Improving QoE for Skype Video Call in Mobile Broadband Network

Jing Zhu and Rath Vannithamby
Intel Labs
Hillsboro, OR, United State
{jing.z.zhu, rath.vannithamby}@intel.com

Christoffer Rödbro, Mingyu Chen,
and Søren Vang Andersen
Skype
{christoffer.rodbro,mingyu.chen,
soren.vang.andersen}@skype.net

***Abstract*—** in this paper, we introduce an app-radio cross-layer framework for improving Quality of user Experience (QoE) of Over-The-Top (OTT) Internet applications in Mobile Broadband Networks, e.g. WiMAX, 3G, LTE, etc. We apply methods similar to the well-known network layer techniques: Explicit Congestion Notification (ECN) and Differentiated Services (DiffServ) to application layer and wireless link layer. We focus on UDP based delay-sensitive real-time video call applications such as Skype, and propose to exchange cross-layer information, e.g. congestion, packet priority, etc., through API (application programming interface) based control signals instead of the existing ECN and DiffServ fields in IP data packet header. We built a prototype system with Microsoft Windows XP OS and Intel Mobile WiMAX network adapter, and use POLQA Mean Opinion Score (MOS) to measure the audio QoE for the video call. Empirical results show that immediate congestion indication effectively speeds up Skype response to bandwidth variation, and intra-flow prioritization further reduces the delay of audio packets. Both significantly improve the audio MOS of a video call.

Keywords-cross-layer; QoE;mobile broadband;Skype

I. INTRODUCTION

Today, we can evidently see the explosive growth in network access demand with the introduction of smart devices and various Internet applications. Mobile broadband network capacity continues to increase with new generation of air-interface standards such as 3GPP LTE [1] and IEEE 802.16m [2]. However, most if not all Internet applications are still running over best-effort pipes, with no guaranteed quality of service (QoS) and little consideration for the user experience of real-time communication. The main reason is that these Internet applications are delivered Over-The-Top (OTT), and mobile broadband network operators are not responsible, nor able to control, their QoS. Oftentimes, the operator may not even be aware of the contents of IP packets generated by these OTT applications due to strong encryption.

Over best-effort networks, the importance or real-time requirements of packets are not known at the radio/network-interface. For example, a VoIP user may have stringent delay requirement while a video streaming user may have high throughput requirement, however, all packets from all applications are treated the same and scheduled in a simple first-come-first-serve fashion. Thus, it is challenging to provide a good user experience for Internet applications in Mobile Broadband networks. 3GPP has taken the effort to support QoS provisioning for Internet applications through IP Multimedia Service (IMS) [3]. However, many practical issues [4], for example, trade-off between benefits and complexity, service monitoring and accounting, etc. are not yet fully addressed.

In this paper, we study how QoE can be enhanced for OTT Internet applications over mobile broadband networks, considering only the last-mile radio access network. Our approach is to take advantage of the fact that applications are running on the end-user device over the top of the radio, and allow direct information exchange between apps and radios, i.e. app-radio cross-layer framework.

Today, applications are unaware of what is happening at the radio layer and the radio is ignorant of the requirements of the applications. In the legacy systems, where operators had the full control or were the providers of the applications such as voice communication, QoS provisioning worked well to provide a good user experience. However, since the Internet applications are supported mainly over the best effort pipes without any QoS provisioning, the app-radio cross-layer framework opens up new possibilities for providing a better user experience. In this paper, we will focus on the widely used Internet video call application - Skype, and apply app-radio cross-layer API based versions of the well-known network layer techniques: Explicit Congestion Notification (ECN) and Differentiated Services (DiffServ) with the app-radio cross-layer framework.

The rest of the paper is organized as follows: Section 2 gives a brief overview of related works, Section 3 introduces the app-radio cross-layer protocol stacks along with the cross-layer information that will be exchanged between apps and radios, Section 4 describes the details of the two cross-layer algorithms, Section 5 explains our experiment setup and QoE evaluation methodologies, Section 6 discusses the empirical results. Finally, conclusions and future work are mentioned in Section 7.

