Performance Analysis of TCP and UDP-based Applications in a IEEE 802.16 deployed Network

Hemant Kumar Rath[1] and Abhay Karandikar[2]

[1]TCS Networks Lab, Bangalore 560 066, India, Email:hemant.rath@tcs.com
[2]Department of Electrical Engineering, IIT Bombay, Mumbai 400 076, India, Email: karandi@ee.iitb.ac.in

***Abstract*— In this paper, we focus on various experiments conducted to analyze the performance of Transmission Control Protocol (TCP) and Universal Datagram Protocol (UDP) based applications in a IEEE 802.16 deployed network. We analyze the effect of Medium Access Control (MAC) and Physical layer characteristics on the performance of TCP and UDP-based applications. Our key findings are: (i) throughput achieved by TCP-based applications is lower than that of UDP-based applications, (ii) slot utilization of TCP-based applications is lower than that of UDP-based applications and (iii) throughput of TCP-based applications suffers in the presence of UDP-based applications for similar channel states. We also observe that throughput achieved by both TCP and UDP-based applications with Automatic Repeat ReQuest (ARQ) set in system are higher as compared to that of without ARQ. The findings of these experiments can be adopted while designing efficient scheduling schemes for IEEE 802.16-based network, such that higher throughput, utilization and better delay performance can be achieved.**

I. Introduction

The location dependent and time varying nature of wireless channel, the requirements of the applications such as delay and throughput guarantee and limited network resources etc., pose some of the key challenges in designing efficient systems for wireless networks. As suggested in the literature, cross-layer resource allocations [1]–[4] including opportunistic scheduling [5]–[7] are the major approaches designed to handle these challenges. In opportunistic scheduling, the user with the best channel condition is scheduled. This improves the overall network performance considerably. Further in wireless networks, it has been shown that guaranteeing Quality of Services (QoS) at one layer of Open Systems Interconnection (OSI) stack does not guarantee proportional QoS in other layers [8], [9]. This necessitates cross-layer network design in which layers other than adjacent layers participate, such that informed decision can be obtained and throughput can be maximized. Therefore, the impact of one layer on the performance of other layers needs to be investigated. Experiments are required to investigate this, and also to understand the performance of the current deployed networks. This not only provides information on the draw-backs of the current deployment, but also provides ideas to improve the system performance.

*Research of Abhay Karandikar is supported by India-UK Advanced Technology of Centre of Excellence in Next Generation Networks (IU-ATC) project and funded by the Department of Science and Technology (DST), Govt. of India and UK EPSRC Digital Economy Programme.

In this paper, we focus on our experiments conducted on IEEE 802.16-based network [10], which is being used to provide Broadband Wireless Access (BWA) in India and other parts of the world. We conduct experiments for both TCP and UDP-based applications in laboratory as well as in live-network, and investigate the performance of the deployed network. Our study includes performance analysis of both uplink and downlink scheduling schemes and effect of Automatic Repeat-reQuest (ARQ) on the throughput of the applications. The TCP-based applications we are referring belong to Non Real Time Polling Service (*nrtPS*) and Best Effort (*BE*) services, whereas the UDP-based applications belong to Real Time Polling Service (*rtPS*) services of IEEE 802.16. The rest of this paper is organized as follows. In Section II, we describe the experimental setup and measuring parameters. In Section III, we present the key findings of these experiments and analyze. In Section IV, we provide concluding remarks and discuss on some of the suggestions to improve the performance of IEEE 802.16-based network in future.

II. Experimental Setup

In Fig. 1, we illustrate the laboratory setup, in which two Subscriber Station (SS) modules are connected to one Base Station (BS) module. In the laboratory setup, we use a channel emulator to emulate the time varying wireless channel characteristics between SSs and the BS. We emulate sudden and gradual changes in the channel states by manually varying the parameters of the channel emulator. We connect one application terminal (Laptop) to each of the SS modules and one to the BS module. In Fig. 2, we illustrate the network setup involving IEEE 802.16 deployed network and the Internet. We connect two terminals as shown in Fig. 2 - one at the remote end within wireless network and the other on the Internet. For both the setups we employ traffic generator such as IPERF [11] to generate TCP as well as UDP packets. In the laboratory setup, we perform experiments with and without ARQ set, whereas in live-network we do not employ ARQ for experimentation. We use Wireshark [12] traffic monitor tool to monitor sequence number of packets, packet types, size, arrival and departure time, etc. The IEEE 802.16 deployed network used by the service provider employs "Weight-based Scheduler", the operation of which we illustrate in the following section. We then study system parameters of the network and describe the measuring parameters as follows.

