ExBox: Experience Management Middlebox for Wireless Networks

Ayon Chakraborty†, Shruti Sanadhya, Samir R. Das†, Dongho Kim, Kyu-Han Kim
† Stony Brook University, Hewlett Packard Laboratories
† {aychakrabort,samir}@cs.stonybrook.edu
{shruti.sanadhya,dongho.kim,kyu-han.kim}@hpe.com

ABSTRACT

Enterprise wireless networks face significant challenges to deliver Quality-of-Experience (QoE) with the variety of mobile applications. One of the fundamental challenges is that the traditional definition of network capacity (often defined as throughput capacity) is not sufficient to reflect applications' requirements in wireless networks. In this paper, we propose to rethink the network capacity of wireless networks to better incorporate QoE. Specifically, we first propose a novel concept of an Experiential Capacity Region (*ExCR*) for wireless networks. *ExCR* is defined as a set of simultaneous application flows whose QoE requirements can be satisfied by the network. Next, we present the infrastructure based ExBox system that measures per-application QoE metrics and determines the *ExCR* for wireless networks to better serve a set of mobile application flows. In its core, ExBox employs light-weight machine learning techniques that are tailored for dynamic wireless environments. Through both large-scale simulations and extensive real-life experiments on WiFi and LTE networks, we show that ExBox delivers QoE in admission control decision with a precision of $\approx 0.8-0.9$, even when clients experience diverse channel quality. Moreover, ExBox quickly adapts to changing network environments without much overhead.

1. INTRODUCTION

Enterprise wireless networks (e.g., building, campus, metros) are moving towards highly mobile environment. At the same time, the diversity of applications as well as network infrastructures is growing. For example, applications range from web downloads to audio/video streaming, speech recognition and desktop sharing. Networks range from various incarnations of WiFi alongside LTE small cells [20]. It is often a challenge for enterprise network administrators to develop a good balance between network capacity and user experience. Blindly over-provisioning access networks to support all possible user demands is clearly prohibitive due to both operational and spectrum costs. There is thus a need for Quality of Experience (QoE) management so that specific demands are directed to right networks (i.e., network selection) according to their respective loads. Moreover, even when only one network is available, admission control policies are needed such that aggregated user experience of existing traffic does not suffer due to newly arriving flows.

There is already a large volume of research in network selection and/or admission control (e.g., [10, 7, 31, 65, 51, 45, 56, 63]) that appears to be addressing these challenges. However, these solutions have three critical limitations:

Focus on single QoS metric or application type: Most approaches focus on modeling Quality-of-Service (QoS) metrics, such as throughput, delay or loss. Often the focus is on using only one such metric, typically throughput [65, 51, 45], either directly or via a utility function-based formulation [55]. However, such metrics do not relate in a straightforward manner to the eventual quality of experience (QoE) of the user, specifically for modern interactive and streaming applications. In particular, there is a growing understanding that *multiple QoS metrics influence user QoE in complex ways* [29, 28]. Commercial admission control solutions focus purely on authentication [10] or use simplistic metrics, such as maximum number of flows [7, 13] or bandwidth [55, 9, 2, 22]. While QoE based capacity has been proposed for VoIP [62] and video flows [59], these models still cannot be applied to real enterprise networks, which often contain a mix of applications [34]. For instance, these models cannot answer, "How does the QoE of existing VoIP flow change when a new streaming video flow enters the network?"

Lack of ability to capture dynamic behavior: Unlike wired networks, wireless networks suffer from multi-path fading, interference and user mobility. Accordingly, the total *capacity* of these networks also changes. Network administrators, thus, need to adapt to these dynamic network scenarios. For example, at some instant one VoIP and one Web flow may be admissible in a WiFi network, but if the VoIP user moves away from the access point, the observed QoE

CoNEXT '16, December 12-15, 2016, Irvine, CA, USA
 ISBN 978-1-4503-4292-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/2999572.2999597

for VoIP flow may go down. Existing approaches cannot capture such dynamic behavior.

Lack of 'blackbox' solution: Conventional admission control solutions such as [56, 53, 63] depend on a meticulous modeling of packet queuing and traffic behavior. Most network elements today (e.g., APs, middleboxes, switches) are closed devices. This makes it hard to model their performances accurately. For example, commodity WiFi routers with apparently similar specifications can exhibit widely different performances in benchmarking [17]. Thus, a 'blackbox' modeling approach is preferred where a specific network could be modeled via external measurements even when complete internal specifications are unknown.

To address these limitations, we propose ExBox, an Experience MiddleBox, which continuously explores the capacity of a wireless network.[1] ExBox models network capacity in terms of a new notion of *Experiential Capacity Region* (*ExCR*) that we develop in this work. While *ExCR* is motivated by the general idea of capacity region [65, 51, 45] long been used to model the throughput-based capacity of multi-user wireless networks, it specifically considers experiential capacity helpful to manage QoE of users. Overall, *ExCR* models the set of flows of different application classes that can be accommodated in the network without hurting user experience.

ExBox is capable of modeling diverse set of applications with diverse QoE requirements and is able to capture dynamic environments. It models *ExCR* using a *measurement-driven, online* learning approach that can be applied in a *blackbox* setting, where detailed characteristics of individual network elements are unknown or cannot easily be modeled. ExBox can be deployed as a middlebox in a gateway device in the enterprise setting, as shown in Figure 1. For enterprise networks with different types of access networks, a separate *ExCR* can be learnt for each access point/eNodeB and can be used for implementing network selection.

Roadmap and Contributions: In summary the paper makes the following contributions.

- We motivate the need for a new approach for capacity modeling in wireless networks, based on user experience, and introduce the notion of *Experiential Capacity Region (*ExCR*)* (Section 2).
- We formulate the ExBox approach to learn *ExCR* and layout its different components (Section 3). We discuss the deployment issues for ExBox in real world (Section 4).
- We evaluate ExBox using WiFi and LTE network testbeds involving 10 mobile clients and a single WiFi access point or LTE eNodeB respectively (Section 5). We also perform a scale-up study using the `ns-3` simulator [18] and demonstrate the performance with 50 concurrent client devices (Section 6). Overall, ExBox is able to manage network-wide QoE through admission control decisions with precision and recall rates of $\approx 0.8-0.9$ and $\approx 0.65-0.8$, respectively.

[1]Here, by network we refer to coverage of a single WiFi access point or LTE eNodeB. See more on this in Section 4.4

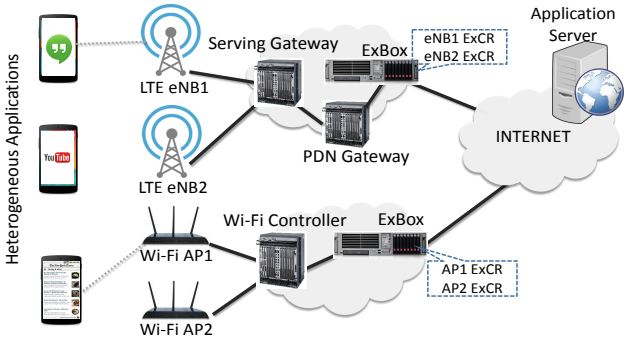


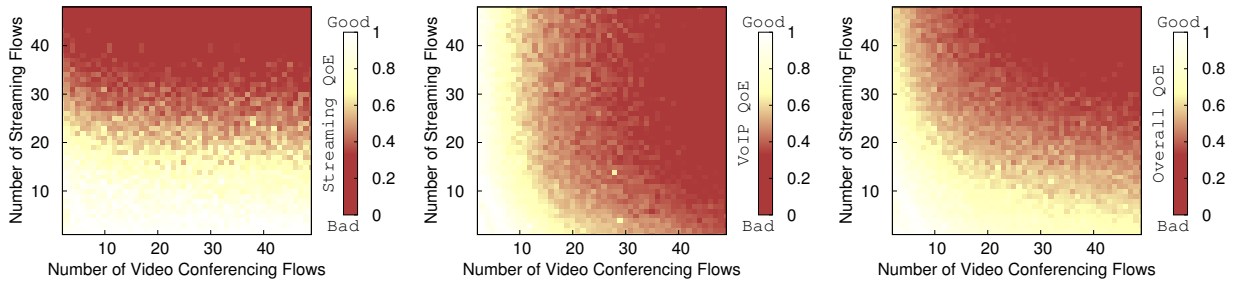
Figure 1: ExBox is implemented as a middlebox collocated with gateway devices.

2. RETHINKING NETWORK CAPACITY

Our broad goal is to model network capacity in terms of QoE across diverse application classes. This goes much beyond the single metric-based network capacity definitions. The general notion of this *experiential capacity* is not entirely new. For example, prior work has developed a similar notion of network capacity for QoE-aware admission control [59], which focuses on increasing the number of simultaneous video flows in 802.11 networks without compromising the QoE. However, studies such as this are unable to model diverse application classes that co-exist in the network [34]. An efficient admission control mechanism needs to capture this mixed traffic.

To further highlight this, we have conducted a simulation study using `ns-3` and packet traces from Skype video conferencing and YouTube high-definition video streaming. The simulation setup is described in Section 6. In a simulated WiFi network, we increase the number of video conferencing and streaming flows simultaneously. For each of these applications, we calculate the per-application median QoE and the average QoE of the network. QoE refers to user-perceived quality of experience. For Skype video conferencing we use PSNR (peak signal-to-noise ratio) of the received video as the QoE metric, while startup delay[2] is used as the QoE metric for the YouTube video streaming. Our simulation does not have a real video conferencing application or a streaming player, we estimate QoE in the following way. QoS is modeled as the ratio of average throughput to delay. We use the IQX model (to be discussed in Section 3.2) to map such QoS values to corresponding QoE values. Figure 2 shows this QoE as a heatmap. The QoE values are normalized for comparison purposes and also to calculate the average QoE of the network which is the mean QoE for all apps in the network (Figure 2c). We observe in Figure 2a that as number of streaming flows increases in the network, the QoE for streaming goes down. However, the increase in number of conferencing flows does not impact streaming QoE significantly. This shows that with ≈ 20 streaming flows, the

[2]Startup delay is the time difference between the events when a video playback is requested and it actually starts playing.



