MoViDiff: Enabling Service Differentiation for Mobile Video Apps

Satadal Sengupta*, Vinay Kumar Yadav†, Yash Saraf‡, Harshit Gupta*,
Niloy Ganguly*, Sandip Chakraborty*, Pradipta De§
*Dept. of CSE, IIT Kharagpur, Kharagpur 721302, India; †Dept. of CSE, IIT Patna, Patna 801103, India
‡Dept. of CSE, IIT BHU, Varanasi 221005, India; §Dept. of CS, Georgia Southern University, GA 30458 USA
Email: satadal.sengupta@iitkgp.ac.in, {vk7055,yash29saraf}@gmail.com,
{niloy,sandipc}@cse.iitkgp.ac.in, pde@georgiasouthern.edu

***Abstract*—Among the mobile applications contributing to the surging Internet traffic, video applications are some of the biggest contributors. Most of these video applications use HTTP/HTTPS tunneling making it difficult to apply port based or packet data based identification of flows. This makes it challenging for network operators to enforce bandwidth regulation policies for app based service differentiation due to lack of flow identification mechanisms for mobile apps. We explore a packet data agnostic feature of video flows, namely packet-size, to identify the flows. We show that it is possible to train a classifier that can distinguish packets from streaming and interactive video apps with high accuracy. We design and implement a system, called *MoViDiff*, with this classifier at the core, that allows bandwidth regulation between video traffic of two different categories, streaming and interactive. We show that we can achieve an average accuracy of 96% in classifying the traffic, with the maximum accuracy reaching as high as 98%.**

I. Introduction

Video applications on different mobile devices, including wearables, are steadily adding to the volume of Internet traffic. To overcome limited connectivity of the devices, often the traffic is routed through a mobile hub, like a smartphone [1]. When all the video traffic is funneled through a single device, identifying the source of the traffic can be challenging. Consider a scenario as shown in Fig. 1, where our smartphone acts as a personal mobile hub that interconnects multiple devices, such as smart-watches, fitness monitoring devices (e.g., Fitbit), smart switches for door control and so on. All of these devices are connected to the Internet through the mobile hub via wireless tethering. In this setting, while watching a streaming video from YouTube on the smartphone, one can chat with a friend via Skype on the smartwatch [2]; this is prevalent in many scenarios, for example, in remote medical help, remote tech support, a long-distance couple trying to emulate the experience of watching a movie together [3], etc. [1]

Unfortunately, in such scenarios, the limited wireless bandwidth of the smartphone to the Internet now must be shared by multiple bandwidth hungry video applications. This can hurt the Quality of Service (QoS) for each application. Assume that the user can pre-configure preference between YouTube and Skype. Then, would it be possible to ensure QoS for at least the preferred application, if bandwidth requirement for both cannot be supported? Unfortunately, with current techniques, it is difficult for the network to distinguish among traffic from different applications originating from a single device since typically the traffic is tunneled over HTTP/HTTPS. In addition, often the data traffic is encrypted making it impossible to use traditional techniques to identify traffic sources.

[1]Applications such as Rabbit [4] have also been developed, which combine streaming and video-chatting experience.

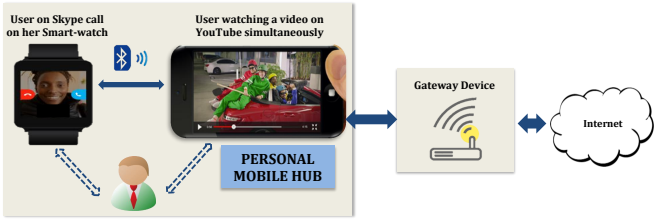
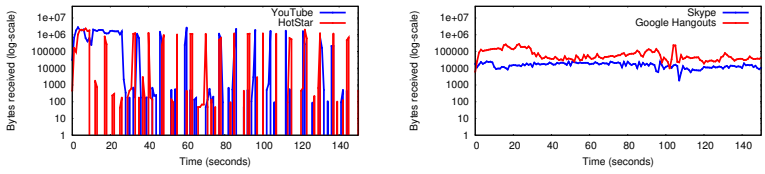


Fig. 1: Use Case: Multiple mobile devices connected to the network through a Personal Mobile Hub. The video applications running simultaneously on the mobile devices must share the bandwidth.