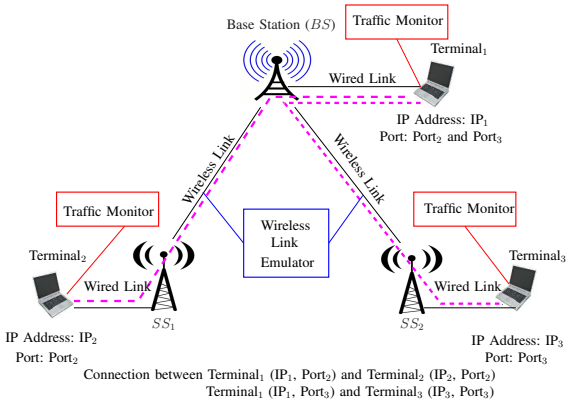

Fig. 1. Laboratory Experimental Setup

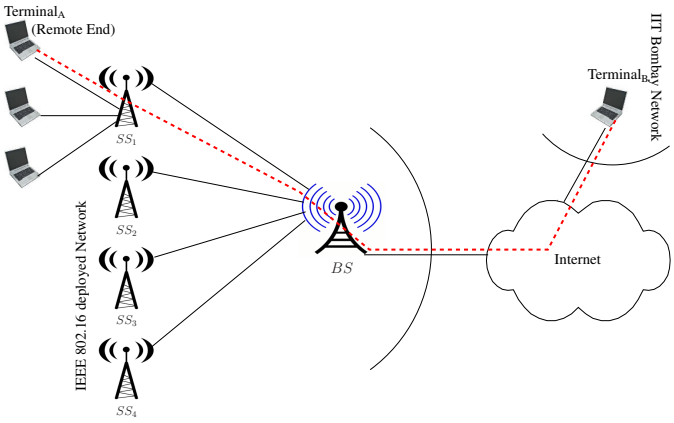

Fig. 2. Experimental Setup involving Live-network

A. Weight-Based Scheduler

Weight-Based (WB) scheduler is a proprietary adaptive modulation and coding scheduler implemented at the BS to schedule both uplink and downlink traffic. It assigns scheduling priority or weight to each user in a weighted fair scheduling way. If W_i is the weight of user i, then resource allocated (BW_i) to user i is such that the ratio $\frac{BW_i}{W_i}$ is same for all users. Weights of the users can be modified dynamically. For example, in a wireless network, if the rates of transmission of users are different, then the amount of bandwidth assigned among the users should be such that $\frac{BW_i}{\frac{1}{R_i}}$ is same $\forall i$, where R_i is the rate of transmission associated with i^{th} user. This kind of scheduling is known as Channel Dependent Weight-Based scheduling (WB-CD). If the variation in the rate of transmission is not considered during scheduling, i.e., slots are assigned only if there are requirements, then it is known as Channel Independent Weight-Based scheduling (WB-CI). The weights of each user in both WB-CI and WB-CD can be interpreted as a function of its requirements. Though, there are provisions for both WB-CD and WB-CI scheduling, WB-CD is used in the deployment and experimentation.

B. Measuring Parameters

Both setups employ short-term averaged Carrier to Interference-plus-Noise Ratio (CINR) based link adaptation algorithm to select modulation schemes and coding. We set the duration for CINR averaging used to select modulation schemes to 10 sec. Further, to add robustness in the noisy environments, two parameters: *protection* and *hysteresis* are defined. "protection" provides a margin in the CINR levels to select modulation schemes, whereas "hysteresis" prevents continuous modulation switching at modulation thresholds due to CINR fluctuations. For reliable and low Bit Error Rate (BER) operation in fast fading and high interference radio environment, high values of protection and hysteresis are desired. However, the values of protection and hysteresis are set to 4 dBm each (chosen by the operator). In Table I, we summarize other major system parameters used in the network setup.