(a) Median QoE for Streaming Flows (b) Median QoE for Video Conferencing Flows (c) Average QoE of the Network

Figure 2: QoE as a function of the number of flows of two applications in the network

network may not be able to accommodate any more streaming flows without hurting QoE, but it may still be able to accommodate ≈ 10 more video conferencing flows. The video conferencing QoE shows a different capacity in Figure 2b. Up to 50 streaming flows can co-exist with ≈ 10 conferencing flows, without hurting the conferencing QoE. When we compute the overall network QoE, i.e. median QoE of all flows, we get the multi-dimensional capacity region seen in Figure 2c. A simple count based admission control [7, 13] will not be able to capture this multi-dimensional capacity as there is no single number to limit all applications. For the capacity region shown in Figure 2c, the maximum count of admissible conferencing flows is ≈ 40, but the maximum count of admissible streaming flows is only ≈ 25.

In addition to application diversity, there is also channel diversity in wireless networks. We conducted testbed experiments to show the impact of wireless channel quality, measured in terms of Signal-to-Noise Ratio (SNR), on the experiential capacity. We use 4 phones connected to the same Wi-Fi AP, where all or some of them have high SNR (placed close to the AP, received signal strength of –30 dBm) and others have low SNR (placed further away from AP, received signal strength of –80 dBm). The phones simultaneously play a YouTube video and record the video startup delay as the QoE metric. Let us assume a desirable value of this QoE metric is 5 seconds. In Figure 3 we see when all phones have high SNR, all 4 video flows satisfy the QoE requirement, however it is not the case when 2 of them have low SNR. The video does not even play in some of the phones when all phones have low SNR. It is interesting to note that the QoE of clients in high SNR location is also impacted when some clients move to a low SNR location. A pure rate-based admission control [51, 9, 2, 22] will not capture this characteristic. It will continue admitting low SNR clients as long as the maximum rate usage allows, without considering the impact on the high SNR clients.

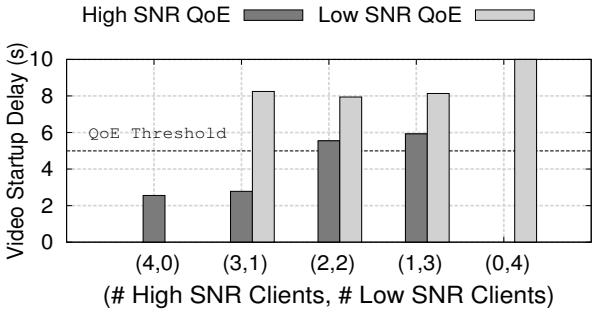


Figure 3: Impact of SNR on Video Streaming QoE.

2.1 Experiential Capacity Region

To accommodate these insights, we need a new model of capacity that can handle QoE of multiple wireless users using diverse applications. One way to do this is to rethink the traditional modeling approach of the rate-based capacity in terms of capacity region [51, 65]. Here, the multi-dimensional region representing the set of all possible achievable rate vectors $< r_1, \ldots, r_n >$ (where r_i is the rate of flow i and n is the number of total flows) models the capacity region. In the following, we extend this notion to develop the *experiential capacity region.*

Assume that there are k application classes and r levels of SNR. SNR levels are formed by simply splitting the range of SNRs possible in r discrete bins. Further, $a_{i,j}$ denotes the number of active flows in the network for class i with SNR of the wireless link in SNR level s_j. For each flow, we assume a 'thresholded' model for QoE, where the QoE metric has been thresholded to *acceptable* or *unacceptable* levels. This thresholding works because QoE often indeed changes sharply from an unacceptable to acceptable level with changes in related QoS parameters. This makes a traffic matrix representing a number of flows of different classes from different users $< a_{1,1}, \ldots, a_{k,r} >$ either achievable or not achievable depending on the whether acceptable levels of QoE for all these flows can be satisfied by the network simultaneously. We call the discrete set of achievable traffic matrices $< a_{1,1}, \ldots, a_{k,r} >$ the *Experiential Capacity Region* (*ExCR*). Our goal in this work is to develop empirical methods to derive this region for a given wireless network.

A careful reader will note that this model of *ExCR* bears similarity to the notion of 'call admission region' or 'admissible load region' concepts developed in connection with broadband networks and mobile cellular networks during the 90s and 00s. Many such studies do consider a mix of different types of traffic, e.g., realtime and non-realtime. See, e.g., [56, 53, 63]. But they still use a single QoS metric:

rate. Further, these studies use an analytical modeling approach that requires full knowledge of system components, e.g., queueing network model, service rates, etc. Instead, we are interested in a blackbox approach where the network must be probed to learn its characteristics. We elaborate on the learning process next.

3. THE EXBOX DESIGN

In this section, we present ExBox, an Experience MiddleBox that continuously explores the *experiential capacity region* (*ExCR*) of a wireless network. Following the description in the previous section, ExBox captures the network traffic matrix as $< a_{1,1}, \ldots, a_{k,r} >$, where $a_{i,j}$ denotes the number of flows for application class i with SNR of the wireless link in SNR level s_j. We restrict ourselves to SNR as the wireless channel metric in this work as this directly influences PHY layer bit rate and bit error rate, and thus has a direct correlation with the overall QoS of the link. In our experiments reported in Section 6 only two levels were found to be sufficient (low and high).

Classifying flows into application classes is the first step in ExBox. Traffic classification has been extensively studied for both legacy and mobile applications [41, 58, 69, 47, 42, 67, 54, 32, 33] and we can leverage this work to classify an incoming flow into an application class (more in Section 4.5). For the rest of the discussion, we assume that the class of a flow is determined. ExBox defines the *ExCR* as set of all traffic matrices of the form $< a_{1,1}, \ldots, a_{k,r} >$ that have acceptable QoEs. Exploring this search space in a brute force manner is: (i) *prohibitive* with multiple users, multiple application classes and multiple SNR categories, and (ii) *indeterminable* in scenarios where network characteristics fluctuate. Instead, ExBox tries to *learn* the boundary of the capacity region by using a binary classifier implemented using off-the-shelf machine learning tools. The learning problem in ExBox lends itself easily to SVM-based solution [40]. This is the approach we take here.

SVM is typically used for supervised deterministic binary classification of multi-dimensional data. Intuitively, SVM uses the training data to construct a hyperplane in the $kr+1$ dimensional space, where k is the number of application classes and r is the number of SNR levels, as above. This hyperplane separates the universe in two separate classes. The hyperplane intuitively defines the boundary of the capacity region. While other supervised classification methods (e.g., decision trees) could be used by ExBox as well, we investigate SVM for its intuitive fit with ExBox. The performance results later will show that it works quite well relative to existing baselines. Regardless, the actual learning technique is not central to the concept of ExBox and can be implemented as a separate module that can be refined as needed.

Overall, ExBox consists of two main components: (i) *Admittance Classifier*: it uses SVM to classify incoming flows as admissible or non-admissible and (ii) *QoE Estimator*: it builds QoE estimates of existing flows on the network-side. We discuss these components next, followed by the software architecture of ExBox.

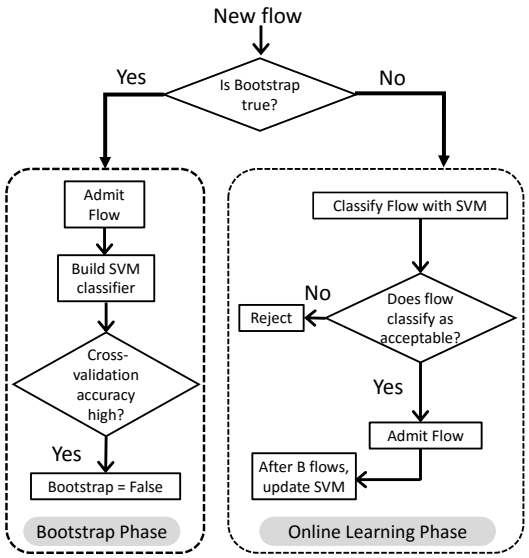


Figure 4: Flowchart for the Admittance Classifier

3.1 Admittance Classifier

One can visualize the boundary of experiential capacity region as a multi-dimensional surface that separates the traffic matrices with *acceptable* QoE from the traffic matrices with *unacceptable* QoE. This forms the basic intuition of the Admittance Classifier, which *classifies* the universe of possible traffic matrices in two regions. Further, in order to adapt to changing network conditions, Admittance Classifier continues to re-learn the capacity region boundary. Our current implementation of the Admittance Classifier uses off-the-shelf Support Vector Machines (SVM) with batch updates for online binary classification. Prior literature [64, 52] has shown suitability of incremental or online learning with SVM. Figure 4 shows the flowchart for Admittance Classifier. The algorithm operates in two phases:

Bootstrap Phase is the phase where ExBox initializes the classifier from zero information. In this phase, ExBox simply *observes* the network. All flows are admitted in the network just as they were in the absence of ExBox. This allows ExBox to learn traffic matrices with both acceptable as well as unacceptable QoEs. It starts with $a_{i,j} = 0, \forall i \in 1, \ldots, k$ and $\forall j \in 1, \ldots, r$. As a new flow m of class c and SNR s_ℓ joins the network, $a_{c,\ell}$ is incremented by 1. Next, the Admittance Classifier estimates the QoE of entire set of active flows (including the new one) as $Y_m \in \{+1, -1\}$ using the IQX hypothesis [44]. This is learnt using the QoE Estimator described in the following subsection. A value of +1 for Y_m denotes that if flow m is admitted then still the new traffic matrix will have an acceptable QoE. Similarly, a value of -1 for Y_m denotes that upon admitting the flow m the QoE of one or more flows will be unacceptable.