Traffic identification relies on identifying unique characteristics of the application traffic which can be used as signatures. Common signatures used to identify desktop applications are the port number used by an application, host address of the server, or any other identifier in the network packets [5]. This assumes unencrypted traffic that allows deep packet inspection. Since majority of mobile app traffic is tunneled over HTTP [6], [7], simple port based classification does not work. For most mobile video applications, where the data packets are encrypted and sent over HTTPS, packet inspection based approaches will fail.

Although inspection based approaches may fail, other side-channel information can be exploited to identify the traffic categories. For example, streaming and interactive traffic, which are our major concerns in this paper as discussed with the use case illustrated in Fig. 1, will have different traffic patterns due to inherent differences in the characteristics of streaming video and interactive video traffic. Assuming realistically that a user cannot engage in multiple video apps of same category simultaneously (like watching videos over two YouTube instances), it is possible to identify the source type



(a) Streaming Video (b) Interactive Video

Fig. 2: Traffic distribution in bytes/sec for two types of mobile video applications showing the difference in traffic characteristics.

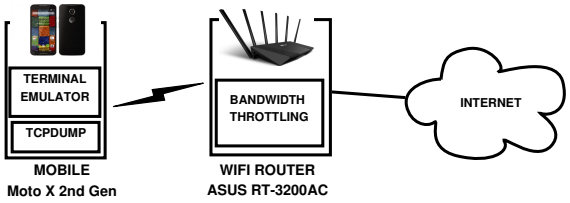


Fig. 3: In-Laboratory Testbed Setup for Data Collection

when the video traffic categories are different. In a preliminary study, we analyzed the temporal characteristics of traffic from streaming video and interactive video sources running on mobile devices. As shown in Fig. 2, video apps within the same category show similar characteristics, while differing significantly from apps of another category. Such properties can be used to distinguish between data packets originating from video apps of different categories.

In this work, we investigate a simple packet-level feature of video traffic that can be used to identify the sources (§ IV) – *packet-size*. Packet-size varies significantly between streaming and interactive videos. The feature is used to train a classifier to identify packets from pre-defined video sources, in presence of mixed traffic flows. We design and implement a system called *MoViDiff*, which can identify packets from different video sources, and apply pre-configured bandwidth control policies (§ V). The core of *MoViDiff* is the classification engine which runs at the gateway connecting the end devices to the ISP's network. We demonstrate the efficacy of the system using the scenario, as shown in Fig. 1 (§ VI). The classification results exhibit an accuracy of at least 92%, while reaching up to 98% in some cases, thus propelling *MoViDiff* to work successfully as a bandwidth regulator.

The key contributions of this work are, (1) We show that, using packet data agnostic properties of video traffic, it is possible to identify sources of video traffic on a mobile device from mixed flows. (2) We implement a system, called MoViDiff, that demonstrates the efficacy of the packet data agnostic features in video traffic differentiation. MoViDiff can be used by network operators, and enterprise IT managers for resource optimization and policy enforcement.

II. Related Work

Classifying traffic from mobile applications has spurred active research recently due to its practical necessity, as well as, the challenges due to lack of features suitable for traffic classification. If the traffic is unencrypted, deep packet inspection (DPI) techniques, surveyed in [8], can help in classification. For example, port number can help in identifying application groups, like browsing, email, maps [6]. Similarly, by peering into the HTTP header, one can read the HTTP *User-agent* field that shows the application name. Xu et al. showed how to use this field to identify app sources [9]. Since setting this field is not mandatory for app developers,

