Human-centric Computing and Information Sciences

RESEARCH Open Access

Efficient policing for screen mirroring traffic

Orestis-Stavros Loizides[1], Marios Kastrinakis[1], Ghada Badawy[2], Mohammed N. Smadi[3], David Murray[4] and Polychronis Koutsakis[4*]

*Correspondence: p.koutsakis@murdoch.edu.au
[4] School of Engineering and Information Technology, Murdoch University, Science and Computing Building 245, SC1.012, 90 South Street, Murdoch, WA 6150, Australia
Full list of author information is available at the end of the article

Abstract

The greediness of multimedia applications in terms of their bandwidth demands calls for new and efficient network traffic control mechanisms, especially in wireless networks where the bandwidth is limited. In an enterprise-like environment, an additional burden is expected to be added to the network by screen mirroring traffic. Smart mobile devices are displacing personal computers in many daily applications but at the same time users still need to use a large display, keyboard and mouse. Hence, the transmission of low-latency, high fidelity video over a Wi-Fi link can lead to significant unfairness among users in terms of the bandwidth that is available to them, if this wireless video traffic is not accurately policed. In this work, we focus on the problem of policing screen mirroring traffic. We evaluate various classic and new traffic policing mechanisms, and we propose a new mechanism which is shown to clearly outperform all other mechanisms, including the widely used token bucket policer.

Background

Smart mobile devices such as smartphones and tablets are becoming more powerful every day with the advancement of mobile computing chips, while at the same time major software and operating system companies develop their products in a single code base suitable for many platforms [1]. In addition, a 2013 survey [2] placed smartphone popularity at around 85%, surpassing all other kinds of computing devices. The above facts seem to point towards a future where smart mobile devices might replace computers completely, in personal and corporate environments.

On the other hand, users still want to use a large display, keyboard and mouse for content creation and when they are not on the move. This use case has been addressed using tether technology such as the mobile high-definition link (MHL) [3], however to maintain the grab-and-go experience there is a trend of replacing the need for cables with a wireless link that connects the mobile device to a wireless docking station. In such a setup, the video on the mobile device display is mirrored on a larger display. Low-latency connection is required to ensure the interactivity of the end-user is maintained, but the actual latency limits are application dependent. Its widespread adoption and high bandwidth make Wi-Fi a key candidate for carrying this docking traffic. Miracast [4], which was recently ratified as a standard to allow mobile device display mirroring over Wi-Fi,

Springer Open

is also a candidate for enabling the video component of any docking station. Miracast is widely adopted across a variety of devices, with more than 7000 devices certified, including most leading smartphone, tablet and smart TV brands [5]. Screen mirroring is already an integral part of many video mediated collaborations [6] and used to give presentations through smart phones [7].

At the heart of a Miracast source, an H.264 encoder streams over a Wi-Fi peer-to-peer (P2P) mode without needing access to an overlaid Wi-Fi network. However, in enterprise environments where an overlaid Wi-Fi network is not only present but heavily used, contention due to medium access between the Wi-Fi P2P docking link and the overlaid Wi-Fi network will have a direct impact on the quality of experience (QoE) for the docking video. The impact and potential QoE degradation will depend, among many other factors, on the network's resiliency [8], as well as on the characteristics of the video traffic, which in turn is dependent on the video content being compressed, the encoder implementation and the target latency profile.

Hence, careful network design is necessary to ensure efficient use of the network's limited resources [9]. Given that video processing requires tremendous resources in terms of computation and transmission of the encoded video [10] it is an uphill battle to deploy innovative services on existing wireless networks [11]. For this reason, traffic characterization and modeling of multimedia services are very important for achieving efficient network operation. The generated models can be used as traffic rate predictors during the network operation phase (online traffic modeling). Traffic models are very useful for traffic management algorithms and congestion control schemes, which prevent the network from possible overload [12]. Dynamic Resource Allocation (DRA) schemes are especially important for live streams, where the video stream characteristics are not known in advance. In order to accurately estimate the required network resources for a certain flow the chosen DRA scheme must be able to predict the required bandwidth for future video frames. To adjust the bandwidth assignment for a certain video stream, DRA renegotiates the assigned bandwidth for that flow. The main goals for a DRA scheme are: to predict the longest possible period with the least prediction error, and to provide the best possible resource utilization with the lowest achievable frame delay [13]. This will ensure that the QoE of accepted video sessions is not degraded [14].

Hence, in our recent work in [15] we developed H.264 video traffic models for low-latency, candidate Miracast source implementation for content that resembles a typical desktop user in an enterprise-like environment. We proposed an accurate model based on the combination of clustering with Markov chains and the use of the Jaccard index similarity coefficient.

In order to provide the required quality of service (QoS) and QoE guarantees, network resources need to be reserved according to both the QoS/QoE requirements and the specified traffic parameters of each application. The role of source policing mechanisms is critical towards this goal. The source policing mechanism protects network resources against intentional or unintentional traffic overflow from certain sources. Several policing mechanisms have been proposed for network control in the literature. Traffic that is considered by the mechanisms to be exceeding a user's contract is either dropped immediately or marked as non-conforming, in order to be dropped if needed at any network node, depending on the total traffic load (this approach is relevant both to an edge router and to software-defined networking [16]). Four of the mechanisms which have been most extensively studied, all of them static in nature, are: the token bucket and its variations [17–19]; the jumping window [20, 21]; the moving window (also known as the sliding window) and the exponentially weighted moving average (EWMA) [20].