Each incoming flow can impact the QoE of all ongoing flows. ExBox defines the tuple $X_m = (< a_{1,1}, \ldots, a_{k,r} >, (c, \ell))$ to indicate a newly arrived flow m of application class c and SNR s_ℓ when the current traffic matrix is $< a_{1,1}, \ldots, a_{k,r} >$. The tuple (X_m, Y_m) describes the influence of such a new flow on the

existing traffic matrix, where Y_m classifies this influence in one of two possible classes as described earlier. Over time, Admittance Classifier learns many such (X_m, Y_m) tuples and uses them to train the SVM. Note that in the bootstrapping phase, ExBox does not perform any admission control – all flows are admitted regardless of any impact on QoE of other flows. This phase is needed in order to collect a diverse training set for Admittance Classifier. We show in the evaluation that this phase is typically not too long.

As ExBox builds the Admittance Classifier from the training set, it also performs n-fold cross validation on the training set periodically. This is done by splitting the *training set* into n random subsets, out of which the Admittance Classifier is trained using $n-1$ subsets and is tested with the remaining subset. To test the Admittance Classifier when a flow p arrives, an X_p tuple is created and the $\hat{Y_p}$ label is computed for X_p from the trained SVM model. This estimated $\hat{Y_p}$ is then compared with the Y_p observed in the network to compute accuracy. This process is repeated for all the n subsets. When a predefined accuracy threshold is reached, ExBox stops the bootstrapping phase. The Admittance Classifier becomes operational, enters the Online Learning Phase and starts classifying each incoming flow as $+1$ (admissible) or -1 (inadmissible).

Online Learning Phase executes in rounds and each round is determined by a 'batch' of flows. After a batch-size B number of flows has been admitted, ExBox re-computes the Admittance Classifier with all the (X_m, Y_m) observed so far. Additionally, if the new batch contains a traffic matrix $< a_{1,1}, \ldots, a_{k,r} >$ which has been seen in the past, the observed QoE for that traffic matrix is replaced with the new observed QoE. The Online Learning Phase enables ExBox to adapt to changes in network and client population. For example, if clients move farther from the WiFi AP, the QoE of their new flows may worsen. Additionally, popularity of Self Organizing Network (SON) solutions [1, 11, 21] enables operators to dynamically change network configuration and, in turn, change network capacity. In such cases, the Admittance Classifier may classify new flows incorrectly and needs to re-learn the capacity region. The online phase enables this possibility.

3.2 QoE Estimator

The Admittance Classifier described above assumes knowledge of QoE for each application class. In a real network, obtaining QoE values from the users may not be practical as it involves deploying custom applications [26] or modifying user devices [38]. A solution that requires end-device modifications has a significant barrier for adoption, given the BYOD trends prevalent in enterprise and campus networks today. At the same time, user experience can be most accurately measured at end device as different applications react differently to network characteristics.

Thus, ExBox uses an intermediate modeling-based approach where the QoE is estimated on the 'network-side' but with some help from client devices introduced in the network specifically for model training purposes. Such training needs in ExBox are fairly modest and can be accomplished by the network administrator. Based on this initial training, ExBox can estimate QoE of future flows from passive network measurements on the network-side. Several approaches have been proposed in the past [37, 28, 29, 26, 44] to measure application QoE from network measurements. Among these, the IQX hypothesis [44] proposes a generic relationship between QoE and QoS for multiple applications. This is useful to network administrator, who can measure the network's QoS but may not have knowledge of application level experience. The basic formulation of the IQX hypothesis is as follows:

$$QoE = \alpha + \beta.e^{-\gamma.QoS}, \quad (1)$$

where QoS and QoE are suitable metrics modeling dominant QoS or QoE. The parameter γ regulates the contribution of QoS in QoE. The dominant QoS parameter varies across applications, e.g. latency for web browsing and VoIP, throughput for streaming, etc. The IQX hypothesis estimates different QoE metrics for different applications. For instance, QoE of web browsing is measured in terms of *Page Load Time*, QoE of VoIP applications is measured in terms of *Peak Signal to Noise Ratio (PSNR)*, QoE of video streaming is measured in terms of *Startup Delay*. ExBox uses the IQX hypothesis to estimate per-application QoE. From a training device in the network, controlled experiments can be done to find the best fitting α, β and γ for a given application. ExBox uses pre-defined thresholds [39] to map the QoE of each flow m to $Y_m \in \{+1, -1\}$. Section 5 presents performance evaluation of the IQX model in a real testbed.

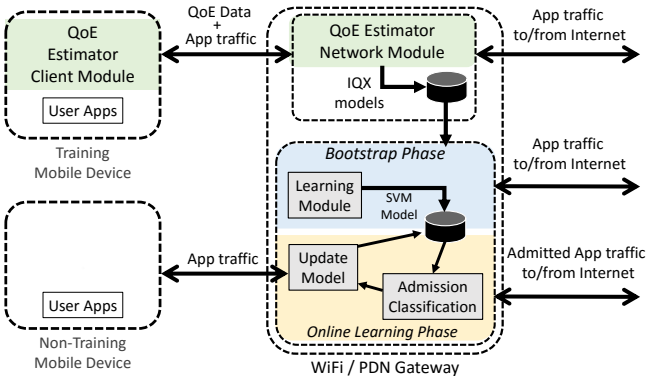


Figure 5: Software Architecture of ExBox

3.3 Software Architecture

Figure 5 shows the software architecture of ExBox. ExBox learns the IQX model [44] for each application class via an initial training with actual measured QoE values on the client device. For this training the application class is identified via established traffic classification methods [41, 58, 69, 47, 42, 67, 54, 32, 33][3].

A single training client device is used to record per application QoE information via direct measurement using instrumented apps, and share the data with the QoE Estima-

[3]Note that the proposed traffic classification solutions work for encrypted traffic as well.

tor module in the network. The QoE Estimator fits an IQX model per application class and stores it for future use by the Admittance Classifier. Finally, once relevant IQX models are built, any regular flow (i.e., from a non-training mobile device) is passed to the Admittance Classifier and steps described in Section 3.1 and Figure 4 are executed. The SNR level of a flow is computed from the SNR reported by the WiFi access point (AP) or Channel Quality Indicator (CQI) collected by the LTE base station (eNodeB).

4. QOE MANAGEMENT WITH EXBOX

We now discuss how ExBox can be used for QoE management and address various practical operational challenges.

4.1 Network Selection

ExBox, as defined in previous section, can be used to explore the experiential capacity region of a given wireless network. In enterprise networks where WiFi and LTE Small Cells may co-exist, ExBox can be used to simultaneously explore the capacity regions of both networks. It can bootstrap and learn two separate Admittance Classifiers. Further, when a new flow comes in, it can be used to decide whether the flow can be admitted in either of the available networks. If the answer is $+1$ from both classifiers, ExBox can select the best suited network based on how much 'inside' the capacity region the new test point is. There is a straightforward mechanism to do this in SVM by evaluating how far away from the separating hyperplane the test point lies. ExBox can operate from within the infrastructure with limited support from a training device in the network. In the case of hybrid WiFi-LTE networks, ExBox can be collocated on the LTE PDN gateway, which can view both WiFi and LTE traffic and perform network selection.

4.2 Admission Control

Before ExBox can make the admission decision for a flow, it needs to know the application class of that flow. Existing traffic classification solutions [41, 58, 69, 47, 42, 67, 54, 32, 33] analyze the first few packets of the flow to decide which category it belongs to. Thus, a flow needs to be *admitted* briefly before any admission control decision is made. Once ExBox makes the admission decision, it can be executed in multiple ways. In a single network scenario, if ExBox determines the flow to be admissible it goes through the network. However, if a flow is determined to be inadmissible, then the flow can be (i) discontinued from the gateway or (ii) admitted in a low priority access category,such as in 802.11e. In case a flow is not admitted the user is notified by a message (e.g., Smart TVs already reject flows because of insufficient bandwidth [3]). In a wide multi-cell deployment, ExBox determines the best network suited for an incoming flow and moves the flow to the accepting network. In multi-cell WiFi networks, flows can be easily migrated from one AP to another through the WiFi controller [6]. In LTE networks, the serving gateway (SGW) can assist mobility across 3GPP entities. In hybrid LTE-WiFi deployments, the 3GPP IP flow mobility standard [14] allows for seamless offload between LTE and WiFi. Note that executing ExBox's decision of rejecting flows may leave some users disconnected or unable to use intended applications. This is governed by the *admittance policy* of the network administrator. They can choose to allow best-effort connectivity to as many flows or maintain their promise of good QoE to certain users at the cost of disappointing other users. ExBox aids them in making a more informed choice.

4.3 Dealing with Network/App Dynamics

While the Admittance Classifier defined in the previous section focuses on admission control, a flow's behavior may change after being admitted. One reason is that the application adapts to changes in the end-to-end network throughput or delay. This may lead to change in QoE of the flow, e.g. YouTube can downgrade the video quality from high-definition to low-definition. Another reason could be user mobility. The wireless link quality between a device and the WiFi AP (or LTE eNodeB) can change depending on the distance of device from AP, multi-path fading, etc. If a flow is admitted in the network when the device is close to the AP, its QoE may deteriorate when the device moves farther from the AP. Additionally, in a contention based system, such as 802.11 WiFi, a low rate device can reduce the rates experienced by others (throughput fairness). In such cases, ExBox needs to revise its decision of admitting such a flow. For this purpose, ExBox periodically polls the network to check if the flow characteristics (throughput, delay, loss) have drastically changed or if the SNR levels of any device have changed. It then constructs a new tuple X_m from the current traffic matrix and computes Y_m from the Admittance Classifier. If the new Y_m is -1, this flow is considered inadmissible and offloaded to another network or discontinued, as governed by the *admittance policy* discussed above.