TABLE I: Traffic Trace Collection Scenarios

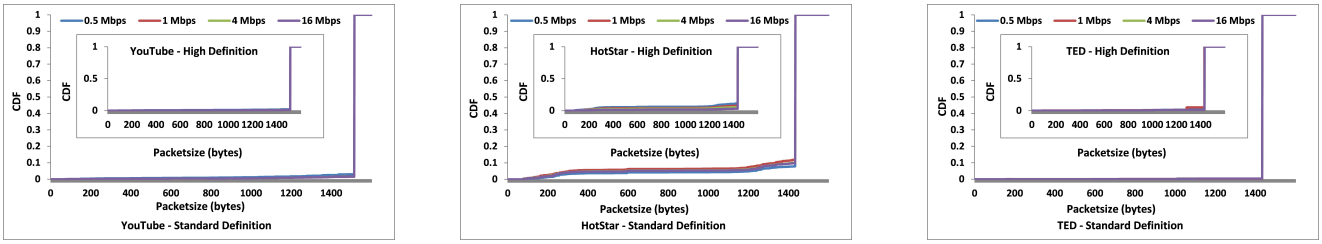
App Category	Application/s	Total Trace Volume (MB)	Resolution	Trace Duration Range (mins)	Bandwidth (Mbps)
Streaming Video	YouTube	304	360p, 720p	5-40	0.5
	HotStar	645	SD, HD		1.0
	TED	188	HQ-off, on		4.0
Interactive Video	Skype	442	Default	5-60	16.0
	Google Hangouts	230	Default		
	ooVoo	356	Default		

the technique works well for popular apps. Other indirect information, like in-app advertisement sources, can also be used to reveal the flow identity, as shown by [10]. Statistical methods can be used to identify patterns in app traffic. Dai et al. showed that by executing different apps in an emulator and gathering network traces from these executions, it is possible to build signatures for identifying specific traffic sources [11]. Yao et al. reduced the overhead of building signatures in a recent work where they used context of signatures to improve identification accuracy and scalability [12]. Automatic generation of classifiers for traffic identification has been presented in [13], [14]. AppPrint also uses extensive traffic observations to build signatures based on header fields [15]. Since most of these works rely on information that must be read from the packets, they are ill suited for classifying encrypted traffic.

Video traffic tends to exhibit more strongly identifiable flow patterns as compared to any generic traffic class. SkyTracer is a tool that shows the patterns in Skype traffic [16]. Jesudasan et al. have identified generic features across different versions of Skype that can be used as features for statistical identification of Skype flows [17]. Similarly, there have been characterization of YouTube traffic for PC versions [18] and YouTube mobile application [19]. Recently, Shi et al. also showed that video source can be identified for encrypted traffic just by analyzing only few timing features of the packet trace [20]. Our work builds on these works by carefully analyzing groups of mobile video applications, and finding statistical properties that can enable classification of flows from multiple video sources simultaneously.

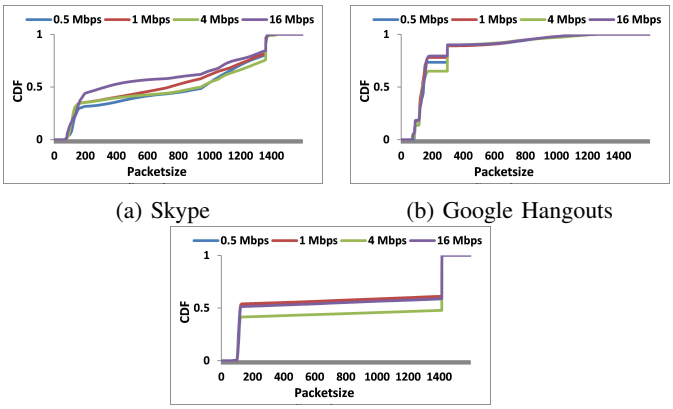
III. Experimental In-Laboratory Setup

In order to study different characteristics of mobile video traffic, we must collect traffic traces under controlled operating conditions. Important control parameters include background



(a) YouTube (b) HotStar (c) TED

Fig. 4: Packet-size Distribution for Streaming Video Apps



(a) Skype (b) Google Hangouts

(c) ooVoo

Fig. 5: Packet-size Distribution for Interactive Video Apps

traffic, channel conditions, and traffic from advertisements embedded in the videos. Background traffic could originate from services, such as, Dropbox sync, Email sync, etc., which requires to be weeded out. The variation in channel conditions may lead to video streaming adapting to a different bit-rate and different resolution. Similarly, advertisements within the actual video content may distort content-agnostic pattern analysis. Therefore, we have implemented an in-laboratory testbed to enable controlled collection of the packet traces.