The token bucket mechanism has been widely studied, is currently used in Cisco equipment [22] and has been widely integrated into home routers [23]. The reason for its popularity is its ability to verify easily whether a source conforms to its declared (at call setup) traffic parameters. It allows a certain amount of burstiness (which is necessary for video traffic) while imposing a limit on the average source transmission rate [24].

The upper bound on the source's burst length is equal to the token bucket size and this bound can be described by the formula

$$\mathrm{A(s,t)} \leq \sigma + \rho(\mathrm{t} - \mathrm{s}), \quad \mathrm{s} < \mathrm{t} \tag{1}$$

where A(s, t) denotes the amount of traffic leaving the bucket between times s and t, σ is the maximum burst size and ρ is the token generation rate.

The jumping window (JW) mechanism uses windows of a fixed length T side by side through time. A new window starts immediately after the conclusion of the previous one. During a window, only K bytes (or packets) can be submitted by the source to the network. In the case that a source attempts to transmit more than K bytes, the excessive traffic is dropped (or marked as nonconforming, as in the case of the token bucket). The mechanism is implemented with the use of a token counter, similar to the one of the token bucket, and in each new window the associated packet counter is restarted with an initial value of zero [20].

The moving window (sliding window) mechanism is similar to the jumping window, but more stringent and more complex to implement. This mechanism again ensures that the maximum number of bytes transmitted by a source within any given time interval of duration equal to the fixed window size, *T*, is upper bounded by K bytes. The difference with the Jumping Window mechanism is that each video frame size is remembered for the width of exactly one window, starting with the specific video frame and ending T frames later. This mechanism can be interpreted as a window which is steadily moving along the time axis, with the requirement that the frame sizes of T frames are stored for the duration of one window [20].

The EWMA mechanism uses consecutive-time windows like the JW mechanism. The difference is that the maximum number of accepted packets in a time window is a function of the allowed mean value of the video trace per interval and an exponentially weighted sum of the number of accepted packets in the preceding intervals.

In [25], it has been shown that dynamic traffic policing based on accurate H.263 videoconference traffic modeling can clearly outperform the classic static mechanisms, in terms of the percentage of marked packets of conforming users (i.e., users that adhere to their declared traffic parameters). The reason is that the static mechanisms are unable to cope with the burstiness of video traffic, and hence cause the marking of a significant percentage of the transmitted packets. However, accurate prediction is not possible for all types of video sequences, and even when it is, it often involves a higher degree of complexity which would incur additional computational requirements for the system.

In our work in [26] we proposed a new traffic policing mechanism, the Frame Size Aware Token Bucket (FSA-TB), which was shown to outperform all of the widely used mechanisms in the literature. However, FSA-TB is based on the standard group of pictures (GoP) structure of a video and for this reason, as we will explain in "The Jaccard index-infused Markovian clustering policer (JIMC-P)" of the paper, it cannot be used for policing video encoded in a similar way to a Miracast hardware encoder. The use of classic traffic policing mechanisms, on the other hand, can lead to significant and unnecessary packet dropping, which especially for screen mirroring traffic would be unacceptable. Users transmitting from their smart device to a screen, in a shared environment, would expect the highest QoS and QoE.

For this reason, in this work we propose a new traffic policing mechanism for screen mirroring traffic, which is based on the accurate model proposed in [15]. We use two different datasets for our experiments and we compare the performance of our proposed mechanism against nine other policing mechanisms for both datasets. To the best of our knowledge, this is the first time in the relevant literature that traffic policing on screen mirroring traffic is studied.

The paper is structured as follows. "Video encoding and data collection methodology" presents the relevant details for video encoding and for the way we collected our data. In "Traffic policing mechanisms" we present existing traffic policing mechanisms, against which we will compare our proposed mechanism. "The Jaccard index-infused markovian clustering policer (JIMC-P)" presents our new proposed mechanism for screen mirroring traffic. In "Results and discussion" we discuss our results. "Conclusions" includes our conclusions and ideas for future work.

Video encoding and data collection methodology

In our work in [15] we worked with two different datasets, encoded with the H.264 video coding standard. H.264 or MPEG-4 Part 10, AVC is a video coding standard developed by ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG). It is the most widely accepted video coding standard (since MPEG-2) and it covers a wide area of video applications ranging from mobile services and videoconferencing to IPTV, DTV and HD video storage [27]. According to the H.264 standard, an encoded video trace features two distinct characteristics: (1) Every video frame comes from one of three different types of frames, and (2) video frames are organized in groups with a specific structure.

There are three different types of frames, I-Frames (Intra-coded Frames), P-Frames (Predicted Frames) and B-Frames (Bi-directional predicted Frames). P-Frames are smaller than I-Frames and B-Frames are the smallest [28]. Video frames are grouped together in GOP structures that specify the order in which intra- and inter-frames are arranged. A GOP pattern specifies the amount and order of P and B-Frames between two successive I-Frames. Every GOP contains a single I-Frame with which it starts. The GOP pattern is defined by the distance X between I-Frames and the distance Y between P-Frames or between the I-Frame and the succeeding P-Frame. In general, in the H.264 standard the amount of B-Frames is greater than the amount of I or P-Frames inside a GOP structure.