II. RELATED WORKS

Traffic Class (TC)-based Quality of Service provisioning has been very well studied in the literature. The authors of [8] provide a good list of ways to classify traffic in ITU-T, IETF, 3GPP, and IEEE. In the 3GPP Evolved Packet System [9], traffic class is also called "bearer", and each bearer may consist of one or multiple flows that receive a common QoS treatment. In a WiMAX network, the authors of [10] showed how to map traffic class based on the DSCP value of IP packet

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

and provision end-to-end QoS for VoIP applications. However, many roadblocks and challenges that can be scientific, technical, economic, and legal as discussed in [8], still exist for deploying TC-based QoS provisioning in real-life, and today's Internet, wired or wireless, is still operating on the basis of Best-Effort. For example, it is not clear which party should initialize the QoS provisioning procedure. Should the required quality level be selected by network or suggested by a user? Who should pay for the cost of QoS provisioning, e.g. end-user, application vendors, etc.? Should QoS provisioning be soft or hard? How to monitor QoS performance in a scalable and efficient way?

ECN (Explicit Congestion Notification) [11] allows end-to-end notification of network congestion without dropping packets, and has been adopted by 3GPP to support codec rate adaptation. However, it is not currently possible to use of ECN with UDP because network APIs do not give access to ECN bits. Moreover, because it relies on end-to-end feedback that is delayed by at least one RTT, ECN does not overcome the problem of late reaction to congestion.

III. APP-RADIO CROSS-LAYER PROTOCOL STACKS

Today's Internet can be generally described by the well-known four-layer model [5]: the Application layer, the Transport layer, the Network layer, and the Link layer. In this paper, we are mainly interested in wireless links and therefore "Radio" will be used instead of "Link" in the following discussions.

Figure 1 shows the proposed app-radio cross-layer architecture, where new interface is added between application layer and radio layer to allow direct information exchange between them. This architecture is completely transparent to the two middle layers, e.g., Transport and Network, and can therefore work with all existing TCP/IP implementations on today's operation systems.

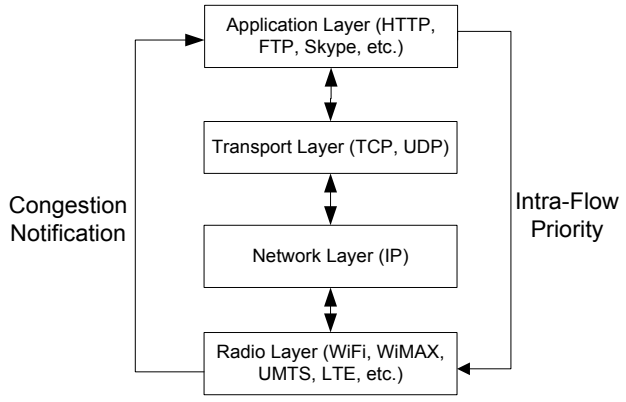

Figure 1: App-Radio Cross-Layer Protocol Stacks.

Next, we will consider the widely used Skype video conference application, and show how to improve user experience in today's commercial mobile broadband networks by using the following cross-layer information:

- **Congestion Notification (CN):** a binary flag to indicate if the radio access network is experiencing congestion or not.
- **Intra-Flow Priority (FP)**: packets belonging to the same IP flow that is uniquely identified by its source IP address, destination IP address, source port number, destination port number, and protocol type are further classified into sub-flows based on its priority.

Specifically, **Congestion Notification** is provided by the Radio layer, and used by the Application layer to reduce its source rate to alleviate network congestion; **intra-Flow Priority** is provided by the Application layer, and used by the Radio layer to better schedule packets when congestion happens.

The cross layer information exchange for CN is straightforward, but less so for FP. **Inter-flow** packet prioritization can be implemented through the IP packet priority field [6], whereas exploiting this mechanism for **intra-flow** prioritization is trickier because it requires the application to deliver its data on different sockets, in turn requiring substantial changes in the application. Here, we propose a method specifically for UDP traffic that is easier for an application to utilize. It encodes the "Priority" information in the existing "Total Length" field, aka packet size, of the IP header, and works as follows.

