Traffic Splitting for Delay Jitter Control in Multi-access Systems

Megha Sahu[1*], Sri Pramodh Rachuri[1], Ahtisham Ali Ansari[1] and Arzad Alam Kherani[1]

[1]Department of Electrical Engineering and Computer Science, IIT Bhilai, Raipur, 492015, Chhattisgarh, India.

*Corresponding author(s). E-mail(s): meghas@iitbhilai.ac.in;
Contributing authors: rachuris@iitbhilai.ac.in;
ahtishama@iitbhilai.ac.in; arzad.alam@iitbhilai.ac.in;

Abstract

This work addresses the problem of traffic splitting for improving the overall delay jitter performance in the uplink multi-access system. We propose a packet-scheduling paradigm based on stochastic approximation algorithm to distribute the source traffic across the multiple network paths/interfaces. We first provide an analytical model and the delay jitter analysis for an individual interface. Later we formulate the traffic splitting problem as an optimization problem to learn the optimal split across the interfaces. We share the experimental results for the video and Constant Bit Rate (CBR) traffic on real networks (Wi-Fi or cellular networks) and convergence of our system using the proposed scheme in the dynamic network environment. The paradigm proposed in the paper is general and can be adapted to different objective functions.

Keywords: Multi-Access Systems, Delay Jitter

1 Introduction

Many multi-media services in the online applications, such as Facebook live, Instagram stories, Whatsapp video calls, and on-demand video applications like Youtube, Netflix, Hotstar, are becoming a part of our daily lives. Most of the upcoming media applications require 4k/8k video quality streaming

services. We have been observing a rapid growth in the development of fast multi-media technologies and applications such as on-demand video, online cloud gaming, and video conferencing platforms responsible for the heavy amount of video traffics in the network [1]. It has been reported by Cisco and Ericsson in [2] [3] that most of the global mobile network traffic is consumed by video traffic (around 55% and this will keep on increasing through 2021).

In the area of medical emergencies for immediate treatment/response. A vision of "Smart Ambulance" is getting worldwide attention and has been emphasized in the last few years [4], [5] to enhance patient care in the area of active safety and emergency response. This work is motivated by the use-case of massively connected smart ambulance to stream real-time Ultra High Definition (UHD) video to the central healthcare provider for in-time inputs/instructions (when an ambulance is carrying a patient to the nearest emergency and trauma center).

The main challenge is to deliver high-quality video streaming services while ensuring the user's need (measured in Quality of Experience (QoE)) as well as improving the network performance. To stream such high-quality video in the uplink requires a bandwidth of around $40 - 50$ $Mbps$ for $4K$ videos sampled at 50 *frames per sec (fps)* [6], and achieving this speed from a single cellular/wireless network interface is impractical. One of the solutions to tackle this heavy video traffic is *Multi-access Systems* as depicted in Figure 1, where multiple wireless network interfaces (like, Wi-Fi and LTE) are connected to the mobile system. Figure 1 emphasizes the LTE user plane protocol stack for the LTE connection with the different Service Data Units (SDU) at the Packet Data Convergence Protocol (PDCP) and Media Access Control (MAC) layers.

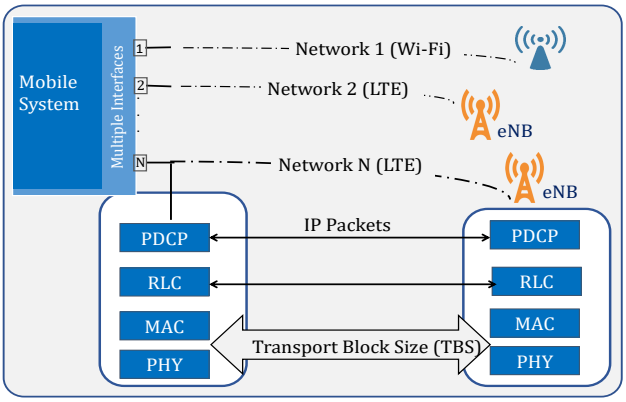

Fig. 1: A Multi-Access System with Wi-Fi and LTE connections, and emphasizing the user plane protocol stack interaction for the LTE connection in a multi-access system.