Our work is based on real user-generated data from a large shared cube space resembling an enterprise environment. Each participant in the data collection ran trace collection scripts for about a month. One script polled the operating system every 33.3 ms to record the name of the main application that the participant was working on. Another script recorded the participant's screen at 30 fps using encoding parameters that resembled a Miracast hardware encoder as closely as possible. The actual video was not recorded, but only the statistics of the encoded video were collected (i.e., I and P frame sizes). The scripts started automatically each work day at 8 am and stopped at 7 pm. When a participant locked his/her screen the scripts would report that the user is idle for that duration and video traces collection would stop till the user unlocks his/her machine. All of the users were using Windows 7 machines.

Some details on the network architecture follow. A smartphone will either host applications natively (e.g. Windows Mobile Phone) or will act as a thin client to a backend computing platform (e.g. Citrix Thin Client running on Android Phone). In both instances the actual applications will be rendered by the smartphone GPU before being re-encoded and sent over Wi-Fi to a Miracast dongle connected to a large display. Applications will appear on the phone and on the large display exactly as if they were being rendered onto a desktop PC with one application in the foreground and multiple applications in the background (i.e. either docked or in the background of the display, so users were allowed to use multiple applications at a time). The Windows 7 setup was merely used to capture the application that an end-user has in the foreground and to capture and feed the raw video data (i.e. the output of the GPU showing foreground and background applications side by side) into a carefully configured video encoder (i.e. FFmpeg, as explained below). This encoder is configured to "mimic" the configuration of an H.264 Miracast source configuration running a mobile phone. The encoder configuration parameters have been set using datasheets of Miracast sources.

A recording framework was deployed on every host machine. It was running and logging in the background during the recording period. The FFmpeg [29] program was used for video traffic recording. It logged the compressed H.264 video information (i.e., frames sizes, GOP structure, frames' time of arrival etc.) of the host's machine desktop. The frame resolution was the same as the PC's screen resolution, i.e., it is not a constant; we used different resolutions depending on the different PC screens that were being used. The default windows resolution is 1920 × 1080, so this was often the case for our videos. The frame rate was 30 fps. Although FFmpeg was running constantly, it was capable of logging video traffic information only if the host's machine GPU was active (i.e., the host machine was not in hibernation, sleep or monitor energy saving mode). A Windows PowerShell [30] script was used for logging and timestamping the active foreground applications. Windows PowerShell was programmed to log the application's name every 33.3 ms (in order to keep up with FFmpeg logs, where we had one frame every 33.3 ms). We should also note that Windows PowerShell is capable of reporting the application's name only if the host machine is unlocked and the user is not logged off. The average PSNR of our videos was 65 dB.

The main difference between the two datasets we collected lies in the different encoding of video traces. The first dataset (Dataset 1) has been encoded with the High 4:2:2 Profile of the H.264 standard, which is typical for professional applications. This profile can generate I, P and B frames. However, in our datasets the—*tune zero latency* command was used in FFmpeg to prohibit the encoder from producing B-Frames, in order to minimize latency. For this dataset, we have a GOP structure of 60 frames in length, where every GOP starts with an I-Frame and the following 59 frames are of type P. The second dataset (Dataset 2) has been encoded with different encoding parameters. Those parameters try to resemble a Miracast hardware encoder as closely as possible. Since I-Frames' sizes are much larger than P-Frames, Miracast encoders do not use I-Frames but use periodic intra refresh [31] instead. This enables each frame (in our case each I-Frame) to be capped to the same size by using a column of intra blocks that move across the video trace from one side to the other, thereby "refreshing" the image. In effect, instead of a big keyframe (in our case an I-Frame), the keyframe is spread over many frames (in our case P-Frames). For this dataset, we do not have a GOP structure. We only have P-Frames with an exception of one I-Frame whenever the host computer starts or its user logs on.

Our recording framework ran on different periods of time, between March and May 2015 for the first dataset and between June and July 2015 for the second dataset. We replaced every user's name with a different letter from the alphabet for reasons of anonymity. In Table 1 we present some general statistics of our two datasets, such as general information about our records, as well as total, minimum, average and maximum sizes of our video traffic frames over all applications. It is worth mentioning, that the average P-Frame size in Dataset 2 is larger by a factor of ≈ 14 in comparison with the P-Frames in Dataset 1, as shown in Table 1. Our collected data, for both datasets, can be downloaded from [32].

Traffic policing mechanisms

In [26] we first evaluated five static traffic policing mechanisms from the literature. We then proposed a new paradigm in video traffic policing, which focuses on the activity of the source to dynamically tailor the traffic policing accordingly. We compared the five static mechanisms against five new activity-based traffic policing mechanisms we proposed. Of those ten mechanisms, six (two static and four activity-based ones) were shown in [26] to provide the best traffic policing results. These were: (a) the token bucket (TB) mechanism, (b) the jumping window (JW) mechanism, (c) the variable exponentially weighted moving average (V-EWMA) mechanism, (d) the hybrid V-EWMA token

Table 1 Dataset 1 and dataset 2 encoding and statistics over all applications

Encoding and statistics	Dataset 1	Dataset 2
Encoding	High 4:2:2: Profile of the H.264 standard	Miracast hardware encoder
GOP structure	60 frames (1 I, 59 P)	No
# of recordings days	24	22
# of applications	29	26
# of video traffic records	14,932,183	20,892,611
Total size of video traffic (Gbytes)	120	424
Min video traffic size (bytes)	159	190
Average video traffic size (bytes)	8032	20,298
Max video traffic size (bytes)	598,613	422,435

bucket (VEWMA-TB), (e) the frame size aware token bucket (FSA-TB), (f) the GOP modeling based jumping window (GMB-JW).