4.4 Scaling to Large Networks

In this work we modeled only one access device or cell (WiFi AP or LTE eNodeB) to limit the state space to be considered. In a typical enterprise or campus network, multiple WiFi or LTE cells are deployed to cater to the large population. To better serve all users, the network administrator can scale up the ExBox system to simultaneously learn classifiers for multiple cells. Being located on the WiFi controller or PDN gateway enables ExBox to have this wide view of the network. The classifier is a $kr + 1$ dimension vector for k application classes and r SNR classes. One such classifier needs to be periodically learned per cell.

One can argue that the single access device model limits the applicability of ExBox, as it cannot consider interference arising from neighboring cells. However, we expect that in most enterprise deployments the channel/cell planning is done carefully enough that the influence of such interference actually affecting capacity is minimal. On the other hand, this simplification is computationally convenient. In our future work, we plan to perform evaluations on larger networks to determine whether any refinement of the basic ExBox will be necessary.

In ExBox can be deployed widely, it is also possible to

share IQX models over different networks of similar characteristics. This will reduce the training effort substantially and/or will help in more robust model creation, e.g. influence of device heterogeneity on IQX models can be considered. Existing work on QoE estimation [26, 28, 29, 38] does not consider such heterogeneity in their models. This is an interesting direction of future work.

4.5 Flows, Apps and Users

We have used ExBox to do flow-based admission control in this work. However, many modern applications use multiple flows in the same app. For example, YouTube uses separate flows to play the main video and to load video recommendations. Thus, flow-based admission control may not always be appropriate and an app-based admission control may be preferred. ExBox can be extended to handle such cases through an app-based admission control. This can be achieved by leveraging the flow classification tool to identify the dominant flow that determines the app's QoE [60, 38]. For example, in a video streaming app, the flows carrying video data and control are dominant although other flows carrying app-analytics data or advertisements may be present. The admission control now can use a heuristic that admits all flows for that app if the dominant flows are admitted. This is an interesting direction of future work.

In the same note, a single user may use multiple apps concurrently, that are to be considered separately. It is possible that some apps are admissible and others are not for the same user. The inadmissible flow/app may be discontinued from the WiFi controller or PGW, or selectively offloaded to another network. Distributing flows over multiple wireless interfaces or wireless networks has also been studied in prior research [46, 50, 35].

5. EXPERIMENTAL EVALUATION

In this section we evaluate ExBox using QoE traces collected from a mobile testbed. We first explain how we collect the traces and then present results from our trace-based evaluation. We evaluate the Admittance Classifier and the QoE Estimator modules separately over the same testbed. In the next section we evaluate the entire ExBox solution through large-scale `ns-3` simulations.

5.1 Testbed Setup

Mobile clients: We use 10 Samsung Galaxy S6 phones, running Android OS, as clients or user equipments (UEs) to generate different types of flows. The clients either connect to a WiFi access point or to an LTE eNodeB that are deployed as a part of our testbed.

WiFi Network: We host the WiFi access point on a laptop whose WiFi interface can be switched into access point mode and serve as a WiFi hotspot. We use a well provisioned laptop (an HP ZBook with 16 GB memory and quad-core Intel i7 processor). The Linux utility `hostapd` is used to configure it as a WiFi hotspot. This setup enables us to capture network traffic from the UEs connecting to the hotspot at the very first hop. It also allows us to shape traffic parameters (throttle bandwidth, introduce delay, losses) using Linux utilities `tc` and `netem` on the laptop. We use WiFi channel number 5 that is interference free in our lab area. The WiFi testbed setup is shown in Figure 6a. We run capacity tests on the WiFi link and observe 20 Mbps `iperf` UDP throughput and $\approx 30-40$ ms `ping` latency. The low throughput is an artifact of the WiFi driver on the laptop but it does not impact our evaluation as ExBox does not make any assumptions on the maximum bandwidth.

LTE Network: We have an LTE small-cell testbed in our lab. It consists of LTE core network as well as eNodeB. We use ip.access E-40 [15] as our eNodeB. In the core network we use licensed OpenEPC [19] software stack. The core network consists of 3GPP [23] standard-compliant EPC components such as MME, SGW, PGW and HSS. Each component runs in a Linux-based virtual machine. For each UE, we have our own user subscription information (*e.g.* IMSI) programmed on a SIM card. We are able to simultaneously connect 8 UEs to the E-40 eNodeB. This is the maximum bound of the current version of software in ip.access E-40. This is the best we can currently do in a lab-grade EPC deployment[4]. We run the packet capture (`tcpdump`) and traffic shaping tools (`tc`, `netem`) in the virtual machine hosting PGW. The LTE testbed setup is shown in Figure 6b. `iperf` and `ping` tests on the LTE link show more than 30 Mbps capacity and $\approx 30-40$ ms latency.

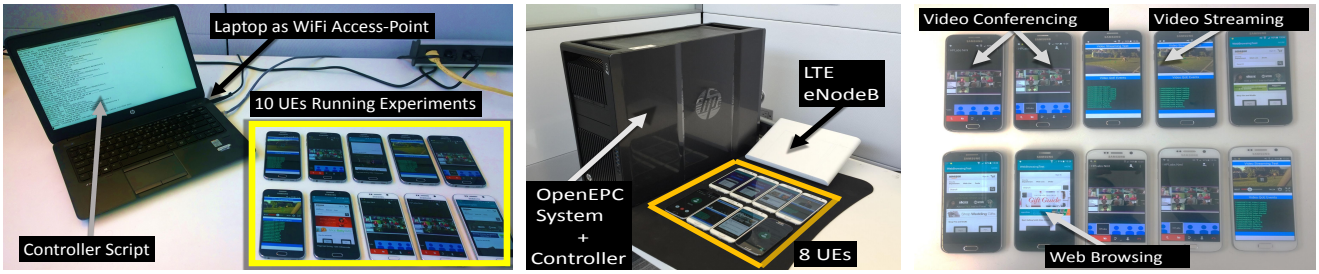
5.2 Experiment Methodology

Client Apps: For our experiments we developed two Android apps and a UI automation script to generate web, video streaming and video conferencing flows. We chose these three application classes because their respective QoE is influenced by different underlying network attributes. Each app also records the corresponding QoE ground truth. None of the apps require root privilege.

- **Web Browsing App:** The web browsing app embeds a `Webview` client acting as the browser which continually loads a specific sequence of webpages. The app records *page load time* as the QoE metric. For benchmarking we focused on similar sized webpages: we use mobile websites of Amazon, BBC and YouTube. No video is played on YouTube as we just load the homepage. The app clears the browser cache after a page load is over to make sure a fresh copy of the page is fetched every time.
- **Video Streaming App:** The video streaming app embeds a YouTube–video player that repeatedly plays a video. We use a 2–minute video clip from YouTube that is available in HD-format (720p).[5] We use the YouTube Android Player API [25] that captures video playback events like *video loaded*, *video started*, *video buffering*, *video playing* and so on. Every time a video is played the app logs all

[4]Open Air Interface (OAI), another popular EPC implementation, currently supports only a couple of UEs stably.

[5]The median video clip playback length in YouTube is ≈ 2.5 sec [61].



(a) Wi-Fi Testbed with 10 UEs (b) LTE Testbed with 8 UEs (c) Apps running on UEs

Figure 6: WiFi and LTE testbed setup for evaluation of ExBox

such player events with corresponding timestamps. The video *startup delay* is used as the QoE metric. This is measured as the time it takes the video to start playing from the instant the video is requested [68]. We rarely observe additional video stalls as the most of the content is downloaded during the initial start-up delay period. Hence we do not consider buffering ratio or buffering rate or average bit rate as the QoE metric in our setup, however they can be used in principle.

- **Video Conferencing App:** For video conferencing, we use Google Hangouts along with a UI automation script. One peer of the Hangouts call runs in a cloud server and the other peer on the phone initiates a call. For tasks such as choosing the contact, placing a call and ending the call, we use the AndroidViewClient library [5] that can perform UI automation over Android Debug Bridge (`adb`) tool[4]. This allows us to control the Google Hangouts sessions on the phone from the client controller described below. On the cloud peer a prerecorded video file is played through a virtual camera software (e2e Vcam software [12]) and is put as camera input to the Google Hangouts call. The video received on UE is screen recorded. The QoE of video is measured in terms of PSNR (Peak Signal-to-Noise Ratio) in dB [66]. PSNR measures the relative degradation in the frame quality between the sent and received videos.

Client Controller: Figure 6c shows all 10 UEs running a combination of the apps defined above. The apps are scheduled by a central controller that resides in the access-point (laptop), in case of the WiFi network, or PGW, in case of the LTE network. The clocks on UEs and the controller are synchronized with an NTP server. The controller communicates to the UEs, wirelessly, using the `adb` tool. The controller can start or stop any application on any of the UEs, including the Google Hangouts calls. It also periodically runs a `ping` command to the UEs and logs corresponding RTT values. It can also capture network traffic using `tcpdump`. The controller gives a systematic interface to conduct experiments on the mobile testbed. For example, in order to measure the ground truth for experiential capacity we need to run different number of web, streaming and video conferencing flows. The controller takes the traffic matrix ($\#web$, $\#stream$, $\#videoconf$) as input and launches corresponding number of apps in terms of the number of flows belonging to the three different classes on a random subset of the UEs.

Traffic Patterns: We use two schemes to generate these traffic matrices:
(i) Random: This scheme considers completely random flow arrival/departure traffic pattern. Thus ($\#web$, $\#stream$, $\#videoconf$) can change randomly and drastically.
(ii) LiveLab: To create more realistic traffic pattern we mined Rice University's LiveLab dataset [16]. This dataset has 34 users and application usage logs with $\approx$1.4 million entries. Each entry consists of the application name and the time it is triggered. It also contains the length of time the application was used. We filtered the set of application to a few limited ones that specifically indicated web (e.g., Opera, Safari, Chromium etc), video streaming (e.g., YouTube, Netflix etc.) or video conferencing activities (e.g., Google Hangouts, Skype etc.). Using this set of apps and their start and end time, we were able to compute the number of flows of each application class that are active simultaneously. This provided us with the traffic-matrix. We converged to $\approx$1700 traffic matrices of ($\#web$, $\#stream$, $\#videoconf$), which are sorted in chronological order. Several traffic matrices do repeat in this set. In our experiments, we only consider those traffic matrices where total number of flows is less than 8 (LTE Network) or 10 (WiFi Network) due to limitation in our testbed. We assume that all users are present in the same network. We place all devices in high SNR locations. We evaluate diverse SNR scenarios later, using simulations.