First of all, a set of packet size to priority mapping rules are negotiated between the application layer and the radio layer. When a rule is agreed, the application will packetize each of its packets in the way that its priority can be deduced from its packet size. For example, a single-threshold rule may say "the packets smaller than 200 bytes have high priority, otherwise they have low priority." In another example, "the packets with even packet size have high priority, and the ones with odd size have low priority." Notice that Intra-Flow priority applies only to packets belonging to the same flow, and packets from different flows still follow the first-come-first-serve rule.

IV. CROSS-LAYER ALGORITHMS

Available bandwidth for individual users in mobile broadband networks is changing due to interference, fading, and other users' traffic. A video conferencing application like Skype will adapt its coding rate to these variations, in a range from a few 10s of kbps to several Mbps. Typically, due to the 'asymmetric available bandwidth' feature in many Internet access networks, the uplink often turns out to be the bottleneck for a video conference call. When available bandwidth of the uplink at the sender side drops, it will take at least one round trip time (RTT) for the application to respond.

Let us assume that source rate is equal to available bandwidth of the uplink at the source side, both of which are constant bit rate (CBR), that is, the application is utilizing the full capacity. Now, if available bandwidth reduces by k times and after one RTT the application reduces its source rate to zero, we have total time (T) for sending out all the packets generated during the RTT as

$$T = RTT \times k \quad (1)$$

If *RTT* = 200ms and *k* = 10, we have T = 2 seconds, that is, over 200 ms the end-to-end delay increases by 2 seconds. Such sudden delay increases lead to under-runs in the far side jitter buffer causing significant audio degradations (cut-outs). These problems become even worse if the downlink becomes congested simultaneously, further delaying the end-to-end feedback.

To reduce feedback delay and speed up Skype's response to bandwidth variation, we consider the following two cross-layer algorithms:

- intra-flow prioritization
- radio-aware source rate adaption

A. Intra-Flow Prioritization

Figure 2 shows the system diagram of the two-queue intra-flow prioritization scheduler in Intel WiMAX radio.

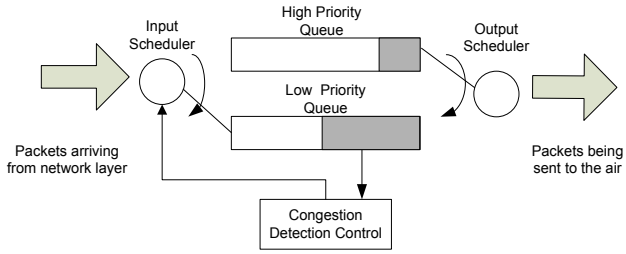

Figure 2: Intra-Flow Prioritization Scheduler System Diagram.

The input scheduler puts an incoming packet to the high priority queue if both the queue is open and the packet is high priority based on the current packet size to priority mapping rule, otherwise to the low priority queue.

The output scheduler follows the hard priority policy, and always schedules packets in the high priority queue for transmission first, and schedules low priority packets only if the high priority queue is empty.

The congestion detection control monitors packet delay in the low priority queue, and drops the packets that have waited for > 1 second. Furthermore, it notifies the input scheduler to open and close the high priority queue based on the following algorithm:

- open the high priority queue if the delay of any packet in the low priority queue exceeds D_1
- close the high priority queue if the delay of last N_2 packets in the low priority queue is below D_1

In our experiment, we set D_1 = 100ms and N_1 = 10. The main reason for not keeping the high priority queue open all the time is to avoid excessive packet reordering as will be introduced by intra-flow prioritization.