A. Testbed Setup

The tested setup is shown in Fig. 3. The end device connects to the ISP's network through a wireless router. The wireless router is instrumented such that we can control the bandwidth between the router and the end device. We used a Motorola Moto X 2^{nd} generation Android smartphone as the end device where we execute each mobile video app separately. This device is rooted to allow trace collection using `tcpdump`. Android `Terminal Emulator` app provides the interface to execute `tcpdump`. The router is an ASUS RT-3200AC router with IEEE 802.11ac Wi-Fi standard. This router provides the `Adaptive QoS` feature, which allows manual configuration of the `Download Bandwidth` field to specify a controlled download rate.

B. Data Collection Scenarios

We identified six popular mobile video applications on Google Play – three from each video class (buffered and interactive) – as shown in Table I. We capture download traffic for these apps with various combinations of video lengths, video resolutions, and network traffic bandwidth (by throttling at the router). We also log associated events, such as start and end time of embedded advertisements, and of re-buffering instances. The packet traces collected for different scenarios with these apps are analyzed in the following section.

IV. Traffic Pattern Analysis of Video App Traffic

In this section, we analyze the traffic traces collected in our testbed with respect to packet-size – a packet data agnostic feature. Our objective is to understand whether the feature is suitable to uniquely identify streaming video and interactive video.

Preprocessing – Flow Segregation: In order to reliably consider only downstream (server to client) data for our analysis, and to weed out network control packets, we divide the traffic into individual flows, and consider packets belonging to downstream data flows only. We differentiate the flows based on 5-tuple information `<source IP, source port, destination IP, destination port, transport protocol>`, present in the header of each packet. It may be noted that such 5-tuple based flow differentiation, most of the times, may not lead to app source identification, because the video apps change their content server frequently for load balancing and other policy related reasons, whereas Skype like interactive apps use a super-peer architecture where the peer gets changed at almost every session.

A. Packet-size Distribution

We consider the transport layer (TCP or UDP) packet-size for our analysis. We show the packet-size distributions (in Fig. 4a (YouTube), Fig. 4b (HotStar) and Fig. 4c (TED)) for streaming video apps, and (in Fig. 5a (Skype), Fig. 5b (Google Hangouts), Fig. 5c (ooVoo)) for interactive video apps. As the streaming video apps are available in standard definition ($360p$) and high definition ($720p$), we show the distribution for both. From the figures, we make the following observations: (1) For the streaming video apps, packet-size saturates at a fixed value, which is the maximum transmission

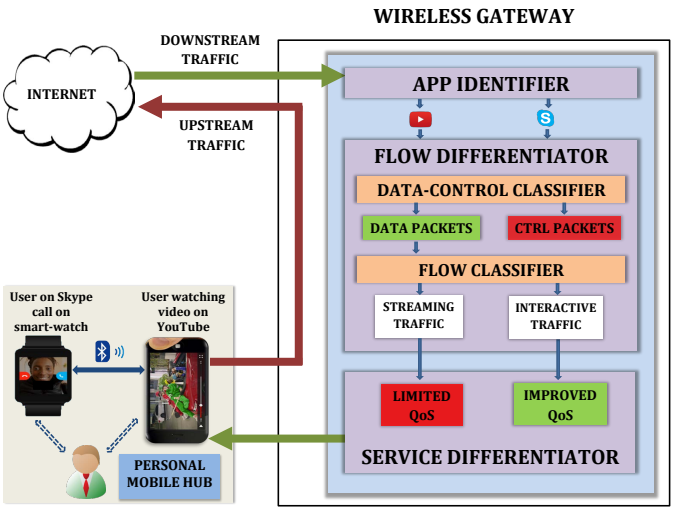


Fig. 6: System Architecture of *MoViDiff* shows an end device connected to the wireless gateway. The gateway, which can be a wireless router or any computer with a wireless NIC, implements various software components enabling service differentiation between streaming video and interactive video applications running on the end device.

unit (MTU) for the transport layer packets at Wi-Fi. The packet-size distribution shows a sharp transition indicating that most of the packets are of large size (more than 1400 bytes). However, for the interactive video apps, most of the packets are of lower sizes. (2) The packet-size distribution shows hardly any variation with video quality (for streaming videos) or network bandwidth. (3) Even though packet-size distributions for individual apps are consistent, there is a marked difference between the distributions of streaming video apps and interactive video apps. Packet-size can, therefore, be exploited to build an efficient traffic classification system.