A multi-access system provides data transmission from multiple connected interfaces to achieve higher data rates by aggregating the bandwidths available on each network interface. However, such multi-access capability increases the system performance but, there are still many challenges that we face while transmitting the incoming source traffic over the multiple heterogeneous network interfaces. The challenges are as follows:

1. *Controlling the number of interfaces* to be used for transmitting the given source stream.
2. *Controlling the split of traffic* into the available/selected interfaces.

In the first challenge, we use the optimal number of interfaces due to the following reasons: 1. To reduce the complexity of the traffic splitting scheme that increases with an increase in the number of interfaces, 2. To avoid the wastage of resources due to unused interfaces. Once we select the interfaces, we start streaming data across them. But the question of *how to split among the multiple interfaces* remains. The task of traffic distribution is more difficult in diverse networks leading to delay differences among the interfaces. Moreover, during transmission, packets arrive in an out-of-order manner at the destination further impacting the delay jitter of the overall output stream.

This work focuses on dealing with the second challenge *controlling the split of traffic* and *provide an application layer solution for traffic splitting on the per-packet level over the given number of interfaces to improve the overall delay jitter performance in the multi-access system.*

The main contributions of this work can be presented as follows:

- *Delay Jitter Modelling for an Interface*: We first provide an analytical model based on the packet scheduling on an individual interface and the delay jitter analysis for it. We give a closed form expression for the delay jitter.
- *Online Scheme for Traffic Splitting*: We propose an online adaptation algorithm to adapt the amount of traffic to be transmitted on each interface in the changing network environment through an effective feedback mechanism to improve the delay jitter performance of the system. Further, we formulate the traffic splitting problem as an optimization problem and show that the algorithm is general and can be applied to different objective functions.

The first challenge, *Controlling the number of interfaces* is not part of the current work. In one of our previous works [7], we have highlighted the challenges in limiting the delay and jitter characteristics in the multi-access system and given a solution to control their performance.

The paper is structured as follows: Section 2 presents the related work. Section 3 covers a brief introduction of multi-access systems and video streaming across them. It also gives an analytical model for the delay jitter on an interface. Section 4 consists of the proposed online traffic splitting adaptive algorithm to find the optimal traffic split. In Section 5, we give a real-world implementation of the proposed online adaptive algorithm and share the experimental results. Finally, Section 6 gives the conclusion of our work.

2 Related Work

In recent years, the growth in video streaming applications has resulted in thousands of peer-reviewed publications. We categorize a few relevant pieces of literature into three groups: multi-access solutions, video traffic scheduling in multi-access systems, and delay jitter control techniques.

2.1 Multi-access Solutions

In the past, various solutions and protocols were provided at different Open Systems Interconnection (OSI) layers to implement multi-access control [8]. Many existing multi-path transmission protocols are Multi-path Transmission Control Protocol (MPTCP) [9], Concurrent Multi-path Transfer for Stream Control Transmission Protocol (CMT-SCTP) [10], and Multi-path Real-time Transport Protocol (MPRTP)[11].

We use the widely used video streaming protocol RTP protocol [12]. It is a UDP-based protocol used for unidirectional real-time video streaming applications. The advantage of this protocol is that it has a very low overhead. It has an inbuilt property to manage the QoS performance in conjunction with the RTP control protocol (RTCP) to send the QoS reports. In multi-path approaches, the extension of RTP is MPRTP [11] and MRTP [13]. Both MPRTP and MRTP use Constant Bit Rate (CBR) approaches. In terms of congestion control, many pro-priority congestion control algorithms are proposed, but none of them are integrated into the protocol. MPRTP and MRTP use the RTCP report to manage the traffic by categorizing congestion based on packet loss whereas we use the delay jitter measurements. Talking about MRTP, it uses Multiple Description Coding (MDC) as an error resilience technique good for high lossy networks but is improper for low loss rate networks due to increased overhead. Thus recently, an Scalable Video Coding (SVC)-based video bit rate adaptive solution using MPRTP is provided in [14] that outperforms MDC in low loss rate network scenarios. It considers the throughput and packet loss rate of each path. But [14] is an overlay-based approach that generally leads to additional protocol overheads. Thus, we prefer the control to be available at the application layer, noting that the developed logic can be incorporated in other multi-path schemes like MPTCP and MPRTP. In terms of traffic distribution, the MRTP has not any dedicated method and MPRTP is a bit unfair to traffic distribution decisions. In our approach, we also try to provide an adaptive approach for optimal traffic splitting.