5.3 Evaluation Results

We first evaluate the performance of Admittance Classifier. We run all the traffic matrices observed in the traffic schemes defined above, multiple times in each testbed. For every traffic matrix, we record *ground truth* QoE of each application running on each UE. If the QoE falls beneath a certain threshold (e.g., 3 secs page load time in case of web browsing) we deem that particular flow to be inadmissible in that traffic matrix. This gives us a large dataset of traffic matrices consisting of flows with acceptable as well as unacceptable QoE. Thereafter, we use a Python based implementation of Admittance Classifier, which does a trace based learning of *ExCR*. For each traffic pattern, Admittance Classifier trains the SVM with an initial set of traffic matrices (bootstrapping phase). In our experiments we observe

that bootstrapping can be done with ≈ 50 samples, providing 0.5 – 0.6 precision and recall at the end of bootstrap. Note that each sample in our evaluation represents a traffic matrix. As flows enter and leave the network, a new traffic matrix is recorded and added to the training set. Depending on the activity level in a network 50 samples could be observed over few hours or over a day. In networks with diverse and active users, different traffic matrices can be observed quickly, speeding up the bootstrapping phase in networks where admission control is most effective.

After bootstrapping phase, Admittance Classifier enters the Online Learning Phase, where it takes admission decision for each incoming flow. This is the testing phase. Note that Admittance Classifier also updates its model after a batch of flows has been admitted. We consider a batch size of 20 samples for WiFi experiments and a batch size of 10 samples for LTE experiments. We will present sensitivity results for batch size later in this section.

We also implement two commonly used schemes to serve as baselines in our evaluation.

(i) **RateBased**: This scheme considers a purely rate based admission control. This approach is used exclusively by many vendors (Cisco [9], Rukus [22]) and industry software (Microsoft [2]). It assumes the network capacity as C and rate requirements of each flow f as c_f. A new flow g is admitted in the network only when $C\text{-}\sum c_{\text{ongoing flows}} \geq c_g$. In our experiments we set C as the maximum UDP throughput measured in our WiFi and LTE testbeds.

(ii) **MaxClient**: This scheme admits up to a maximum number of flows and rejects all flows beyond that. In our experiments we set the maximum client limit to 10, as in [7] (Aruba Wireless) or [13] (IBM). This is also the maximum number of clients in our testbed.

Latency benchmarks: We benchmark the latencies incurred by the Admittance Classifiers for our python based implementation of ExBox, RateBased and MaxClient algorithms. The latency is measured as the time interval between the instant a new flow arrives and the admission decision (admit or reject) decision is taken on it. We use a HP ZBook equipped with 16 GB memory and a quad-core i7 processor for our benchmark study. For both RateBased and MaxClient the latencies are negligible ($\leq$ 2 ms median latency). For ExBox, the median latency observed is ≈ 5 ms. Both RateBased and MaxClient do not involve an explicit training phase. However ExBox involves training Admittance Classifier (SVM based) both at the bootstrap phase and during online updates. Training the Admittance Classifier for ExBox with 50 samples takes ≈ 360 ms median latency. The training latency increases to more than 2 seconds when 1000 samples are considered. This is an artifact of our current implementation (e.g., choice of SVM kernel) and platform/libraries. For example, libraries such `libsvm` written in C have more a optimized implementation. A recent work on SVM optimization [36] actually argues that with increasing training size (number of samples) the latency incurred in building the classifier might actually *decrease* with the right implementation. Investigating SVM implementation options is somewhat tangential to our work. But evidently latency of the learning technique is a bottleneck in ExBox.

Macro results: We compare ExBox, RateBased and MaxClient using three well-known performance metrics: viz., *precision*, *recall* and *accuracy*. *Precision* is defined as the ratio of correctly admitted flows to the number of admitted flows, while *recall* is defined as the ratio of correctly admitted flows to the number of flows that could have been admitted. *Accuracy* refers to the overall fraction of correct decisions made (admit or reject). Intuitively, a high value of precision indicates that ExBox makes few mistakes in preserving the network QoE. However, a conservative admission control approach will also preserve QoE quite well (e.g., admit very few flows and reject everything else). The recall metric will catch such overly conservative behavior.

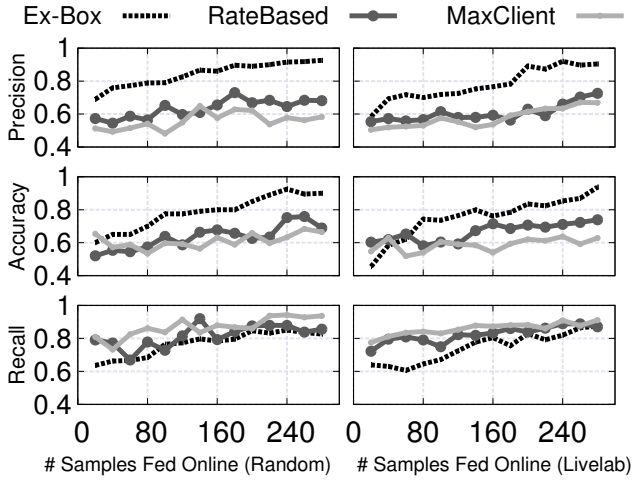


Figure 7: WiFi testbed results for Random Traffic and LiveLab traces, compared with baseline

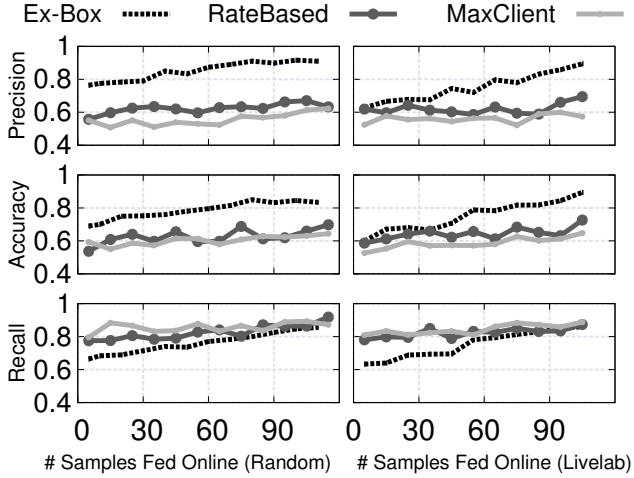
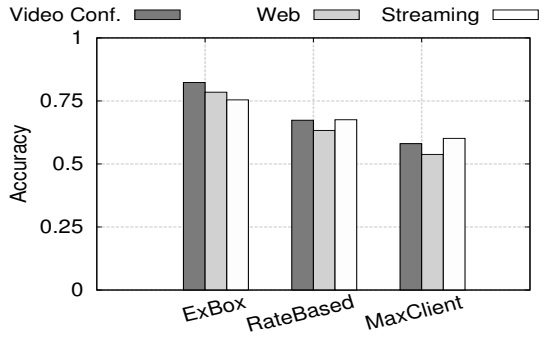


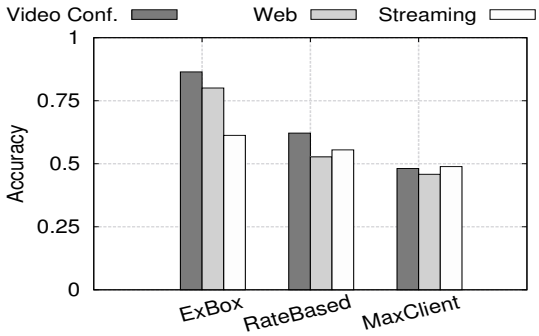
Figure 8: LTE testbed results for Random Traffic and LiveLab traces, compared with baseline

Figure 7 compares ExBox, RateBased and MaxClient with both Random and LiveLab traces on the WiFi testbed.

Compared to RateBased and MaxClient, ExBox always has higher precision ($\geq$ 0.8 mostly) and accuracy ($\geq$ 0.85 mostly), but a lower recall ($\leq$ 0.85 mostly). However, with more training (i.e., number of samples), its recall catches up with other approaches. In the context of admission control, accuracy is an important metric as it captures flows that have been correctly accepted and flows that have been correctly rejected, in order to maintain the QoE of existing users. ExBox has an accuracy of 0.6 with initial bootstrapping and it grows to 0.8 after 160 samples. At the same time, accuracy of RateBased and MaxClient stays less than 0.7. Figure 8 shows similar results for the three approaches on the LTE testbed. Across both networks, the random scheme shows better results than LiveLab traces, as random scheme provides more diverse training.



(a) Accuracy in WiFi Network



(b) Accuracy in LTE Network

Figure 9: Accuracy per application type

Micro results: We further compare the performance of ExBox, RateBased and MaxClient for each application class. Figure 9 compares the accuracy of video conferencing, web and streaming for Random traffic on both the networks. Here, accuracy is computed as the fraction of flows of each application which were correctly *admitted* or *rejected*. ExBox performs better than other baselines for all applications. Its accuracy for streaming is closer to RateBased as streaming is a rate-sensitive application. But for delay-sensitive applications such as web and video conferencing, ExBox performs significantly better.