Difference with respect to the aforementioned feature helps us to understand and differentiate between the flows from an interactive video app and a streaming video app. The detailed system design for this classification and service differentiation methodology for mobile video apps is discussed in the next section.

V. MoViDiff: Design and Implementation

We present the system, called *MoViDiff*, which utilizes the insights from traffic classification to implement a end host specific service differentiation platform for mobile video applications. The overall system architecture is shown in Fig. 6. A smartphone, acting as a mobile hub, for personal wearable devices, communicates with the wireless gateway connected to the Internet. Functionalities specific to service differentiation are implemented at the wireless gateway.

The system needs to detect launch of a video app at an end device to trigger data agnostic techniques to regulate the traffic of specific apps that may be tunneled over HTTPS or encrypted. In order to identify the launch of an app, there are two choices. One can implement a client module running on the end host that notifies the gateway that an app has been launched. This requires instrumenting the end device, as well as, any smartphone app. Alternatively, it is possible to detect app launch by monitoring the packet exchanges during application connection initiation. Handshake packets are unencrypted, and hence packet inspection technique works well on such packets. Subsequently, we apply data agnostic techniques on encrypted packets, where packet inspection cannot be used.

We implement the wireless gateway on a *HP ProBook* laptop equipped with *Intel Core i5*, 2.2 GHz processor, 8 GB of RAM, and a 802.11ac network interface. It runs *Ubuntu 16.04 LTS* OS along with `RTL8723BE` network driver. We configure the laptop to act as a Wi-Fi hotspot using the `HotSpot` feature provided by the OS. The end device(s) connects to the laptop acting as the wireless gateway. We use the Linux `iptables` module, and `Netfilterqueue` (NFQUEUE) for implementing *MoViDiff*. *NFQUEUE* is an *iptables* and *ip6tables* target which delegates the decision (accept, reject, or mark) on packets to a user-space software. We use the `netfilterqueue` [21] Python module to implement the user-space software. Packets reaching the laptop are first intercepted by the following `iptables FORWARD` rule:

```
iptables -I FORWARD -d <ip_range> -j
NFQUEUE --queue-num <queue_num>
```

All packets matched by the rule are enqueued in the target `NFQUEUE` queue (specified by *queue_num* in the rule). We implement the user-space software by writing our own Python module, which we call *pymovidiff*.

Next, we describe the key software components of *MoViDiff*, which are implemented as part of *pymovidiff*.

A. App Identifier

App Identifier is required to identify an application that is newly launched. We use a deep packet inspection (DPI) based approach to analyze the handshake packets that are exchanged when a network-based app is launched from an end device. An app typically establishes a secure connection with the server by using a cipher data exchange protocol over SSL. Since these packets are unencrypted, packet inspection is possible on these packets. The SSL server certificate signature is unique for an app, although the actual server can be geographically load balanced. We use DPI to read the SSL server certificate signature, and accordingly identify the app. This is an one-time low overhead process to check whether the app belongs to one of our intended video apps.

The *App Identifier* module runs in the background, and performs a lazy read on the header of each packet to ascertain if the protocol indicates a certificate exchange. In case it finds one that does, it springs into action, and inspects the packet deeper to identify the origin app. It then communicates this information to the *Flow Differentiator* module, which triggers traffic differentiation accordingly. The *App Identifier* goes back to its lazy mode thereafter.

B. Flow Differentiator

After identifying that a specific app has been launched, the task is to identify the data packets that belong to that app. There are two tasks that are performed, in order. The *Data-Control Classifier* segregates the control packets of a flow from its data packets. The *Flow Classifier* then matches the data packets to a specific application – one of the currently running applications, as communicated by the *App Identifier*. We use a supervised support vector machine (SVM) for classification in each of these cases, the implementation details of which are provided in § VI.

1) Data-Control Classifier: In this study, we refer to network control packets, such as ARP, DNS, IGMP, ICMP, NTP, DB-LSP-DISC (Dropbox Lan Sync Protocol), etc., as *control packets*. Control packets do not carry any video content, but appear in large numbers in the flow. By isolating the control packets from *data packets* (packets which carry video data in their payloads), we can design the flow signature for an app more accurately. The data-control *binary* classifier uses a SVM with packet-size as the lone feature. This is based on our observation that data packets typically tend to be larger in size as compared to control packets; although certain types of control packets have comparable sizes with data packets, those tend to have fixed sizes, and are therefore easily identifiable.