TABLE I
SUMMARY OF SYSTEM PARAMETERS

System Parameters	Value
Air Interface	OFDM
Duplex Method	TDD
Modulation Schemes	64-QAM 3/4, 64-QAM 2/3 16-QAM 3/4, 16-QAM 12 QPSK 3/4, QPSK 1/2, BPSK
Frame Duration (T_f)	10 msec
$T_{ul} : T_{dl}$	3:2
QoS Priority	Upto 16 classes per SS
Hybrid ARQ (HARQ)	No

We categorize the experiments into three categories: Category I: TCP-based applications, Category II: UDP-based applications, and Category III: mixed applications (both UDP-based and TCP-based applications). In Fig. 3 - 5, we illustrate the experimental setup for the laboratory experiments involving all three categories. We conduct experiments for a sufficiently large duration of time and perform measurements through the traffic monitors over every ten minutes for each experiment. We repeat the experiments for different combinations of uplink and downlink transmissions. We also conduct similar experiments in the live-network. We now analyze the results obtained from both the laboratory and the live-network experiments.

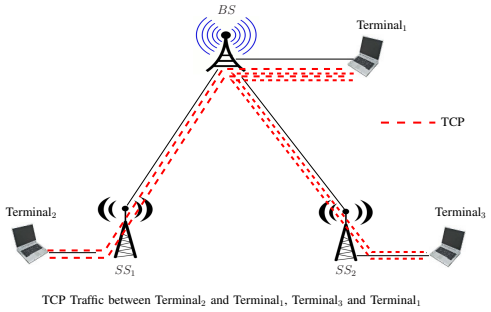

Fig. 3. Setup for Experimental Evaluation: Category I

III. EXPERIMENTAL RESULTS

1) Effect of ARQ: From Table II, we observe that the average throughput achieved by the TCP-based as well as UDP-based applications with ARQ set are higher as compared to that achieved without ARQ for similar channel states.

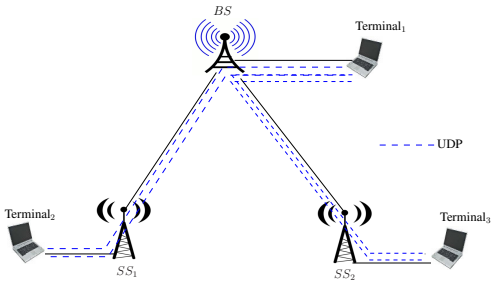

Fig. 4. Setup for Experimental Evaluation: Category II

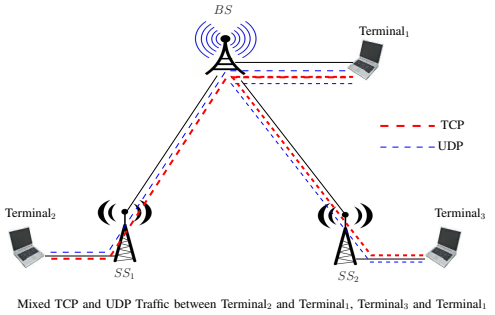

Fig. 5. Setup for Experimental Evaluation: Category III

Further, we observe that throughput achieved by TCP-based applications in laboratory setup is higher than that in live-network in both uplink and downlink, whereas it is reverse for UDP-based applications. In addition, we also observe that the throughput achieved by the downlink flows is higher than that of the uplink flows, in both the setups.

To investigate the reasons behind the higher throughput achieved in ARQ implemented system, we measure the percentage of packet re-transmission occurred in the laboratory setup with and without ARQ set (for TCP-based applications), which we present in Table III. From this table, we observe that the percentage of re-transmission drops drastically when ARQ is set in the network. The drop in re-transmission of packets enables the applications to achieve higher throughput when ARQ is set. We also observe that the percentage of re-transmission of packets in live-network is higher than that in laboratory setup. Moreover, the re-transmission percentage is more for uplink flows than that of downlink flows, when ARQ is not set. However, when the ARQ is set, we observe less re-transmission for the uplink flows than that of downlink flows.