Intra-flow prioritization does not have any feedback delay, and can reduce source rate by x% instantly, where x% indicates the ratio of low priority packets in a flow. Note that this mechanism does suffer from the classical DiffServ problem that an application can specify all its traffic as being high priority, thereby upsetting fairness. However, as the mechanism is running solely at endpoints the lack of fairness is only between a user's own applications. Therefore we are not overly concerned with this problem. In addition, we can limit the input rate of the high priority queue. When the limit is reached, packets will be put in the low priority queue regardless their priority.

B. Radio-aware Source-Rate Adaptation

Considering the two-queue scheduler system as shown in Figure 2, we use the following delay threshold based algorithm to determine the CN value:

- If the delay of *any* packet exceeds a threshold – D_2, CN will be set to 1
- If the delay of last N_2 packets in the low priority queue is below another threshold – D_3, CN will be set to 0

In our experiment, we set D_2 = 200ms, D_3 = 100ms, and N_2= 5. Whenever CN changes, the WiMAX radio will inform the updated CN value to Skype that can then adapt its source rate immediately instead of relying solely on the delayed feedback from the remote end. The reason for applying hysteresis to CN instead of just reporting the per-packet queuing delay is that the reporting itself is a cross system layer call that may be costly. Thus we want to keep the number of necessary CN updates at a minimum.

In Skype, bandwidth estimation and rate control is primarily handled at the receiving side, where a requested transmission rate R is determined from one-way delay and packet loss based congestion signals. The estimated R is fed back to the transmitting side, where it is distributed over the outgoing streams according to their pre-determined priorities. In this paper, we further use the congestion indicator CN to modify R prior to distributing it over the streams. While many possibilities exist, we choose a simple linear regulator to arrive at a target queuing delay D_T:

$$R_{CN} = R + \min(0, \alpha R (D_T - f(CN))) \quad (2)$$

In a simplest implementation, we use constants for $\alpha = 1.6$, $D_T = 0.05$ s, and f(CN) is an aggregation of recent CN values that should allow fast reaction to congestion but avoid too much oscillation:

$$f(CN) = \begin{cases} \min(3D_2,\ f(CN) + D_2), & \text{if } CN = 1 \\ (1 - 0.125\Delta T) f(CN), & \text{if } CN = 0 \end{cases} \quad (3)$$

where ΔT is the time passed since the last update, which is done whenever CN changes and then periodically every 250 ms. The adaption step in Eq. (2) is restricted to negative values because the bottleneck may sit elsewhere on the end-to-end path. Effectively, the congestion notification shortcuts Skype's rate-control feedback loop leading to more agile adaption when the radio uplink is the bottleneck.

Figure 3 shows how R_{CN}/R changes during 10 seconds based on Eq.(2) and (3), assuming CN is 1 for the first 1 second, and 0 in the rest of 9 seconds. The reader should keep in mind that the reported parameters were found to be appropriate for the Skype application, but they may not be the best choice in other embodiments.

At a first glance, it would seem straight-forward to modify the rate adaption step in (2) to a fairness-ensuring policy as in

[12]. However, another application running in the mobile terminal without adapting to CN may constantly make a CN-adapting application back-off. Therefore, we will not strive for fairness here; rather, an application adapting to CN will in general be "more than fair". As discussed in the previous section, this problem is of limited impact in the scenario at hand. Moreover, we may disable CN reporting at the radio layer if multiple flows are competing for resources.

Notice that radio-aware source-rate adaptation and intra-flow prioritization can work simultaneously.

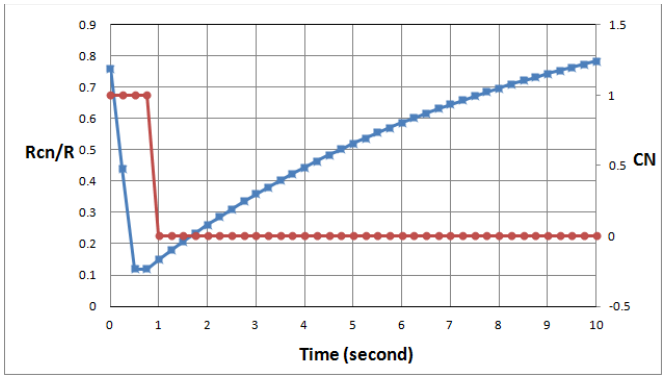

Figure 3: Tracing RCN/R Variation.