2) Flow Classifier: The *Flow Classifier* is the data packet classifier that accepts data packets as input, and based on a feature, classifies each packet into originating from one of the considered traffic classes (streaming video or interactive video traffic). This module is implemented as a binary SVM, where two *competing* (for bandwidth) video apps (one streaming and one interactive) are active. The feature, i.e., packet-size, identified based on *traffic analysis* (§ IV), is used for classification. Since we assume that the flows can be encrypted, we cannot look inside the packet headers. However, `NFQUEUE` provides an API which can extract the transport layer packet-size. We use this API in the packet-size detection module inside *pymovidiff*.

C. Service Differentiator

The *Service Differentiator* module regulates bandwidth offered to individual apps by scheduling release of data packets. A producer thread in *pymovidiff* passes each packet to the classification engine. The classifier inserts the packet into either the streaming video or the interactive video user-level queues. A consumer thread dequeues a packet, and forwards to the outgoing link, based upon a user-defined configuration parameter λ. λ denotes the priority ratio that the user wants to associate with a specific application; e.g., if Skype and YouTube are being run simultaneously, Skype is serviced with λ times more priority than YouTube.

VI. Evaluation

We evaluate *MoViDiff* in two phases: first, we evaluate the performance of the system as a service differentiator over an experimental testbed; then, we evaluate it as a traffic differentiator.

A. MoViDiff Performance as a Service Differentiator

In order to evaluate *MoViDiff* as a service differentiation system, we use two different Moto X Android phones connected to the laptop acting as a wireless gateway; one plays a YouTube video, while the other plays Skype chat. *MoViDiff* performs bandwidth differentiation between Skype and YouTube traffic originating from the phones.

Service Differentiation Performance: We let the two apps run without intervention for the first 50 seconds. At the 50^{th} second, we enable *MoViDiff*, specifying higher priority for Skype and lower priority for YouTube. For the next 50 seconds, the apps run with bandwidths as provisioned by *MoViDiff*. We repeat the experiment for $\lambda = 10$ and $\lambda = 30$, where λ is the configuration parameter as defined in § V. The results are shown in Fig. 7a and Fig. 7b.

We observe that after the 50^{th} second, i.e. when *MoViDiff* has been enabled, the average bandwidth for YouTube drops from 106658 bps to 56594 bps (47% drop) when λ=10, while it improves from 2726 bps to 3765 bps (38% rise) for Skype. This phenomenon is amplified when λ=30; average bandwidth for YouTube drops from 113917 bps to 27408 bps (76% drop), while for Skype it goes up from 2759 bps to 4161 bps (51% rise). This shows *MoViDiff's* capability of prioritizing traffic originating from different apps, according to user-specified priority.

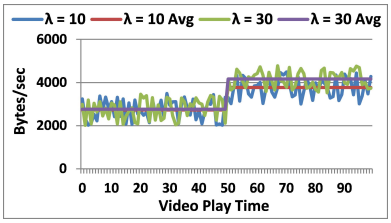
Impact on Video Quality: Bandwidth differentiation for video apps is expected to impact video quality of the streaming app. In order to understand the nature of this impact, we record the same 1-minute segment of a YouTube video playing on a smartphone under 3 conditions: (i) without *MoViDiff*, (ii) with *MoViDiff* when λ=10, and (iii) with *MoViDiff* when λ=30. We compare the PSNR (Peak Signal-to-Noise Ratio) values (using OpenCV [22]) for case (ii) vs. (i) and case (iii) vs (i), respectively. The results are shown in Fig. 7c.

Note that higher PSNR values indicate higher video quality. We observe that PSNR values are consistently higher (average of 32.8) when λ=10 than when λ=30 (average of 21.9). Thus, quality degradation is little in the former case, and more pronounced in the latter. This shows that the user can control her quality of experience by setting an appropriate value of λ.