A seventh efficient mechanism, the recurrent leaky bucket (RLB) was recently proposed in the literature, in [33]. This mechanism, extends the leaky bucket algorithm which imposes a hard limit on the source transmission rate [34] by including a new input parameter, the cycle duration. This is set approximately equal to the burst cycle of the traffic, which for variable bit rate videos is the same as the period of the GoP. The goal of the RLB mechanism is to allow users to choose a lower sustainable rate for the light traffic periods and obtain extra credit at the end of a cycle called *renewal*.

We will compare our proposed mechanism against nine other traffic policing mechanisms, i.e., the seven mechanisms mentioned above plus two more versions of the Token Bucket mechanism. In these two additional versions we combine the Token Bucket, which is the most widely used traffic policer, with two video traffic models that have been shown to achieve high accuracy for various types of video traffic. The idea is that allowing the policer to generate tokens simply based on the mean rate of the source may be insufficient for bursty multimedia sources. Therefore, we combine the functionality of the token bucket with video traffic models which are better tailored to capture the burstiness of a video user's traffic. The two additional traffic policing mechanisms are the TB-GBAR, where tokens are generated based on the gamma-beta auto-regression (GBAR) model [35] and the TB-SARIMA, where tokens are generated based on the seasonal ARIMA model [13].

In "The Jaccard index-infused Markovian clustering policer (JIMC-P)" we will explain why a new mechanism is needed for screen mirroring traffic.

The Jaccard Index-Infused Markovian Clustering Policer (JIMC-P)

As it will be shown in our results in "Results and discussion", our previously proposed mechanism, FSA-TB [26] confirms, for the first dataset of screen mirroring traffic, its excellence in comparison with all other mechanisms from the literature. However, for the second dataset it is impossible to be implemented as FSA-TB is based on the assumption of a standard GoP structure for the video. FSA-TB used the size of the I-frame of a GoP in order to tailor accordingly the token generation rate for the P-frames of the same GoP, whereas in the second screen mirroring traffic dataset only P-frames exist. For the same reason, the VEWMA-TB mechanism cannot be implemented on the second dataset.

Hence, we propose in this work a new traffic policing mechanism, which is completely different from FSA-TB. This new mechanism, JIMC-P, is based on the Jaccard Index-Infused Markovian Clustering model presented in our recent work on screen mirroring traffic modeling in [15], i.e., it is a token bucket mechanism in which tokens are generated based on the JIMC model instead of being generated based on a mean rate. It should be emphasized that the high accuracy of the model (3–4% prediction error on average for I-frames and P-frames) does not guarantee an equally good performance when the model is incorporated into the policer. For example, if the error is steadily caused by higher predictions, the policer will be too lenient against misbehaving users, whereas if the error is steadily caused by lower predictions, the policer will be too strict, leading to the unnecessary possible packet dropping. Hence, a thorough examination of the potential use of the model within the policer is important.

We briefly present the model below.

We view the video trace sequence as a vector containing all the I-frames' sizes (we explain below the different procedure for P-frames). We place all the vector's elements as points on the 1-D plane and we then use the K-Means clustering algorithm in order to cluster similar-sized frames. The distance metric that we used for clustering is the cityblock distance, which calculates the sum of absolute differences (i.e., the L_1 distance). Even though K-Means is a powerful clustering algorithm, it has a significant drawback. The *K* amount of clusters has to be selected heuristically. In our case, we concluded after several experiments that the optimal number of clusters per tested application (i.e., the number of clusters that leads to the highest modeling accuracy) ranged between 4 and 11, depending on the application.

Next, we constructed a Markov chain in which each cluster corresponds to one state of the chain. We computed the Transition Probability Matrix $T = [P_{i,j}]^2$ for the Markov chain, which contains $K \times K$ elements, via Eq. (2).

$$P_{i,j} = \frac{\#\text{ of jumps from state } i \text{ to state } j}{\#\text{ of jumps from state } i} \tag{2}$$

Finally, we found the best distribution fit for the data in each cluster.

To model P-frames' sizes we used another approach, i.e., moving from the 1-D to the 3-D plane. To do so, we employed, for the first time in the relevant video traffic modeling literature, the concept of the Jaccard Index. The Jaccard Index [36], also known as the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of two sample sets. The Jaccard coefficient measures similarity between finite sample sets and is defined in general as the size of the intersection divided by the size of the union of two sample sets, as depicted in Eq. (3):

$$J(A,\ B) = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

where A and B denote the two sample sets.