One deployment [15] uses Mobile Bonding Routers (MBRs) that provide aggregate high-speed data connection by using multiple cellular dongles. Many other commercial solutions are Speedify (Connectify.me), Peplink, Mushroom Networks, Viprinet, Turnium Technology, Waav Technology, Zifilink, etc.

2.2 Video Traffic Streaming Solutions in Multi-access System

The multi-access capability is extensively used to stream high-quality video. Few video streaming solutions are available in [16–20].

One of the recent solutions, smartstreamer [17], proposed a preference-aware multi-path video streaming algorithm over HTTP using MPTCP. The incoming video stream is divided into chunks with different quality levels and then forwarded to the selected link to optimize the user's QoE depending on the network situation (bandwidth prediction) and buffer size. It is a non-convex optimization problem constrained by the available bandwidth and link preference. An extension [18] proposed an adaptive application-layer throughput-based approach without MPTCP, and this effort is for most of the content providers coming up that do not support MPTCP like, latency controlled adaptive application layer LEAP in [21]. One of the client-based video streaming solutions named MSPlayer [22] presents high-quality video streaming in mobile scenarios where it adjusts the size of video chunks depending on the bandwidth available on the paths. The experiments are using Wi-Fi and LTE resulted in reduced video start-up delays. A cross layer approach is presented in [16] for optimized video streaming solution over multi-paths.

An application layer MPEG Media Transport (MMT)-based approach [19] proposed path-and-content-aware scheduling to improve mobile multi-path video streaming. It improves the user-level experience and provides a solution to work in the rapid network change scenarios. The purpose is more for broadcasting and mostly server requirement QoS performance, not the client. Another, application layer solution is provided in [23] to minimize the latency of the live video streaming service. A Content-Aware Scheduling (CMT-CA) [24] proposes an optimized frame scheduling technique for delivering low-delay HD video streaming over heterogeneous wireless networks.

Loss-Aware Throughput Estimation (LATE) scheduler based on MPTCP in [25] considers the impact of packet loss in multi-path on receiving out-of-order packets across heterogeneous network paths. An online adaptive multi-path scheduler named Peekaboo [26] proposes a stochastic adjustment strategy to adapt to the dynamic characteristics of the heterogeneous network paths (discussed for Wi-Fi and LTE). It fails in the mobility scenarios due to a lack of data to learn the frequent changing network scenarios. An offline solution to overcome this issue is suggested in the work but yet not incorporated in the algorithm. One such solution Reinforcement Learning-based Scheduler for MPTCP (ReLes) [27] which has used the deep learning techniques to learn a neural network that generates an online adaptive packet scheduling strategy based on the offline learning of the network/environment.

[28] gives a dynamic traffic splitting technique for real-time video traffic between Wi-Fi and cellular networks to reduce the delay and energy consumption. The problem customizes the Lyapunov drift plus penalty optimization approach for achieving the desired output. Few other, traffic splitting solutions

between Wi-Fi and cellular networks are discussed in [26, 29, 30], and few others on different networks like, [31] for LTE/HSPDA, [32] for Wi-Fi/UMTS, [33] for Wi-Fi/HSDPA.

2.3 Delay Jitter Control Techniques

Most of the existing solutions in the literature have used the multi-access capability to improve the throughput and delay performance of the system, however, one can also look at the problem of improving end-to-end delay jitter performance for real-time applications that require such guarantees.