2) Effect of MAC Layer Scheduler: From Table II, we observe that the throughput achieved by UDP-based applications are substantially higher (more than even 100% in some cases) than that of TCP-based applications for similar channel states, irrespective of the network types. We also observe that even though both TCP and UDP-based applications have same priority for scheduling, the UDP-based applications transmit more packets as compared to that of TCP-based applications. This may have resulted due to the following reasons.

TABLE II

THROUGHPUT ACHIEVED (IN KBPS)

Type	Laboratory	Live-network	Laboratory
TCP	**Without ARQ**		**With ARQ**
Uplink	408.4	238.1	558.9
Downlink	684.6	400.3	743.1
UDP	**Without ARQ**		**With ARQ**
Uplink	858.8	905.0	910.6
Downlink	847.7	995.4	952.8

TABLE III

RE-TRANSMISSION OF TCP PACKETS (IN %)

	Without ARQ		**With ARQ**
Type	Laboratory	Live-network	Laboratory
Uplink	20.68	9.93	2.2
Downlink	20.28	6.1	4.18

Since the WB-CD scheduler used for uplink and downlink scheduling does not differentiate TCP and UDP flows, it assigns time slots to both type of flows in the same order. Therefore, by assigning equal number of slots, a TCP flow with a small congestion window ($cwnd$) size will not be able to utilize all the slots assigned to it, if the number of slots assigned are more than its requirement (which is a function of $cwnd$ size). On the other hand, when the $cwnd$ size is very large and the number of slots assigned are not sufficient, then it will result in $cwnd$ drop and degradation in throughput. In addition, for UDP flows the rate of transmission is independent of the number of slots assigned. Therefore, UDP flows will be able to transmit at their peak rates, resulting in complete utilization of the slots assigned and higher throughput. Hence, by assigning equal number of slots to both TCP and UDP flows, TCP flows will not be able to fully utilize the slots assigned resulting in lesser throughput as compared to the UDP flows. The lower throughput and lesser channel utilization of TCP flows, which we observe, may have resulted due to the nature of the scheduling scheme implemented.

3) Effect of Changes in Channel States: We also observe the performance of TCP-based applications with gradual and sudden change in channel states. In Fig. 6, we plot a snapshot of throughput achieved by an uplink TCP flow in a laboratory experiment (Category I). From this figure, we observe that the instantaneous throughput fluctuates between 10 Kbytes/sec to 95 Kbytes/sec. The large fluctuation in instantaneous throughput can be attributed due to the high percentage of re-transmission of TCP packets. We also observe similar throughput variations for the downlink TCP flows. In Fig. 7, we plot a snapshot of the throughput achieved by an uplink TCP flow in a live-network experiment. From this figure, we observe that even though the instantaneous throughput fluctuates frequently, the average throughput is almost constant for the entire duration of the experiment. We also observe similar variations for the downlink TCP flows.

In Fig. 8, we plot the instantaneous throughput of an uplink TCP flow with manual change (sudden change) of the channel states. We observe that the sudden drop in throughput (after 0.8 min) occurs when we introduce more noise to the channel. We also observe that for a relatively noisy channel, the fluctuations in instantaneous throughput are less as compared to a less

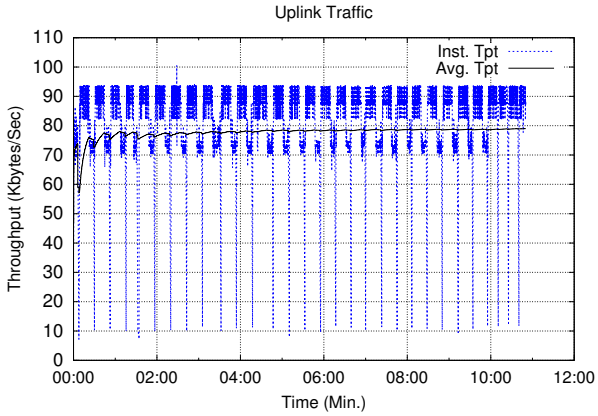

Fig. 6. TCP Throughput (Uplink, Laboratory, Without ARQ)

noisy channel. This may have resulted due to the nature of adaptive modulation schemes employed in the system.