In our case, we wanted to find for every P-Frame in a GOP (for Dataset 1) or in a Window (for Dataset 2, where the size of a Window is equal to the size of a GOP), the two "closest" P-Frames. We wanted to use an approach different from a simple autocorrelation calculation (as it was shown to perform poorly when used in other video traffic models). This approach is the use of the Jaccard Index in the following way.

For every P-Frame, denoted as X, in a GOP or a Window, we calculate its Jaccard Index with every other P-Frame, denoted as Y, in the same GOP or Window. The sample sets A and B in this Jaccard Index calculation are the "neighboring frames" of X and the "neighboring frames" of Y, respectively. As a "neighboring frame" P^* of a P-Frame P, we define every P-Frame that satisfies the following two rules:

1. The absolute difference between the sizes of P and P* does not exceed the standard deviation of P-Frames' sizes.
2. The arrival of P* does not change the autocorrelation (lag-1) of P-Frames in the trace, more than 10% compared to the change that occurred from the arrival of P.

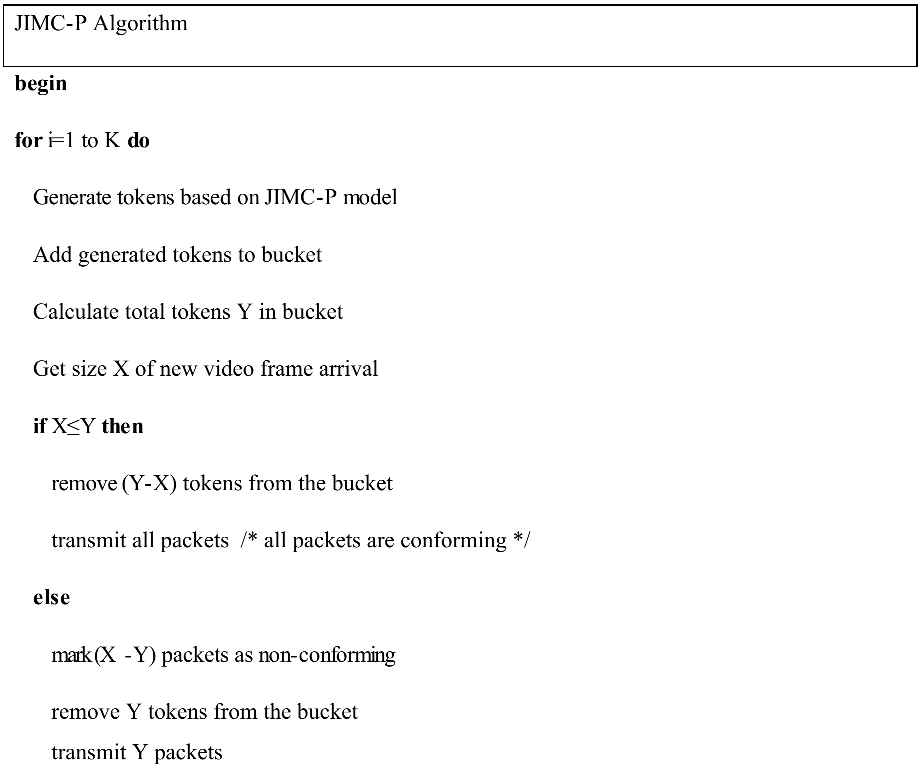
Via this definition, we found that the two "closest neighbors" of each P-Frame are the previous and the following one, for all major applications of both datasets.

We then view the video trace sequence as a vector, $\langle R_p(t), R_c(t), R_n(t) \rangle$, t = 2, 3, 4, … Here $R_c(t)$ denotes the frame size of the tth P-Frame, $R_p(t)$ denotes the frame size of the P-Frame before $R_c(t)$ (i.e., $R_p(t) = R_c(t-1)$) and $R_n(t)$ denotes the frame size of the P-Frame after $R_c(t)$ (i.e., $R_n(t) = R_c(t+1)$). We place all the $\langle R_p(t), R_c(t), R_n(t) \rangle$ pairs as points on the 3-D plane, where $R_p(t)$, $R_c(t)$ and $R_n(t)$ is viewed as the x-coordinate, y-coordinate and z-coordinate of the corresponding point, respectively. Hence, each P-Frame is clustered by taking into account not only its own size but also the size of its adjacent frames.

Finally, as with the I-frames, we find the best distribution fit for the data in each cluster. This concludes the brief presentation of the model. The interested reader is referred to [15] for more information.

Our proposed traffic policing mechanism, JIMC-P, follows the logic of the token bucket, however it does not generate tokens based on the declared token generation rate of the source, but based on the estimate of the model for the size of the upcoming video frame. In this way we hope to capture the fluctuations in the sources' traffic needs in a much more accurate way than the classic token bucket policer does. Hence, JIMC-P is similar in nature with the two other policing mechanisms that are evaluated here for the first time, TB-GBAR and TB-SARIMA. The difference is in the underlying traffic model which is incorporated into the token bucket mechanism for each of the three mechanisms.