V. Evaluation Methodologies

Two PCs running Windows XP work as Skype transmitter and receiver respectively. The receiver is connected to Internet through high-speed DSL, and the transmitter equipped with Intel WiMAX network adapter is connected to CLEAR's mobile WiMAX network deployed in Hillsboro, Oregon.

As the proposed methods target at the uplink direction we will only consider uplink performance in our experiment. This is reasonable because uplink is usually more challenging than downlink in a mobile network.

In our experiments we use a special Skype build that allows reading a wav file and loop it as audio input in place of the microphone signal. As video input, we play a short movie repeatedly on a big monitor, and let the webcam of the Skype transmitter face the monitor and capture the video in real time. For intra-flow prioritization, we use the single-threshold rule where Skype packetizes its audio data into packets smaller than 200 Bytes, whereas video goes into larger packets, thereby giving audio priority over video.

At the Skype receiver, the audio output is saved to a wav file, which by offline processing is split into segments corresponding to the looped input signal. The resulting excerpts are compared to the reference wav file by the POLQA algorithm [7] to estimate audio MOS.

We also measure source rate (kBps) and average delay in the WiMAX radio at the Skype transmitter side.

All experiments are conducted in a commercial network with no control of background traffic from other users.

We consider the following two types of test: lab test and field test. In the lab test, the Skype transmitter is static, and we add a programmable attenuator between the WiMAX network adapter and its antenna to control link quality. In the field test, we put the Skype transmitter in a car and drive around in the city of Hillsboro following the loop route (A→ B→ C→ D→A) shown in Figure 4. The route consists of freeway as well as city streets with traffic light, and one loop takes about 30 minutes.

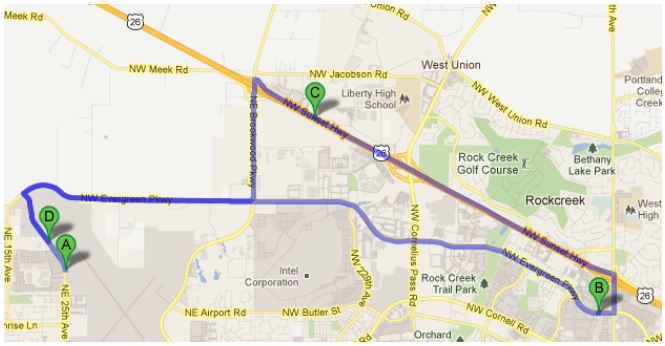

Figure 4: Drive Route Map for Field Test (Hillsboro, OR, United States).

VI. Empirical Results

A. Lab Test

First, let us study the performance of rate adaptation and intra-flow prioritization in the lab test with periodic link quality degradation. We adjust the attenuation periodically so that the WiMAX RSSI stays in good link quality (-72dBm) for 90 seconds, and then switches to low link quality (-80dBm) for 20 seconds. We are mainly interested in the switching point when the RSSI drops from -72 dBm to -80dBm, and its impact on the voice quality in a Skype video call.

Figure 5 shows the traces of source rate and average delay for three cases: baseline, rate adaptation, and intra-flow prioritization. Three curves in the figure correspond to three independent runs, and we align them roughly in time for easy comparison.

We see huge delay increase in the baseline case, and it takes long time for the delay to fall back to the same level as before signal quality drops. With rate adaptation or intra-flow prioritization, both magnitude and duration of delay spike reduces significantly.

Figure 6 compares the three cases in terms of voice quality as measured by POLQA MOS. The average score of 30 consecutive samples is 3.61, 3.78, and 4.02 for the baseline, rate adaption, and intra-flow prioritization methods, respectively. More importantly, we see that both features significantly reduce the number of "bad" (MOS<3) excerpts.