In Fig. 9, we plot the instantaneous throughput achieved by a downlink TCP flow with manual change in the channel state. In this experiment, we tune the channel states in every one minute (at 1, 2, 3 min). From this figure, we observe that the instantaneous throughput varies significantly when the channel state changes. Even though the short-term CINR averaging is being used in the system, we still observe fluctuations in throughput. This may be due to the improper scheduling scheme selected in the system.

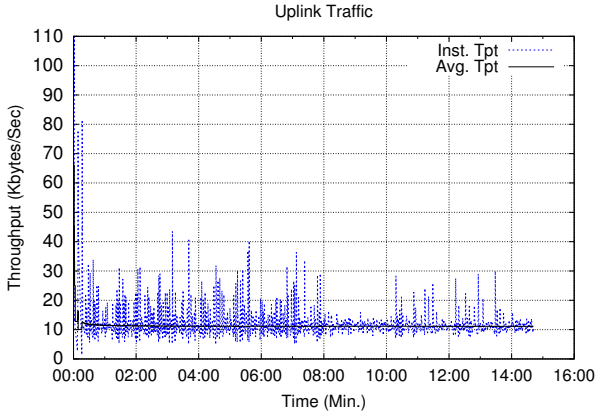

Fig. 7. TCP Throughput (Uplink, Live-network, Without ARQ)

4) Effect of Parallel TCP Flows on Performance: We also investigate the effect of one TCP flow on another TCP flow running through the same SS. For this, we conduct experiments with two TCP flows running simultaneously, the second flow starting 1.30 mins after the start of the first flow. From Fig. 10 - 11, we observe that the fluctuation of instantaneous throughput is more, when more than one TCP flows are running. This can be accounted as follows: by assigning equal number of slots to both the flows, the scheduler may force one flow to drop its $cwnd$ due to insufficient slots assigned to it, even when the slots assigned to the other flow are not being utilized fully.

5) Effect of Concurrent UDP Flows on TCP Performance: We also investigate the effect of UDP flows on the performance of TCP flows when both TCP and UDP flows run simultaneously (Category III). For this, we conduct experiments with one TCP and one UDP flow running simultaneously through the same SS, and, then stop the UDP flow after one minute. We repeat this experiment both in the uplink and downlink directions. We observe that when both flows run simultaneously, the TCP-flow gets starved and all the slots assigned to one SS is being utilized by the UDP flow only. The TCP flow transmits only when the UDP flow stops. Through

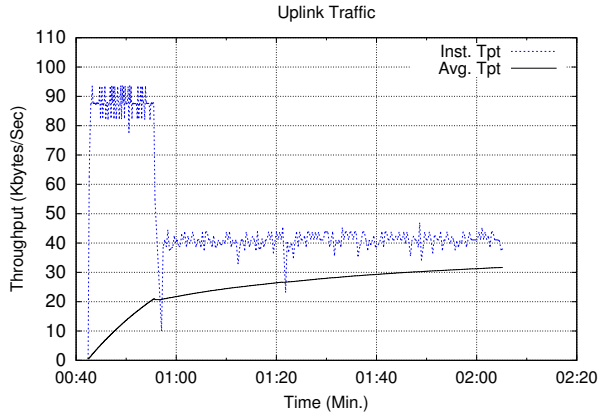

Fig. 8. Effect of Wireless Channel on TCP (Uplink, Laboratory, Without ARQ)