Sensitivity to batch size: While the macro-results shown above consider a single batch-size for updating Admittance

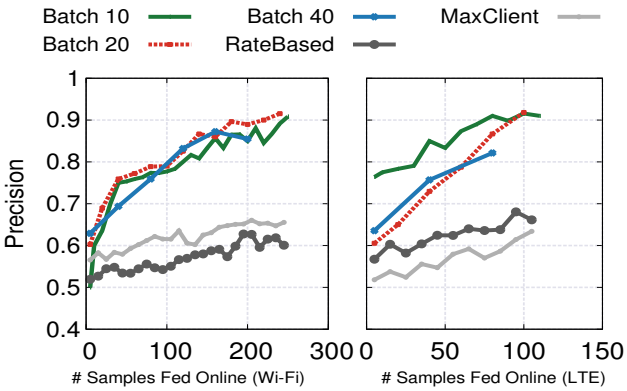


Figure 10: Sensitivity to batch size: WiFi and LTE

Classifier, we also tested Admittance Classifier with varying batch-sizes. Figure 10 shows the precision for batch sizes of 10, 20 and 40. The Admittance Classifier performance is sensitive to the batch size. Size 20 works better for the WiFi network while size 10 works better for the LTE network. The RateBased and MaxClient approaches are not sensitive to batch size as they do not have any online updates. Thus, we only present one plot for each of them. Note that Admittance Classifier has higher precision than others for all batch sizes.

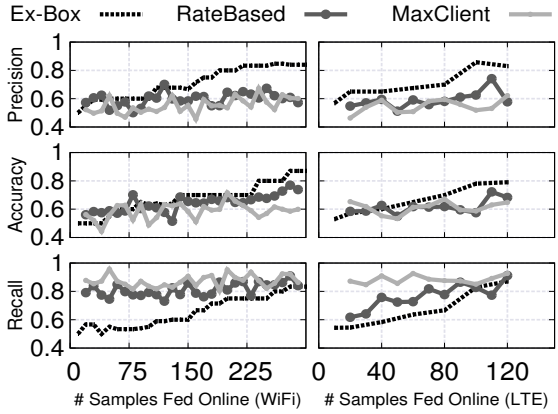
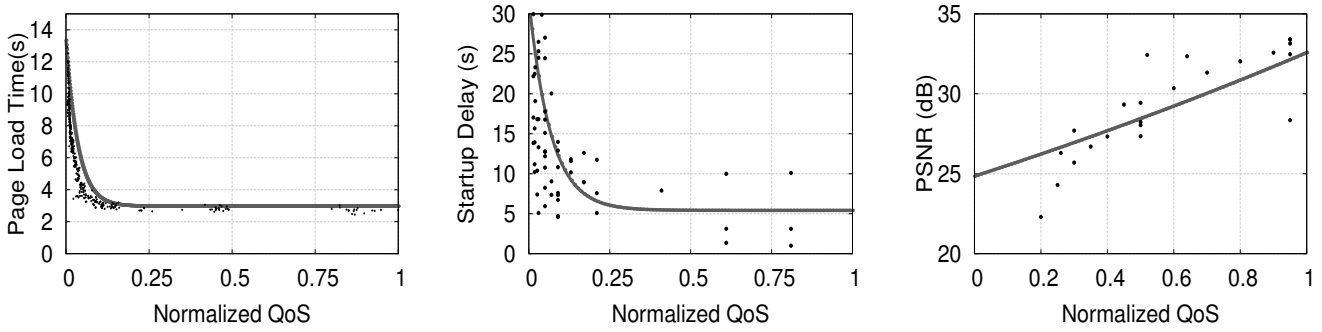


Figure 11: Admittance Classifier performance when network behavior changes in WiFi and LTE testbeds

Adapting to network changes: In the macro results, we make sure that the wireless network does not experience any fluctuations in data rate and latency. Now, we evaluate Admittance Classifier when network behavior fluctuates. We run the traffic matrices from both Random and LiveLab traffic in a throttled WiFi and LTE network (using `tc` utility). In Figure 11, we bootstrap the classifier using 10% data points from the unthrottled network and then start testing the model from a traffic-shaped network with a latency of 200 ms. Note that the initial performance is extremely poor (precision $\approx$ 0.5), however with subsequent batches the model adapts to the network condition. With 200+ data points in WiFi it catches up to precision level of



(a) Web QoE IQX (RMSE 1.37s) (b) Video Streaming QoE IQX (RMSE 3.64s) (c) Video conf. QoE IQX (RMSE 4.462dB)

Figure 12: Fitting the IQX Equation for Web, Video Streaming and Video Conferencing flows

≈ 0.8. ExBox over LTE adapts faster. This shows the necessity of the online updates and how ExBox adapts to fluctuating environments [57].

Estimating QoE using IQX: As discussed in Section 3, ExBox uses IQX hypothesis to estimate per-application QoE. We now show empirical evidence of the same. We use `tc` utility to vary the data rate from 100 Kbps to 20 Mbps and latency from 10 ms to 250 ms in our WiFi network setup. For each data rate-latency profile we run each of the three applications defined earlier 10 times on a single client. This is the *training device* shown in Figure 5. In every run we record the relevant QoE metrics on the client and the QoS metrics on the controller. In our experiments, QoS is modeled as the ratio of average throughput to delay.

Figure 12 shows the relationship between QoS and achieved QoE for the Web Browsing app, Video conferencing app (Google Hangouts) and Video Streaming (YouTube) app. For each of these applications, we fit the IQX equation (1) and evaluate the parameters α, β and γ by using the *least squares* approach. The three different fits (web, video streaming and conferencing) show the differing nature of these applications. The IQX relationship learnt here is used by ExBox to estimate QoE purely from the network traces in the simulations in next section. This experiment demonstrates the suitability of the IQX hypothesis. Different applications may have different IQX models that the network administrator needs knowledge of. However this exercise is not prohibitive as the number of popular application classes within a network (say, an enterprise) is limited. A more rigorous study of IQX is part of our future work.

6. SCALE UP STUDY

The E-40 eNodeB in our testbed only supports a maximum of 8 users and the WiFi card on the laptop could not handle more than 10 users. However, real campus and office networks can have up to 50 concurrently active users [8]. To evaluate ExBox in more populous networks, we perform a scaled up study through network simulator `ns-3` [18]. We simulate both WiFi and LTE networks using `ns-3` and feed in real application traces to generate traffic on these networks. As the simulation progresses, we collect QoS information and compute QoE using IQX. The network simulation setup and traffic generation are described below.

6.1 Simulation Setup

WiFi network: We simulate an 802.11n 5GHz Wireless LAN with varying number of clients connected to the Internet through an access point (AP). The AP is connected to three remote servers (Server 1, Server 2 and Server 3) through a 1 Gbps link and 0.3 ms delay. We do not consider wide area network delays in the simulation, as we focus on the WLAN dynamics alone. With the increasing penetration of content delivery networks, last-hop networks are becoming more dominant in application performance. Similar to the testbed study, these simulations also consider three application classes: video conferencing, web browsing and video streaming. So the traffic matrices considered here are in the space <$a_{\text{web}}, a_{\text{streaming}}, a_{\text{conferencing}}$>. For each such traffic matrix, $a_{\text{web}} + a_{\text{streaming}} + a_{\text{conferencing}}$ wireless clients are simulated in the network, each with one flow. One remote server is assigned to each of the three application classes. All clients for one application class connect to the same remote server. Note that we do not split the application classes in different SNR levels as all clients are in high SNR location, unless mentioned otherwise.

LTE network: We simulate an indoor LTE network in `ns-3` with an eNodeB having 23 dBm transmit power, connected to the Internet through a Packet Data Network Gateway (PGW). The PGW connects to three remote servers, each supporting one application class. For each traffic matrix <$a_{\text{web}}, a_{\text{streaming}}, a_{\text{conferencing}}$>, $a_{\text{web}} + a_{\text{streaming}} + a_{\text{conferencing}}$ LTE user equipment (UE) nodes are simulated in the network, each with one flow. All clients for one application class connect to the same remote server. All clients are in high CQI location, unless mentioned otherwise.

6.2 Traffic Generation

In our simulations, all the wireless clients connect to one of three remote servers either over WiFi or LTE network. To simulate a realistic mix of application flows in the network, we need a traffic generator which can accurately simulate diverse application behavior. As `ns-3` does not sup-

port sophisticated traffic generators, we leverage real packet traces. In the topologies described above, each of the three simulated remote servers is configured to have a Tap interface [24]. This tap interface connects to the local host through a bridge. We feed in real traffic traces of different application types to these tap interfaces and eventually, into the simulated wireless network. For Video Conferencing, we record a packet trace of an ongoing Skype Video Call and extract 30 seconds of the trace. Similarly, for web browsing and video streaming, we capture the packet trace while loading BBC webpage on Chrome browser and while playing high-definition videos on YouTube, respectively.

For a traffic matrix <a_{web}, $a_{\text{streaming}}$, $a_{\text{conferencing}}$>, the traffic generator creates a_{web} instances of the BBC packet trace, merges them and injects the merged trace to the tap interface of Server 1. Similarly, it also creates $a_{\text{streaming}}$ ($a_{\text{conferencing}}$) instances of the YouTube (Skype) trace, merges them and injects the merged trace to tap interface of Server 2 (Server 3). We use the `tcpreplay` suite to modify the packet headers in the real traces to comply with `ns-3` IP and MAC address space. Note that we only use the downlink flows in our simulation. So for Skype, we only capture one-way video conferencing traffic.

For brevity, we do not present scale up studies for all traffic schemes discussed in previous section on all wireless networks. Instead, we present one scheme and one network for each scale up study. The results are similar in other schemes.