B. Field Test

For the field testing, we compare the baseline against the performance when turning on both congestion notification and intra-flow priority. As we utilize a public network, test conditions are in general not reproducible, but we try our best to make the two tests comparable. For each test, we ran a continuous Skype video call, and drove two loops following the same route as shown in Figure 4, which takes about one hour. We ran the two tests on the same weekday.

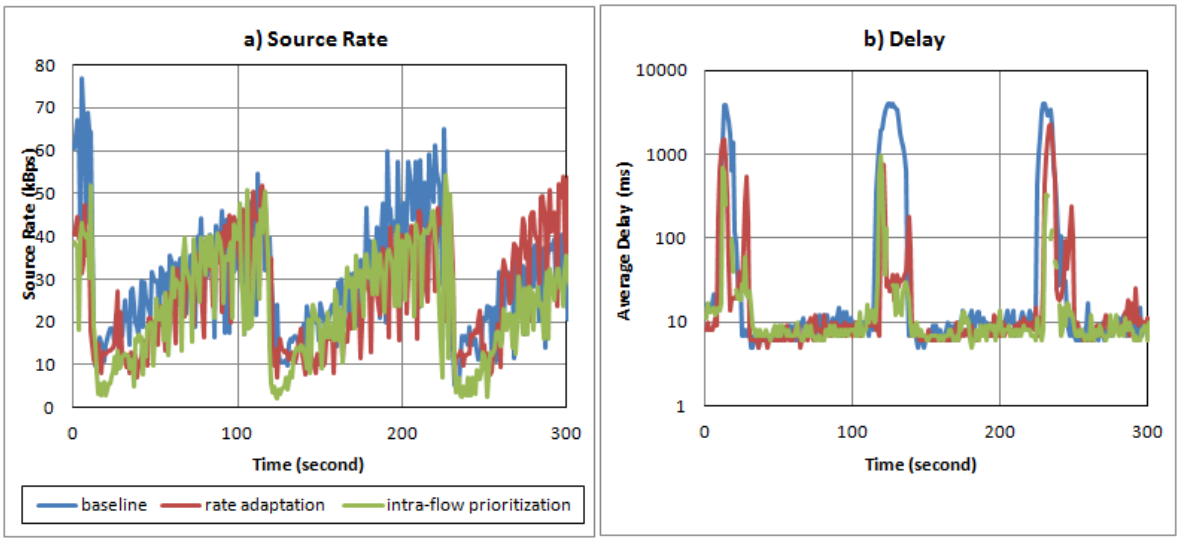

Figure 5: Rate Adaptation vs. Intra-Flow Prioritization: Rate and Delay.