One of the works is the Jitter Buffer Mechanism explained in [34] (post-processing jitter control mechanism implemented at the receiver side) that tries to give smooth video output but often results in the freezing of the video when the buffer length drops to zero [35]. One of the past works, Nearly Equal Delay Path Set Configuration (NEED-PC) [36] focuses on live-streaming applications and tends to reduce jitter by sending duplicate packets over two different paths with nearly similar delays. This intends to find the redundant path to satisfy delay and jitter.

A novel attempt to control delay jitter in [37] proposed a pipeline network coding-based solution to solve the out-of-order packet arrival problem in the multi-path transmission. Jitter-Aware Packet Scheduling (JAPS) [38] has focused on throughput performance by overcoming the limitations of existing scheduling techniques (DAPS [39], Round Robin) that performs badly in high jitter networks.

Further, [40] presents a heuristic approach to minimize the delay in the network by establishing the explicit delay bounds to the paths. However, it does not reduce the delay difference across the interfaces and for which an approach is suggested in [41] to minimize delay difference amongst paths utilized for concurrent multi-path scheduling.

In the field of medical health care, [42] reports a health monitoring system and designs a jitter-free packet scheduler to improve the control performance in a mobile gait rehabilitation system. An extensive delay jitter modeling is provided in [43] for two different traffics (Poisson and deterministic) to facilitate the design of a jitter efficient packet scheduler.

3 System Model

In this section, we first present a brief introduction of the multi-access system and a demonstration of video traffic splitting across it. We define the performance metric delay jitter for the received stream of IP packets and provide a delay jitter analysis for a single interface. Besides, we summarize the notations used in the paper in TABLE 1.

3.1 The Multi-Access Set-up

Figure 2 represents a multi-access system, where the sender is connected to N multiple wireless network interfaces (e.g., Wi-Fi and/or cellular) across which the source traffic is transmitted. We focus on uplink transmission from the sender to the receiver. The traffic source can be a stored media file/camera (live streaming) or a free-running packet generator. The traffic splitter block consists of the interface selection logic for transmitting the source IP packet and thus deciding the distribution across the interfaces.

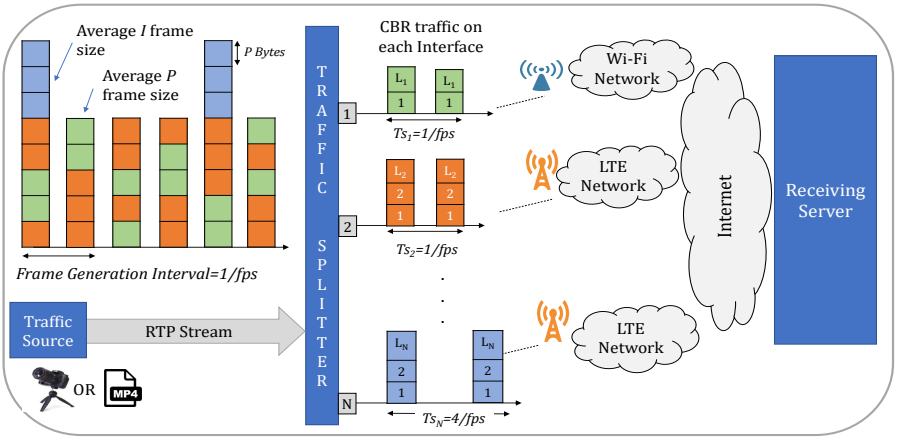

Fig. 2: A Multi-Access System and video traffic splitting across multiple interfaces.

3.2 Video Streaming Traffic

For high-quality video streaming applications $H.264$ Advanced Video Coding (AVC) is preferred. $H.264$ AVC is a video compression technique where the encoded stream generates three types of frames: I, P, and B. I is Intra-frame that is least compressed and doesn't require other frames for decoding. P is Predictive-frame that is decompressed using the previous I/P frame, whereas B is Bidirectional-frame that refers to both preceding and succeeding frames. As per the [44], we say that the sizes of I, P, and B frames are approximately in the ratio $6.6 : 2 : 1$.