The basic algorithm for JIMC-P, for a video of K frames, is shown below.



```
JIMC-P Algorithm
begin
for i=1 to K do
  Generate tokens based on JIMC-P model
  Add generated tokens to bucket
  Calculate total tokens Y in bucket
  Get size X of new video frame arrival
  if X≤Y then
    remove (Y-X) tokens from the bucket
    transmit all packets  /* all packets are conforming */
  else
    mark(X -Y) packets as non-conforming
    remove Y tokens from the bucket
    transmit Y packets
```

Results and discussion

In this work, we consider the case of an edge router (router at the network edge) with a high level of intelligence so that all traffic flows can be efficiently classified and treated according to network policies. Screen mirroring traffic can be classified and treated in a manner similar to the one described in [37], i.e., packets are classified according to an identifier which is left on the data packets by an application (e.g. on the packet header) and is used for identifying the generating application, then marking the class of the data packets and correspondingly policing the data packets according to the classification marker, thus reducing the traffic rate to force the traffic to follow the allocation limit.

Our simulations were conducted in Matlab. We focus only on the actual traffic being transmitted by the users and whether it is conforming or not, i.e., we do not consider the case of "lagging" users (users who have experienced a noisy channel and therefore have involuntarily been transmitting at a decreased rate).

We assume that each user is transmitting only one type of traffic (i.e., the video chosen for the simulation) which arrives at the edge router and we derive our results by implementing the policing mechanisms for the whole duration of the trace.

Figure 1 presents the average marked traffic results over all videos of both datasets, in the case that the users are conforming, i.e., that they adhere to their declared mean and peak traffic parameters. In this case an ideal traffic policer would mark zero percent of the user's traffic. This is of course an unattainable standard, as the burstiness of the video traffic combined with the short windows within which a policer needs to make a decision does not allow for a perfect policer. FSA-TB was shown to outperform all mechanisms for every video of Dataset 1 used in our study, followed by JIMC-P. However, as explained in "The Jaccard index-infused Markovian clustering policer (JIMC-P)", FSA-TB cannot be applied to Dataset 2. For Dataset 2 our new mechanism, JIMC-P clearly outperforms all other mechanisms from the literature, as it marks by far the lowest percentage of traffic for conforming users.

An even more important result is presented in Table 2. As much as we want to prevent conforming users from suffering unnecessary QoE degradation, due to possible packet dropping, the top priority for every policing mechanism remains the handling of non-conforming users. An efficient traffic policer is expected to mark for possible dropping all excess traffic from non-conforming users. Table 2 presents the results for all ten policers used in our study, when a different video is transmitted from the one that is actually

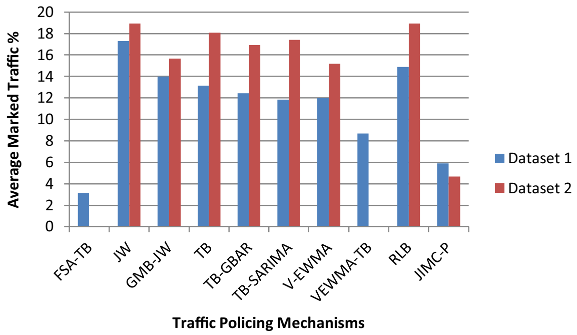


Fig. 1 Average marked traffic over all videos for both datasets, for conforming users

Table 2 Marked traffic for non-conforming users, for dataset 1

Mechanism	Marked traffic (%) for actual transmitted video	Marked traffic (%) for declared video			Average marked traffic (%)
		Video 1	Video 2	Video 3	
FSA-TB	5.01	26.91	28.91	64.00	39.94
JW	20.95	31.1	30.46	63.06	41.54
GMB-JW	14.96	24.96	20.3	25.16	23.48
TB	15.93	28.17	27.73	62.95	39.62
TB-GBAR	15.01	26.45	25.77	59.40	37.21
TB-SARIMA	14.22	27.08	28.42	61.13	38.88
V-EWMA	14.61	27.22	26.38	62.87	38.82
VEWMA-TB	14.96	20.54	17.39	21.76	19.9
RLB	17.99	29.38	28.96	62.98	40.44
JIMC-P	3.22	79.91	43.31	46.13	56.45

declared. The three declared ones, videos 1–3, were videos from our dataset with smaller mean, standard deviation and peak bandwidth requirements than the transmitted one. In this case, where a user either intentionally or unintentionally attempts to overload the network, the JIMC-P policer clearly outperforms, on average, all policers as it marks the highest percentage of non-conforming traffic. Simultaneously, if implemented for the actual transmitted video (therefore, for a conforming user) JIMC-P succeeds in marking the lowest percentage of traffic among all policers.

The results presented in Table 3 and Fig. 2 are derived by running a similar experiment for Dataset 2, with the difference that the declared movies, instead of being screen mirroring traffic videos, were videos from movies. More specifically, we chose three H.265 movies from [28], all of which again had smaller mean, standard deviation and peak bandwidth requirements than a specific screen mirroring video from our dataset. The mean rate of the movies was 4–17 times smaller than our video, the standard deviation was 2–9 times smaller and the peak was 1.5–6 times smaller. We again declared each of these three movies as the one that was going to be transmitted, whereas we actually attempted to transmit the screen mirroring video. Once again, the JIMC-P policer marks the highest percentage of traffic as non-conforming among all policers (as shown from the individual results for the different declared movies, in Table 3, and from the average results over all movies in Fig. 2). JIMC-P also achieves the lowest marked traffic