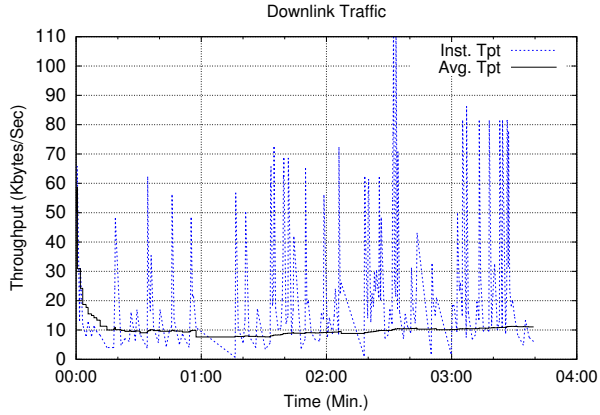

Fig. 9. Effect of Wireless Channel TCP (Downlink, Laboratory, Without ARQ)

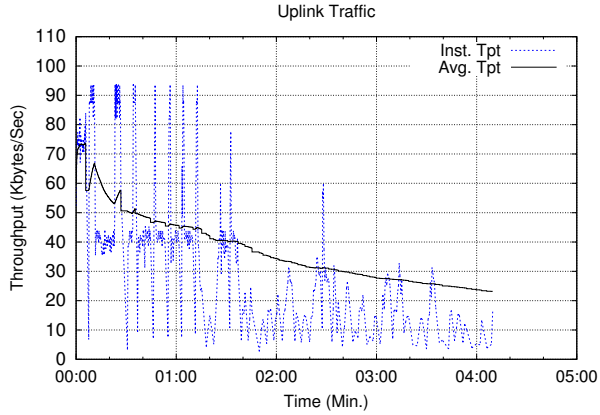

Fig. 10. Effect of a Parallel TCP Flow - Second Flow Starts at 1.30 minute

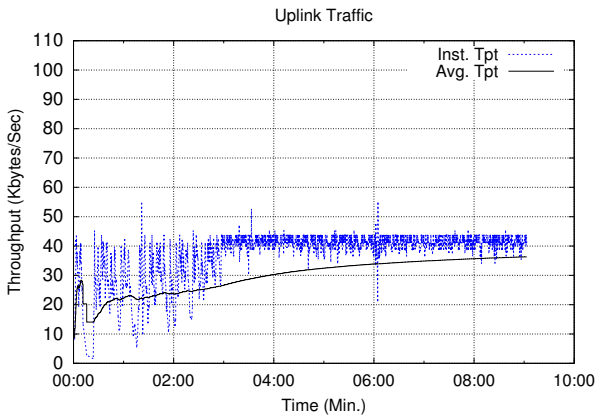

Fig. 11. Effect of a Parallel TCP Flow - First Flow Stops at 3.00 minute

Fig. 12, we demonstrate that the instantaneous throughput of the TCP flow drops to zero just after it starts (initial one minute) and increases its rate of transmission to its usual state only after one minute (when the UDP flow stops). In Fig. 13, we plot the instantaneous throughput achieved by a downlink TCP flow in the presence of an UDP flow and observe similar drops in TCP throughput (drop in throughput between 0.50 min to 02.50 min, in Fig. 13).

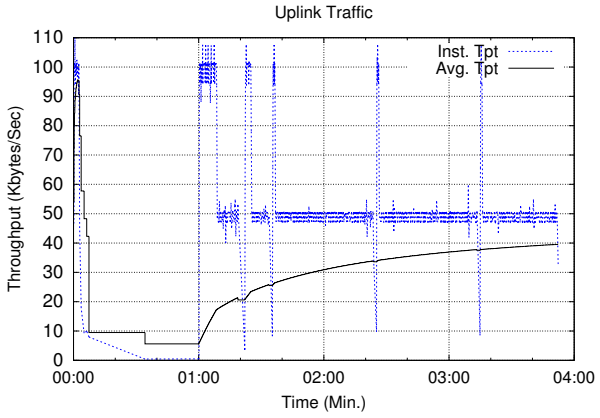

Fig. 12. Effect of UDP on TCP Flow (Uplink, Laboratory, With ARQ)