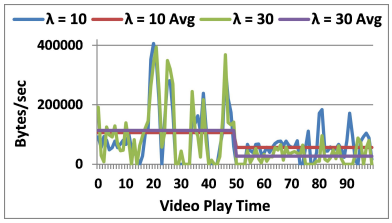
The analysis of PSNR provides an interesting insight on *MoViDiff*. Because of the adaptive streaming strategy adopted by most streaming video apps (e.g., YouTube, HotStar, TED), a limited drop in available bandwidth does not affect their quality significantly, whereas the additional bandwidth provided to the interactive apps can boost up their performance, as those are mostly based on real-time transport protocols. The user can exercise her choice (by tuning the parameter λ) regarding how much she is willing to compromise with the quality of streaming video apps, in order to improve the quality of interactive video apps.

B. MoViDiff Performance as a Traffic Classifier

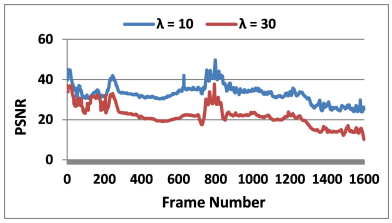
In this subsection, we present the performance of the individual classifiers used in our system. First we evaluate the data-control classifier, followed by the flow classifier. The



(a) Bandwidth Change for Skype



(b) Bandwidth Change for YouTube



(c) PSNR Variation for YouTube

Fig. 7: Effect of Service Differentiation due to *MoViDiff*. The bandwidth allocation of preferred application Skype increases while for YouTube it decreases. Despite bandwidth allocation, the YouTube stream is acceptable as shown by the PSNR.

TABLE II: Data-Control Classifier Performance

App Class	Classes	Average Precision	Average Recall	Average F1-Score	Average Accuracy
Streaming Video Apps	Data Control	1.00 1.00	1.00 1.00	1.00 1.00	1.00
Interactive Video Apps	Data Control	1.00 1.00	1.00 1.00	0.99 0.99	0.99

TABLE III: Classification Performance of *MoViDiff*

Classes	Precision	Recall	F1-Score	Accuracy
YouTube Skype	1.00 0.89	0.89 0.96	0.94 0.92	0.92
YouTube GoogleHangouts	1.00 0.99	0.99 0.98	0.99 0.98	0.98
YouTube ooVoo	0.99 0.99	0.99 0.97	0.99 0.98	0.97
HotStar Skype	0.91 0.98	0.98 0.86	0.94 0.91	0.92
HotStar GoogleHangouts	1.00 0.99	0.98 0.97	0.99 0.98	0.97
HotStar ooVoo	0.99 1.00	0.99 0.97	0.99 0.98	0.97
TED Skype	1.00 0.95	0.95 0.95	0.97 0.95	0.95
TED GoogleHangouts	1.00 1.00	1.00 0.98	1.00 0.99	0.98
TED ooVoo	0.99 1.00	0.99 0.97	0.99 0.98	0.98

SVM module used is as implemented in *scikit-learn* [23], with $C = 1.0$ and *radial basis function* (RBF) kernel. In every case, 70% of available data is used for training, and the remaining 30% for testing. Equal number of samples have been considered for each class to rule out class imbalance.

1) Performance of the Data-Control Classifier: In order to evaluate performance of the data-control classifier, we train and test a SVM with packet-size as the only feature. The results are summarized in Table II. The classifier performs remarkably well for both cases – streaming video and interactive video traffic. The minimum classification accuracy is 99%, which reaches up to 100%. This can be attributed to the fact that control packets are typically of lesser size than data packets, irrespective of the source app.

2) Classification between a Buffered and Interactive Video App: We investigate the classification performance of the flow classifier for pairs of streaming and interactive apps. In all cases, we train a SVM with packet-size as the only feature (as done in [20]). The results are presented in Table III. It is observed that the classifier works extremely well for each of the pairs, with classification accuracy values in the range between 0.92 and 0.98. This is made possible by the distinct sizes of data packets generated by streaming video apps and interactive video apps. The figures and analysis presented in sub-section IV-A support this observation.