6.3 ExBox in Mixed SNR Scenarios

All the experiments in the testbed have all the phones in high SNR location. Since adding the SNR diversity in the traffic matrices explodes the search space, we perform these tests in simulation. We run the LiveLab traffic trace over the simulated 802.11n WLAN. Additionally, for each new flow, we randomly position the client in a high SNR ($\approx$ 53 dB) or a low SNR ($\approx$ 23 dB) location. This generates a large dataset ($\approx$ 21000 samples) of X_m: <$a_{\text{web,high}}$, $a_{\text{web,low}}$, $a_{\text{streaming,high}}$, $a_{\text{streaming,low}}$, $a_{\text{conferencing,high}}$, $a_{\text{conferencing,low}}$, c, l> for flow m of class c and SNR level s_l. The $Y_m \in \{-1, +1\}$ is computed by using the IQX model. We compare ExBox, RateBased and MaxClient on this large dataset of mixed SNR flows. Figure 13 shows the precision results for all three approaches with varying batch sizes. ExBox gives more than 0.8 precision. This is because we could bootstrap with a larger training set here, which is the case in populous, diverse networks. Also note that the batch sizes in this case are larger than the ones in testbed, implying less frequent updates to the model. The batch updates further improve the precision to 0.95. The precision for RateBased stays around 0.65 throughout.

6.4 ExBox in Populous Networks

We simulate multiple traffic matrices with $a_{\text{web}} \in [0, 50]$, $a_{\text{streaming}} \in [0, 50]$ and $a_{\text{conferencing}} \in [0, 50]$. For each traffic matrix, the simulation runs for 16 seconds and we record the QoS observed by each flow. Similar to the testbed study, each run is captured as the tuple X_m :<a_{web}, $a_{\text{streaming}}$, $a_{\text{conferencing}}$, j> and label Y_m, where $j \in \{1, 2, 3\}$. Here Y_m represents the QoE (calculated from IQX) of flow m of application class j in the given traffic matrix. Note that X_m has fewer dimensions here than previous simulations as all clients are in high SNR locations.

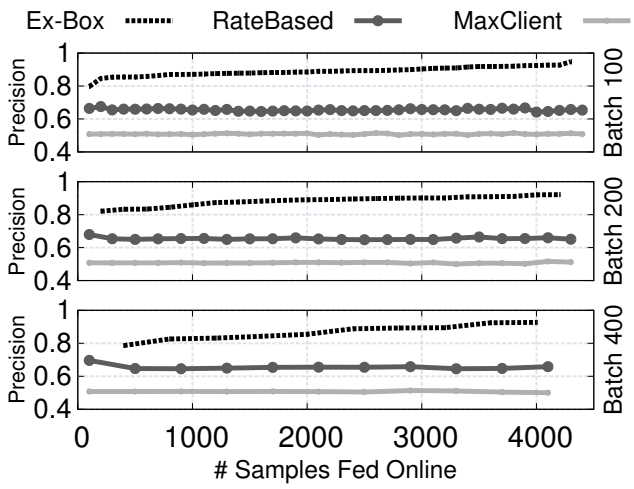


Figure 13: ExBox performance with diverse SNR, compared with baselines

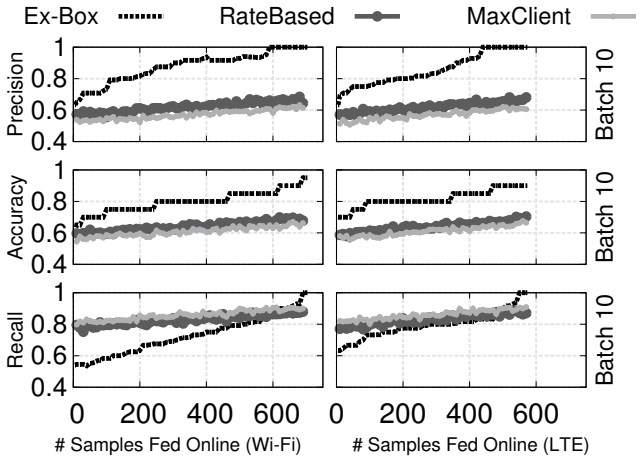


Figure 14: Precision, Accuracy and Recall for admission control in `ns-3` Wi-Fi and LTE simulations

To simulate populous networks we only consider traffic matrices with more than 20 simultaneous flows in WiFi. From these traffic matrices, we generate multiple random sets of 800 (X_m, Y_m) each for our analysis. Figure 14 shows the performance of ExBox with initial training using 10% (80) of the data points. We calculate the *precision*, *accuracy* and *recall* metrics on each batch of data. The model then learns from the flows admitted in that batch, as explained in the previous section. Similar to the testbed results, the performance of ExBox is sensitive to batch size of online updates. The left half of Figure 14 presents results on WiFi network. ExBox has a good precision rate ($\approx$ 0.9), compared to RateBased and MaxClient, indicating the model did a right

job if it admitted a flow. However recall being slightly lower indicates the model behaves conservatively. Note that, like the testbed, a small batch size of 10 is needed for this network as there is no SNR diversity.

The right half of Figure 14 shows the same analysis for LTE network with the LiveLab traces. Here, we do not restrict the total number of flows to be less than 8. Thus, we simulate all unique traffic matrices encountered in the LiveLab traces over LTE network and record 650 different (X_m, Y_m) tuples. With more than 100 samples fed online the classifier's precision reaches to ≈ 0.8 and it improves further on to ≈ 0.9 with 400+ samples fed online. Recall is ≈ 0.75. Similar to our testbed experiments, the Admittance Classifier performs better in LTE than in WiFi.

7. RELATED WORK

There is a huge body of prior work on admission control and network capacity modeling. In this section, we place our work with respect to the current state-of-the-art, which can be broadly categorized as follows:

Flow classification: ExBox assumes a priori knowledge of the application class to which a flow belongs. It thus relies on traffic flow classification techniques. Recent literature has already explored an array of such classification techniques with modest accuracy figures. Some of them handle real-time traffic classification using machine learning techniques [30, 27] while some others focus on classifying encrypted traffic [32, 49].

QoE as a performance metric: The notion of QoE has been around for a long time. Chen *et al*. [37] model Skype QoE in terms of rate, delay and jitter. Balachandran *et al*. leverage Machine Learning to capture cross-layer features that impact web QoE [28] and video QoE [29, 43]. QoE-Doctor [38] extends QoE modeling to mobile applications. However, they require modifications on the device to learn QoE accurately. Authors in [26] propose to model mobile application QoE from passive network measurements, but still require human training to build the model. In this work we leverage the IQX hypothesis [44] that provides a generic relationship between QoE and QoS. The strength of IQX lies in its applicability to different application classes. Our testbed experiments show that it can learn QoE-QoS relationship with minimal human involvement.

Capacity and admission control: In the past decade and half, a significant body of work has been invested in understanding capacity regions of various forms of wireless networks [65, 51, 45]. However, these studies focus on QoS as the predominant performance metric. With recent proliferation of the *app-based* ecosystem we argue QoE as a more suitable metric to measure network capacity. We are not alone in this argument. Song *et al*. [63] proposed a probabilistic admission control parameter for integrated cellular/WLAN networks by modeling QoE, in terms of delay and rate, in these networks. However, they require meticulous modeling of traffic arrival and contention procedure in each network. Similarly, Piamrat *et al*. [59] proposed QoE-aware admission control for simultaneous video calls in 802.11 networks. However, they only consider video calls in their traffic pattern, which is an over-simplification of the traffic mix found in today's networks. In contrast, ExBox aims to define *experiential capacity region* for diverse mix of application flows in dynamic wireless networks. We foresee ExBox to be easily deployed in operational networks.

Experience-oriented architecture: Recently, Jiang *et al*. [48] have proposed an experience-oriented network architecture wherein application providers and network providers collaborate to provide better user experience. Their work is complementary to ours as they try to diminish the administrative boundaries in order to guarantee better QoE to end users.

8. CONCLUSIONS

In this work, we motivate the need to define network capacity in terms of QoE, and not QoS. We define an *experiential capacity region*, which consists of flows of diverse applications and diverse channel quality, all with acceptable QoE. We propose ExBox, a middlebox which can learn the *experiential capacity* of a network, through in-network QoE estimation and machine learning. ExBox also adapts to changing network environments and continues to learn the experiential capacity. Our evaluation in a WiFi testbed and LTE testbed shows that ExBox correctly admits the flows with a precision rate of $\approx 0.8-0.9$ and recall rate of $\approx 0.6-0.8$. As part of future work, we plan to evaluate ExBox on a large scale testbed with multiple WiFi and LTE cells.

9. ACKNOWLEDGEMENTS

The authors would like to thank Abhinav Parate for helping in the initial problem formulation. and the paper's shepherd Patrick Thiran for guiding the camera-ready version. This work was partially supported by NSF award AST-1443951.

10. REFERENCES

[1] 3GPP Specification: Self Organizing Networks. http://www.3gpp.org/DynaReport/32500.htm.
[2] Admission control in Skype for Business Server, Microsoft. https://technet.microsoft.com/en-us/library/gg398529.aspx.
[3] Amazon instant video forum. http://amzn.to/2df0kdy.
[4] Android Debug Bridge (adb). developer.android.com/tools/help/adb.html.
[5] Android view client. https://github.com/dtmilano/AndroidViewClient/wiki.
[6] Aruba mobility controllers for next-gen networks. www.arubanetworks.com/products/networking/controllers/.
[7] Aruba: Working with QoS for voice and video. http://bit.ly/2eNeGCq.
[8] Benchmark 802.11ac Wave 2 performance. http://blog.mojonetworks.com/benchmark-802.11ac-wave-2-performance.
[9] Cisco admission control solutions. http://bit.ly/261Wx8y.

[10] Cisco network admission control. http://www.cisco.com/c/en/us/solutions/enterprise-networks/network-admission-control/index.html.

[11] Cisco SON architecture. www.cisco.com/c/en/us/solutions/service-provider/son-architecture/index.html.

[12] E2E Virtual Camera Software. http://www.e2esoft.cn/vcam/.

[13] IBM admission control solutions. http://ibm.co/1ZWXBU0.

[14] IP flow mobility and seamless wireless local area network (WLAN) offload. ww.3gpp.org/DynaReport/23261.htm.