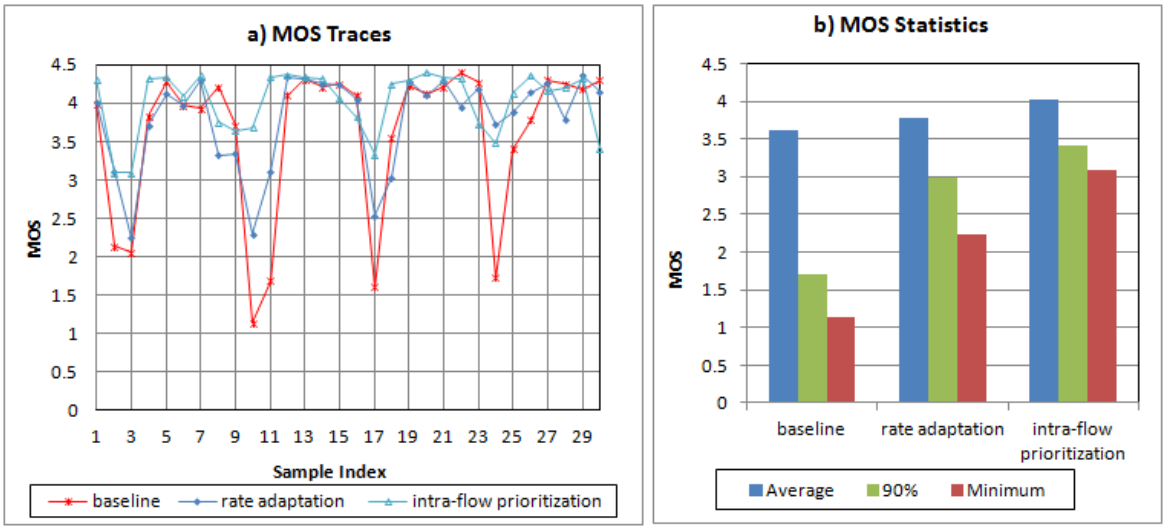

Figure 6: Rate Adaptation vs. Intra-Flow Prioritization: MOS.

Figure 7 compares the two tests in terms of RSSI, source rate, average delay, and MOS, using the cumulative distribution function approach. We see that in general, the two RSSI curves are well aligned, showing the channel conditions in two tests are similar. As expected, in the cross-layer test, with both intra-flow prioritization and radio-aware rate adaptation enabled, Skype becomes more conservative, and uses less bandwidth. The average source rate is 42.5 kBps, about 85% of that of the baseline. On the other hand, the cross layer features significantly reduce both average and long-tail delay. Particularly, the 99% percentile of average delay is reduced from 1.6 seconds to 120 ms by more than 10 times. Moreover, the percentile of MOS < 3 is 13% in the baseline test, and decreases to 6% in the cross-layer test, and the share of "bad" excerpts is significantly reduced by over 50%.

VII. CONCLUSION

In this article, we proposed app-radio cross-layer optimizations bypassing both transport and network layers to improve Quality of user Experience (QoE) in mobile broadband networks. We took a prototyping approach, and showed the benefits and the feasibilities of two techniques: radio-aware source-rate adaptation and intra-flow prioritization, in real-life environments with the popular Skype video conference application.

We were concerned with the uplink direction only. In order to apply the congestion notification to the downlink direction it would have to traverse the network path, essentially turning it into ECN. The intra-flow prioritization on the other hand can also be applied to downlink with the support of new protocols for the client to communicate the intra-flow prioritization rule to the Base Station. This is a topic of ongoing work. Additionally, we plan to further optimize the control parameters (D_1, D_2, D_3, N_1, N_2, α, D_T, f(CN)) of rate adaptation and intra-flow prioritization through extensive simulation studies. Also, while the experiments in this paper were carried out on the WiMAX network, the proposed methods are general enough that they should also be applicable to other mobile