VII. Conclusion

Ability to identify the application source of each traffic flow going through a gateway has many applications. However, for mobile applications, such classification is challenging since apps use HTTP/HTTPS for encapsulating traffic. Typical Deep packet inspection (DPI) based methods fail to recognize the streams. In this work, we focused on mobile video applications, and showed that characteristics of the flows originating from these apps differ with respect to packet-size, which is a packet data agnostic feature. We used this feature to train classifiers that can identify flows belonging to streaming video and interactive video. The classifier forms the core of a service differentiation platform, called *MoViDiff*, that runs on a wireless gateway, and apportions bandwidth among different video applications from an end device as per user preference. We show that *MoViDiff* is able to achieve an average classification accuracy of 96%, with the maximum accuracy reaching as high as 98%.

VIII. Future Directions

In this paper, we limit our study to the possibility and efficacy of classification between streaming and interactive video apps, in the absence of any kind of background traffic. Such background traffic may originate from browsing, email, data sync apps such as Dropbox, etc. In the future, we shall expand our study to include classification between various video sources even in the presence of noisy mixed background traffic. We realize that in such cases, packet-size may not suffice as a feature, and we may need to explore other packet data agnostic features, such as patterns in inter-arrival time, etc. Such directions would serve as interesting future work.

References

[1] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e Silva, "The mobile hub concept: Enabling applications for the internet of mobile things," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 2015, pp. 123–128.

[2] "You can now skype from your android wear smartwatch." [Online]. Available: https://techcrunch.com/2015/09/29/you-can-now-skype-from-your-android-wear-smartwatch/

[3] "Reddit (longdistance): Movie and skype - how do you do it?" [Online]. Available: https://www.reddit.com/r/LongDistance/comments/2mfrqy/movie_and_skype_how_do_you_do_it/

[4] "Rabbit." [Online]. Available: https://www.rabb.it/

[5] A. Callado, C. Kamienski, G. Szabó, B. P. Gerö, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 3, pp. 37–52, 2009.

[6] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 281–287.

[7] A. Gember, A. Anand, and A. Akella, "A comparative study of handheld and non-handheld traffic in campus wi-fi networks," in *Passive and Active Measurement*. Springer, 2011, pp. 173–183.

[8] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 2, pp. 1135–1156, 2014.

[9] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 329–344.

[10] A. Tongaonkar, S. Dai, A. Nucci, and D. Song, "Understanding mobile app usage patterns using in-app advertisements," in *Passive and Active Measurement*, 2013.

[11] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "Networkprofiler: Towards automatic fingerprinting of android apps," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 809–817.

[12] H. Yao, G. Ranjan, A. Tongaonkar, Y. Liao, and Z. M. Mao, "SAMPLES: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15, 2015, pp. 439–451.

[13] Y. Choi, J. Y. Chung, B. Park, and J. W.-K. Hong, "Automated classifier generation for application-level mobile traffic identification," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 1075–1081.

[14] Q. Xu, Y. Liao, S. Miskovic, M. Baldi, Z. M. Mao, A. Nucci, and T. Andrews, "Automatic generation of mobile app signatures from traffic observations," in *Proceedings of IEEE INFOCOM 2015*, ser. INFOCOM '15, 2015.

[15] S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, "AppPrint: Automatic fingerprinting of mobile applications in network traffic," in *Proceedings of Passive and Active Measurement Conference*, 2015, pp. 57–69.

[16] Z. Yuan, C. Du, X. Chen, D. Wang, and Y. Xue, "Skytracer: Towards fine-grained identification for skype traffic via sequence signatures," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 1–5.

[17] R. N. Jesudasan, P. Branch, and J. But, "Generic attributes for skype identification using machine learning," *Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. A*, vol. 100820, p. 20, 2010.

[18] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of youtube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012.

[19] J. J. Ramos-Munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. M. López-Soler, "Characteristics of mobile youtube traffic," *Wireless Communications, IEEE*, vol. 21, no. 1, pp. 18–25, 2014.

[20] Y. Shi and S. Biswas, "Protocol-independent identification of encrypted video traffic sources using traffic analysis," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[21] "The netfilter.org libnetfilter queue project." [Online]. Available: http://www.netfilter.org/projects/libnetfilter_queue/index.html

[22] "Video input with opencv and similarity measurement." [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/highgui/video-input-psnr-ssim/video-input-psnr-ssim.html

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.