Table 3 Marked traffic for non-conforming users, for dataset 2

Mechanism	Marked traffic (%) for actual transmitted video	Marked traffic (%) for declared video		
		Lake house	Big buck bunny	Harry Potter
JW	23.23	77.16	88.60	94.05
GMB-JW	20.98	31.98	64.94	94.23
TB	19.57	77.02	88.59	94.05
TB-GBAR	17.81	75.24	87.03	93.74
TB-SARIMA	18.38	73.67	84.28	93.46
V-EWMA	17.99	77.12	88.58	94.07
RLB	20.61	77.03	88.59	94.05
JIMC-P	8.53	83.01	95.71	97.47

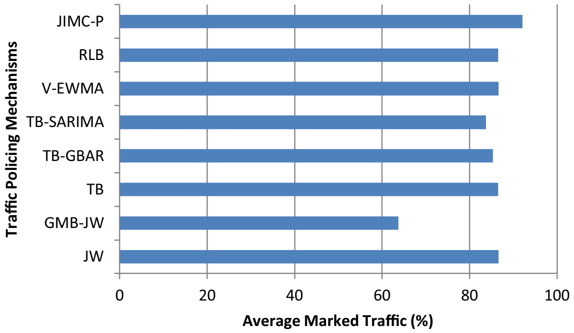


Fig. 2 Average marked traffic for non-conforming users, for dataset 2

percentage among all policers when the true screen mirroring video is declared and transmitted.

A secondary interesting result is that the TB-GBAR and TB-SARIMA mechanisms outperform the classic token bucket in terms of the averaged marked traffic for conforming users (i.e., they mark a smaller percentage of traffic). However, the classic token bucket outperforms them both when implemented for non-conforming users. This result indicates that the use of the models improves the user-tailored experience for conforming users, but the effort of the models to capture user behavior allows more unwanted traffic into the network when the user is non-conforming.

Conclusions

In this work we have proposed, for the first time in the relevant literature to the best of our knowledge, a new traffic policing mechanism for bursty screen mirroring video traffic.

Our proposed mechanism, the Jaccard Index-Infused Markovian Clustering Policer, is shown to outperform nine other mechanisms (seven from the literature and two additional ones studied in this work for the first time) for the two screen mirroring datasets used in our study. The most important result is that JIMC-P not only provides the highest percentage of marked traffic for non-conforming users, but also provides the smallest percentage of marked traffic for conforming users, hence achieving an excellent balance between strictness and fairness.

In future work we intend to collect an even larger amount of data and to evaluate JIMC-P for other types of video traffic (H.264 and H.265) and to evaluate the scheme theoretically as well. Still, we believe that its practical usefulness is already clear by its comparison with the token bucket, which is the most widely used traffic policer. We also intend to extend our work to cover mixed traffic environments where users may transmit multiple types of traffic. Our goal will be to propose new versions of JIMC-P and FSA-TB which will take into account the specific QoS and QoE of each user in a framework comprising call admission control, multiple access control and traffic policing.

Authors' contributions

OSL implemented eight of the ten traffic policing mechanisms used in this study, produced and analyzed the respective results. MK, GB and MS collected and analyzed the data, worked on the JIMC model and on the model's implementation into the JIMC-P traffic policing mechanism. DM worked on the analysis of all the results and on the structure of the paper. PK designed and proposed the JIMC model and its implementation into the JIMC-P traffic policing mechanism. PK also implemented two of the ten traffic policing mechanisms used in this study and analyzed the results of all ten policing mechanisms. All authors read and approved the final manuscript.

Author details
[1] School of ECE, Technical University of Crete, Chania, Greece. [2] Computing Infrastructure Research Center, McMaster University, Hamilton, Canada. [3] ECE Department, McMaster University, Hamilton, Canada. [4] School of Engineering and Information Technology, Murdoch University, Science and Computing Building 245, SC1.012, 90 South Street, Murdoch, WA 6150, Australia.

Competing interests
The authors declared that they have no competing interests.

Availability of data and materials
As explained in the paper, our datasets are available at: http://users.isc.tuc.gr/~makastrinakis/docs/Screen_Mirroring_Video_Trace_Datasets.zip.

Ethics approval and consent to participate
Not applicable.

Funding
This was not a funded research project.

Publisher’s Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 3 December 2017 Accepted: 14 February 2018
Published online: 27 February 2018