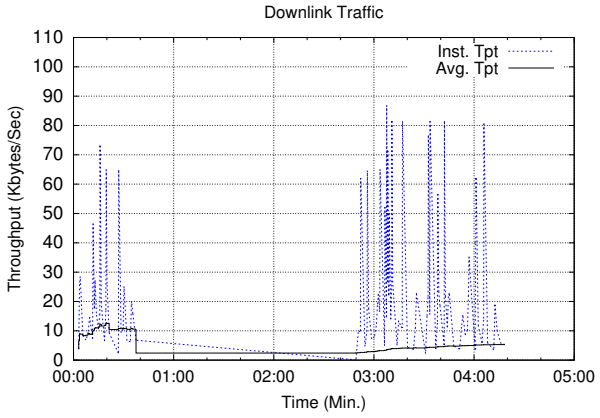

Fig. 13. Effect of UDP on TCP Flow (Downlink, Laboratory, Without ARQ)

Though we have illustrated the results of laboratory experiments, similar results are also valid in live-network setup.

IV. Summary and Observations

From both sets of experiments, we observe that the version of Weight-based scheduler implemented in the deployment is biased towards UDP flows, resulting in degradation of services of TCP flows, both in uplink and downlink directions. Moreover, it does not provide any guarantee in terms of delay and throughput of TCP traffic, resulting in timeout in most of the cases. Since it does not guarantee any scheduling delay, packets of real-time services may get dropped at SS due to deadline violation. In addition, since it does not assign slots based on requirement, slot under-utilization and throughput degradation occurs. We further observe that throughput achieved by both TCP and UDP-based applications with ARQ set in system are higher as compared to that of without ARQ. This suggests that ARQ should be implemented in the system. However, the effect of ARQ on scheduling delay and packet deadline needs further investigation and analysis.

From these observations, we infer that alternative scheduling schemes need to be designed specifically for TCP-based applications which consider TCP parameters like $cwnd$ size and "timeout", such that network performance can be improved in terms of delay, throughput and channel utilization.

References

[1] L. Alonso and R. Agusti, "Automatic Rate Adaptation and Energy-Saving Mechanisms Based on Cross-Layer Information for Packet-Switched Data Networks," *IEEE Radio Communications*, pp. S15–S20, March 2004.

[2] M. van der Schaar and N. Sai Shankar, "Cross-Layer Wireless Multimedia Transmission: Challenges, Principles, and New Paradigms," *IEEE Wireless Communications*, vol. 2, pp. 50–58, August 2005.

[3] X. Zhang, J. Tang, H.-H. Chen, S. Ci, and M. Guizani, "Cross-Layer-Based Modeling for Quality of Service Guarantees in Mobile Wireless Networks," *IEEE Communications Magazine*, vol. 44, pp. 100–106, January 2006.

[4] X. Qiuyan and M. Hamdi, "Cross Layer Design for IEEE 802.11 WLANs: Joint Rate Control and Packet Scheduling," in *Proc. of IEEE LCN*, pp. 624–631, November 2005.

[5] R. Knopp and P. A. Humblet, "Information Capacity and Power Control in Single-Cell Multiuser Communications," in *Proc. of IEEE ICC*, vol. 1, pp. 331–335, June 1995.

[6] H. S. Wang and N. Moayeri, "Modeling, Capacity, and Joint Source/Channel Coding for Rayleigh Fading Channels," in *Proc. of IEEE VTC*, pp. 473–479, May 1993.

[7] S. S. Kulkarni and C. Rosenberg, "Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces, and Short Term Fairness," *Wireless Networks*, vol. 11, no. 5, pp. 557–569, 2005.

[8] M. C. Chan and R. Ramjee, "Improving TCP/IP Performance over Third Generation Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 430–443, 2008.

[9] R. Ludwig and R. H. Katz, "The Eifel algorithm: Making TCP robust Against Spurious Retransmissions," *ACM SIGCOMM Computer Communication Review*, vol. 30, pp. 30–36, August-September 2000.

[10] LAN/MAN Standards Committee, *IEEE Standards for Local and Metropolitan Area Network: Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems (Amendment2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands)*. IEEE Computer Society and IEEE Microwave Theory and Techniques Society, February 2006.

[11] "iperf." [Online] `http://sourceforge.net/projects/iperf/`.

[12] "Wireshark." [Online] `http://http://www.wireshark.org/`.