.

[5] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 678–700, 2013.

[6] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *ToN*, vol. 8, no. 2, pp. 133–145, 2000.

[7] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM ToN*, vol. 11, no. 4, pp. 525–536, 2003.

[8] H. Kuhn and A. Tucker, "Nonlinear programming," in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, 1951, pp. 481–492.

[9] "NS-3 network simulator," http://www.nsnam.org/.

[10] P. Kuusela, P. Lassila, J. Virtamo, and P. Key, "Modeling RED with idealized TCP sources," *Proceedings of IFIP ATM & IP*, 2001.

[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM ToN*, vol. 1, no. 4, 1993.

[12] T. J. Ott, T. Lakshman, and L. H. Wong, "SRED: stabilized RED," in *IEEE INFOCOM'99*, vol. 3, 1999, pp. 1346–1355.

[13] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *ACM SIGCOMM Comp. Comm. Review*, vol. 31, no. 4. ACM, 2001.

[14] W.-c. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," *IEEE/ACM Transactions on Networking (ToN)*, vol. 10, no. 4, pp. 513–528, 2002.

[15] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe – A stateless active queue management scheme for approximating fair bandwidth allocation," in *IEEE INFOCOM 2000*, vol. 2, 2000, pp. 942–951.

[16] J. Wang, A. Huang, W. Wang, Z. Zhang, and V. K. N. Lau, "On the transmission opportunity and TCP throughput in cognitive radio networks," *Int. J. Commun. Syst.*, vol. 27, no. 2, May 2012.

[17] A. Gurtov and R. Ludwig, "Responding to spurious timeouts in TCP," in *Proc. of IEEE INFOCOM*, vol. 3, 2003, pp. 2312–2322.

[18] R. Ludwig and R. Katz, "The eifel algorithm: Making TCP robust against spurious retransmissions," *ACM Computer Communication Review*, vol. 30, pp. 30–36, January 2000.

[19] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang, "Experiences in a 3G network: Interplay between the wireless channel and applications," in *MOBICOM*. ACM, 2008, pp. 211–222.

[20] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis, "CQIC: Revisiting cross-layer congestion control for cellular networks," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 45–50.

[21] T. E. Klein, K. K. Leung, and H. Zheng, "Enhanced scheduling algorithms for improved TCP performance in wireless IP networks," in *Proc. of IEEE Globecom*, vol. 4, 2004, pp. 2744–2759.

[22] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "LEDBAT: The new bittorrent congestion control protocol." in *ICCCN*, 2010, pp. 1–6.

[23] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 329–343, 2002.