References

1. Gilbert D (2015) Microsoft wants to replace your PC with your smartphone, International Business Times. http://www.ibtimes.co.uk/microsoft-wants-replace-your-pc-your-smartphone-1499042. Accessed 15 Jan 2018
2. Naked Security by Sophos (2013) INFOGRAPHIC: users weighed down by multiple gadgets–survey reveals the most carried devices. https://nakedsecurity.sophos.com/2013/03/14/devices-wozniak-infographic/. Accessed 15 Jan 2018
3. http://www.mhltech.org/. Accessed 15 Jan 2018
4. Wi-Fi Alliance (2012) Discover Wi-Fi: Wi-Fi CERTIFIED Miracast™. http://www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast. Accessed 15 Jan 2018
5. Wi-Fi Alliance (2017) Wi-Fi CERTIFIED Miracast™ delivers high-resolution multimedia experience. https://www.wi-fi.org/news-events/newsroom/wi-fi-certified-miracast-delivers-high-resolution-multimedia-experience. Accessed 15 Jan 2018
6. Sorensen H et al (2015) Wireless smart phone mirroring in video calls. Paper presented at the 15th IFIP TC.13 international conference on human-computer interaction, Bamberg, Germany
7. Iqbal F (2016) http://www.presentation-guru.com/how-to-give-a-great-presentation-with-just-your-phone/. Accessed 15 Jan 2018
8. Rathee G, Rakesh N (2016) Packet transmission (RPT) for the buffer based routing (BBR) protocol. J Inf Process Syst 12:57–72
9. Nurelmadina N, Nafea I, Younas M (2016) Evaluation of a channel assignment scheme in mobile network systems. Hum Cent Comput Inf Sci 6:21
10. Sarif BAB et al (2015) Fairness scheme for energy efficient H.264/AVC-based video sensor network. Hum Cent Comput. Inf Sci 5:7
11. Gilani SMM et al (2017) Mobility scenarios into future wireless access network. J Inf Process Syst 13:236–255
12. Doulamis AD, Doulamis ND, Kollias SD (2003) An adaptable neural-network model for recursive nonlinear traffic prediction and modeling of MPEG video sources. IEEE Trans Neural Networks 14:150–166
13. Al Tamimi AK, Jain R, So-In C (2010) Dynamic resource allocation based on online traffic prediction for video streams. Paper presented at the 4th IEEE international conference on internet multimedia services architecture and application (IMSAA), Bangalore, India
14. Qadir QM et al (2015) A novel traffic rate measurement algorithm for quality of experience-aware video admission control. IEEE Trans Multimedia 17:711–722
15. Kastrinakis M, Badawy G, Smadi MN, Koutsakis P (2017) Video frame size modeling for user-generated traffic in an enterprise-like environment. Comput Commun 109:24–37
16. Xu Z (2017) Adaptive flow admission control in a software-defined network. Paper presented at the IEEE international conference on smart cloud, New York, USA
17. Raghunathan V et al (2004) Energy efficient wireless packet scheduling and queuing. ACM Trans Embed Comp Syst 3:3–23
18. Procissi G et al (2002) Token bucket characterization of long-range dependent traffic. Comput Commun 25:1009–1017
19. Fiddler M, Sander V (2004) A parameter based admission control for differentiated services networks. Comput Netw 44:463–479

20. Rathgeb EP (1991) Modeling and performance comparison of policing mechanisms for ATM networks. IEEE J Sel Areas Commun 9:325–334
21. Kong PY, Chua KC, Bensau B (2003) A novel scheduling scheme to share dropping ratio while guaranteeing a delay bound in a multicode-CDMA network. IEEE/ACM Trans Netw 11:994–1006
22. Cisco (2014) Comparing traffic policing and traffic shaping for bandwidth limiting. http://www.cisco.com/en/US/tech/tk543/tk545/technologies_tech_note09186a00800a3a25.shtml. Accessed 15 Jan 2018
23. Gao F, Qian H (2016) Efficient real-world token bucket configuration for residential gateways. IEEE/ACM Trans Netw 24:462–475
24. Ash GR (2006) Traffic engineering and QoS optimization of integrated voice & data networks. Morgan Kaufmann, Burlington
25. Koutsakis P (2009) Dynamic vs. static traffic policing: a new approach for videoconference traffic over wireless cellular networks. IEEE Trans Mob Comput 8:1153–1166
26. Maratsolas E, Koutsakis P, Lazaris A (2014) Video activity-based traffic policing. IEEE Trans Multimedia 16:1446–1459
27. Marpe D, Wiegand T, Sullivan G (2006) The H.264/MPEG4 advanced video coding standard and its applications. IEEE Commun Mag 44:134–143
28. Video traces for network performance evaluation. http://trace.eas.asu.edu/tracemain.html. Accessed 3 Dec 2017
29. FFmpeg Organization. About FFmpeg. https://www.ffmpeg.org/about.html. Accessed 15 Jan 2018
30. Microsoft Corporation (2014) PowerShell. https://technet.microsoft.com/en-us/library/bb978526.aspx. Accessed 15 Jan 2018
31. Techex, X264. http://www.techex.co.uk/codecs/x264. Accessed 3 Dec 2017
32. Screen Mirroring Video Datasets (2017). http://users.isc.tuc.gr/~makastrinakis/docs/Screen_Mirroring_Video_Trace_Datasets.zip. Accessed 15 Jan 2018
33. Wu J et al (2013) Recurrent leaky bucket. IEEE Signal Process Lett 20:1244–1248
34. Ccorimanya JQ, Ling LL (2017) Traffic control in the transmission queue of LTE system downlink based on policing algorithms. Paper presented at the IEEE XXIV international conference on electronics, electrical engineering and computing (INTERCON), Cusco, Peru
35. Heyman DP (1997) The GBAR source model for VBR videoconferences. IEEE/ACM Trans Netw 5:554–560
36. Jaccard P (1908) Nouvelles recherches sur la distribution florale. Bulletin de la Societe Vaudoise des Sciences Naturelles 44:223–270
37. Wang X (2016) Traffic control method, device and storage medium. US Patent 20160028635A1. https://www.google.com/patents/US20160028635. Accessed 15 Jan 2018