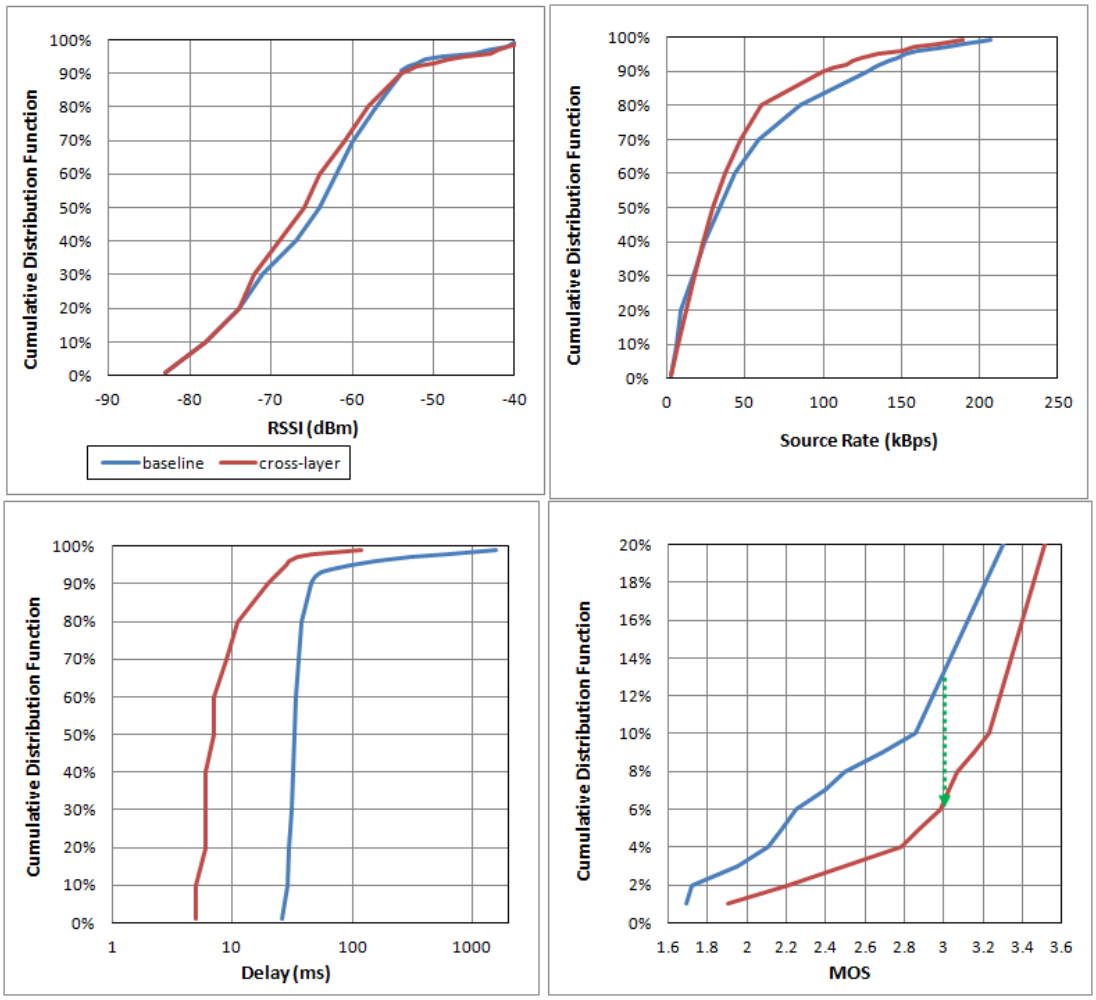

Figure 7: Field Test Performance Comparison.

networks such as 3GPP UMTS and LTE networks, and possibly even to WiFi – another topic under investigation.

Finally, we believe that app-radio cross layer optimizations as proposed herein are becoming practically feasible these years. Thanks to the proliferation of downloadable applications and Cloud-Computing, it has never been easier to update software including applications and radio drivers on end-user device. With the app-radio cross-layer architecture, it is promising for radio vendors and Internet application vendors to collaborate in delivering consistent, seamless, and rich Mobile Internet user experience.

ACKNOWLEDGEMENT

The authors would like to thank Ali Koc, Maruti Gupta and Geoff Weaver from Intel for valuable discussions.

REFERENCES

[1] 3GPP TS 23.300, Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2

[2] IEEE Std 802.16m-2011(Amendment to IEEE Std 802.16-2009, IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 3: Advanced Air Interface

[3] 3GPP TS 23.228, IP Multimedia Subsystem (IMS); Stage 2

[4] Aref Meddeb, "Internet QoS, Pieces of the Puzzle", IEEE Communications Magazine, Jan 2010

[5] IETF RFC 1122, Requirements for Internet Hosts – Communication Layers, Oct. 1989

[6] IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

[7] ITU-T Recommendation P.863: Perceptual objective listening quality assessment

[8] R. Stankiewicz, P. Cholda, and A. Jajszczyk, QoX: What is It Really? IEEE Communications Magazine, April 2011

[9] H. Ekstrom, QoS Control in the 3GPP Evolved Packet System, IEEE Communications Magazine, Feb. 2009

[10] C. M. Lin and S. R. Tsai, Provisioning an End to End QoS for VoIP over WiMAX network, International Computer Symposium (ICS), Dec. 2010

[11] IETF RFC 3168, The Addition of Explicit Congestion Notification

[12] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.