[15] ip.access:lte. http://www.ipaccess.com/en/lte.

[16] LiveLab dataset from Rice University. http://livelab.recg.rice.edu/traces.html.

[17] Miercom wireless controller comparative performance. http://bit.ly/2eNdMWu.

[18] ns-3 simulator. www.nsnam.org.

[19] OpenEPC. http://www.openepc.com/.

[20] Qualcomm extends LTE to unlicensed spectrum to enhance mobile experiences. http://bit.ly/1XYoJ8i.

[21] Qualcomm WiFi SON. www.qualcomm.com/videos/qualcomm-wi-fi-son.

[22] Rukus WiFi calling. http://www.ruckussecurity.com/Wi-Fi-Calling.asp.

[23] The Evolved Packet Core - 3GPP. http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core.

[24] Universal tun/tap device driver. https://www.kernel.org/doc/Documentation/networking/tuntap.txt.

[25] YouTube Android Player API. https://developers.google.com/youtube/android/player/.

[26] AGGARWAL, V., HALEPOVIC, E., PANG, J., VENKATARAMAN, S., AND YAN, H. Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements. In *Proc. ACM MobiSys 2014*.

[27] ALSHAMMARI, R., AND ZINCIR-HEYWOOD, A. N. Machine learning based encrypted traffic classification: Identifying SSH and Skype. *CISDA 9* (2009).

[28] BALACHANDRAN, A., AGGARWAL, V., HALEPOVIC, E., PANG, J., SESHAN, S., VENKATARAMAN, S., AND YAN, H. Modeling web quality-of-experience on cellular networks. In *Proc. ACM Mobicom 2014*.

[29] BALACHANDRAN, A., SEKAR, V., AKELLA, A., SESHAN, S., STOICA, I., AND ZHANG, H. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM Computer Communication Review* (2013), vol. 43, ACM, pp. 339–350.

[30] BAR-YANAI, R., LANGBERG, M., PELEG, D., AND RODITTY, L. Realtime classification for encrypted traffic. In *International Symposium on Experimental Algorithms* (2010), Springer, pp. 373–385.

[31] BARI, F., AND LEUNG, V. Automated network selection in a heterogeneous wireless network environment. *Network, IEEE 21*, 1 (2007), 34–40.

[32] BERNAILLE, L., TEIXEIRA, R., AKODKENOU, I., SOULE, A., AND SALAMATIAN, K. Traffic classification on the fly. *SIGCOMM Computer Communication Review 36*, 2 (2006), 23–26.

[33] BERNAILLE, L., TEIXEIRA, R., AND SALAMATIAN, K. Early application identification. In *Proc. ACM CoNEXT 2006*.

[34] BISWAS, S., BICKET, J., WONG, E., MUSALOIU-E, R., BHARTIA, A., AND AGUAYO, D. Large-scale measurements of wireless network behavior. In *Proc. ACM SIGCOMM 2015*.

[35] CHANDRA, R., AND BAHL, P. Multinet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *Proc. IEEE INFOCOM 2004*.

[36] CHAPELLE, O. Training a support vector machine in the primal. *Neural computation 19*, 5 (2007), 1155–1178.

[37] CHEN, K.-T., HUANG, C.-Y., HUANG, P., AND LEI, C.-L. Quantifying Skype user satisfaction. In *Proc. ACM SIGCOMM 2006*.

[38] CHEN, Q. A., LUO, H., ROSEN, S., MAO, Z. M., IYER, K., HUI, J., SONTINENI, K., AND LAU, K. QoE doctor: Diagnosing mobile app QoE with automated UI control and cross-layer analysis. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), IMC '14, ACM, pp. 151–164.

[39] CHEN, Y., FARLEY, T., AND YE, N. QoS requirements of network applications on the Internet. *Inf. Knowl. Syst. Manag. 4*, 1 (Jan. 2004), 55–76.

[40] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[41] DAI, S., TONGAONKAR, A., WANG, X., NUCCI, A., AND SONG, D. Networkprofiler: Towards automatic fingerprinting of android apps. In *Proc. IEEE INFOCOM 2013*.

[42] DAINOTTI, A., PESCAPÉ, A., AND SANSONE, C. Early classification of network traffic through multi-classification. In *Proceedings of the Third International Conference on Traffic Monitoring and Analysis* (2011), TMA'11, pp. 122–135.

[43] DOBRIAN, F., SEKAR, V., AWAN, A., STOICA, I., JOSEPH, D., GANJAM, A., ZHAN, J., AND ZHANG, H. Understanding the impact of video quality on user engagement. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 362–373.

[44] FIEDLER, M., HOSSFELD, T., AND TRAN-GIA, P. A generic quantitative relationship between quality of experience and quality of service. *Network, IEEE 24*, 2 (March 2010), 36–41.

[45] GASTPAR, M., AND VETTERLI, M. On the capacity of wireless networks: The relay case. In *Proc. IEEE INFOCOM 2002*.

[46] HIGGINS, B. D., REDA, A., ALPEROVICH, T., FLINN, J., GIULI, T. J., NOBLE, B., AND WATSON, D. Intentional networking: Opportunistic exploitation of mobile network diversity. In *Proc. ACM Mobicom 2010*.

[47] HUANG, N.-F., JAI, G.-Y., CHAO, H.-C., TZANG, Y.-J., AND CHANG, H.-Y. Application traffic classification at the early stage by characterizing application rounds. *Inf. Sci. 232* (May 2013), 130–142.

[48] JIANG, J., LIU, X., SEKAR, V., STOICA, I., AND ZHANG, H. Eona: Experience-oriented network architecture. In *Proc. ACM HotNets 2014*.

[49] KARAGIANNIS, T., PAPAGIANNAKI, K., AND FALOUTSOS, M. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM*

Computer Communication Review (2005), vol. 35, ACM, pp. 229–240.

[50] KIM, K. H., AND SHIN, K. G. Prism: Improving the performance of inverse-multiplexed TCP in wireless networks. *IEEE Transactions on Mobile Computing 6*, 12 (Dec 2007), 1297–1312.

[51] KODIALAM, M., AND NANDAGOPAL, T. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proc. ACM Mobicom 2005*.

[52] LASKOV, P., GEHL, C., KRÜGER, S., AND MÜLLER, K.-R. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research 7*, Sep (2006), 1909–1936.

[53] LEONG, C. W., ZHUANG, W., CHENG, Y., AND WANG, L. Call admission control for integrated on/off voice and best-effort data services in mobile cellular communications. *Communications, IEEE Transactions on 52*, 5 (2004), 778–790.

[54] LI, W., AND MOORE, A. W. A machine learning approach for efficient traffic classification. In *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (Oct 2007), pp. 310–317.

[55] MAHINDRA, R., VISWANATHAN, H., SUNDARESAN, K., ARSLAN, M. Y., AND RANGARAJAN, S. A practical traffic management system for integrated LTE-WiFi networks. In *Proc. ACM Mobicom 2014*.

[56] NAGHSHINEH, M., AND ACAMPORA, A. S. QOS provisioning in micro-cellular networks supporting multimedia traffic. In *Proc. IEEE INFOCOM 1995*.

[57] PEFKIANAKIS, I., LUNDGREN, H., SOULE, A., CHANDRASHEKAR, J., LE GUYADEC, P., DIOT, C., MAY, M., VAN DOORSELAER, K., AND VAN OOST, K. Characterizing home wireless performance: The gateway view. In *Proc. IEEE INFOCOM 2015*.

[58] PENG, L., YANG, B., CHEN, Y., AND CHEN, Z. Effectiveness of statistical features for early stage internet traffic identification. *Int. J. Parallel Program. 44*, 1 (Feb. 2016), 181–197.

[59] PIAMRAT, K., KSENTINI, A., VIHO, C., AND BONNIN, J. QoE-Aware Admission Control for Multimedia Applications in IEEE 802.11 Wireless Networks. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th* (Sept 2008), pp. 1–5.

[60] QIAN, F., WANG, Z., GERBER, A., MAO, Z., SEN, S., AND SPATSCHECK, O. Profiling resource usage for mobile applications: a cross-layer approach. In *Proc. ACM MobiSys 2011*.

[61] SHAFIQ, M. Z., ERMAN, J., JI, L., LIU, A. X., PANG, J., AND WANG, J. Understanding the impact of network dynamics on mobile video user engagement. In *ACM SIGMETRICS Performance Evaluation Review* (2014), vol. 42, ACM, pp. 367–379.

[62] SHIN, S., AND SCHULZRINNE, H. Experimental measurement of the capacity for voip traffic in IEEE 802.11 wlans. In *Proc. INFOCOM 2007*.

[63] SONG, W., CHENG, Y., AND ZHUANG, W. Improving voice and data services in cellular/wlan integrated networks by admission control. *Wireless Communications, IEEE Transactions on 6*, 11 (2007), 4025–4037.

[64] TAX, D. M., AND LASKOV, P. Online SVM learning: from classification to data description and back. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on* (2003), IEEE, pp. 499–508.

[65] TOUMPIS, S., AND GOLDSMITH, A. J. Capacity regions for wireless ad hoc networks. *Wireless Communications, IEEE Transactions on 2*, 4 (2003), 736–748.

[66] VELDHUIZEN, T. Measures of image quality. *CVonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision* (2010).

[67] YANG, B., HOU, G., RUAN, L., XUE, Y., AND LI, J. Smiler: Towards practical online traffic classification. In *Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on* (Oct 2011), pp. 178–188.

[68] ZARINNI, F., CHAKRABORTY, A., SEKAR, V., DAS, S. R., AND GILL, P. A first look at performance in mobile virtual network operators. In *Proc. ACM IMC 2014*.

[69] ZHANG, J., CHEN, X., XIANG, Y., ZHOU, W., AND WU, J. Robust network traffic classification. *IEEE/ACM Transactions on Networking 23*, 4 (Aug 2015), 1257–1270.