It is clear, therefore, that when an option of multiple cellular uplink interfaces is available to a system designer, she will try to ensure that each interface is given only that number of IP packets which can be transmitted using the ongoing TBS (in LTE) on that interface. Figure 2 depicts the application layer traffic splitting solution for the RTP stream which would generate a huge frame size periodically $(1/fps)$. Even though the encoding (say, $H.264$) would introduce significant variations in the number of IP packets for each video

Table 1: Description of Notations.

Notation	Description
L	CBR parameter: Burst size (in *Packets*).
P	CBR Parameter: Packet size (in *Bytes*).
T_s	CBR Parameter: Burst generation interval (in sec).
R	CBR traffic generation rate, $\frac{L \times P \times 8}{T_s}$ *bps* on an interface.
R_j, S_j	Receive timestamp of j^{th} packet received on an interface and S_j is sender timestamp associated with it.
J	Average delay jitter.
L_i	Random size burst generated on i^{th} arrival on an interface.
d_i	Random uplink scheduling delay experienced by the i^{th} burst arrived (measured by delay experienced by first IP packet served from L_i). We assume d_i is exponentially distributed with parameter λ.
$w_{j,i}$	Random delay experienced by $j = 2$ *to* L_i packets after the first packet scheduled in the burst L_i. We assume $w_{j,i}$ is uniformly distributed between interval $[0, \theta]$.
ω_i	Sum of uniform random variable $\{w_{j,i}\}$ following Irwin-Hall distribution with the parameters 0, θ and $(L-1)$.
N	The total number of interfaces used in a multi-access system.
λ_n, θ_n	Delay parameters on interface n.
J_n, L_n	Average delay jitter on interface n and CBR burst size sent on it.
$n(m)$	Packet count used for the m^{th} packet received on interface n.
k	The index for the update instants.
$\mathrm{p}(k)$	Interface probability vector at k.
$\mathrm{J}(k)$	Delay jitter vector obtained in feedback from the receiver at k.
$\mathrm{T}(k)$	Reward vector at instant k formed by inverting jitter vector i.e., $1/\mathrm{J}(k)$.
$\sigma(\mathrm{T}(k))$	Softmax function over $\mathrm{T}(k)$.
p_n	Probability to choose n^{th} interface for transmitting the incoming IP packet.
$U_n(\cdot)$	Utility function of interface n.
$\mathcal{S}_n(\cdot)$	Direction function which provides the additive or subtractive correction to the interface probabilities.

frame (due to inter-frame coding, like in I, P, B frames), *when splitting the video traffic into multiple interfaces, our objective is to ensure that a particular interface should converge to a CBR traffic with a moderate number of IP Packets per frame*. Here, the effective period seen by the different interfaces would be at least frame interval or multiple of it (for the case when smaller video frames like P or B, skips any interface making traffic interval more than $1/fps$ on it) (as shown in the Figure 2).

We consider the problem of splitting a video (RTP) stream into N available interfaces while presenting each interface with a CBR traffic composed of bursts of L IP packets of length P *Bytes* being sent into an interface every T_s time. It is to be noted that the burst size (in *Bytes*) is interface-dependent. Based on the value of T_s, we have classified the traffic in two regimes: 1. *Single Interface with Lazy Arrivals (SILA)*: where T_s is very large such that the burst $(L \times P)$ *Bytes* arrived on an interface is served completely before the next burst arrival. It has been covered in [6] for the LTE system, where delay jitter analysis is done under the uplink LTE-Hybrid Automatic Repeat Request

(HARQ) control on an interface for the CBR traffic and the adaptation algorithm is proposed for a static end-device. Further, we extend our work for the mobile scenario (under significant changes in system parameters) in [45]. In this paper, for the RTP traffic to be in $SILA$, the sampling rate (fps) considered should be ≤ 60 always which makes frame generation interval at least 15 ms [6]. 2. *Fluid Model*: Where T_s is very small such that most of the time, enough IP packets are waiting in the PDCP queue to get the service. It is covered in [46] for a single interface in the LTE, we provide the delay jitter modeling and map our system to the $M/G/1$ queuing model to obtain the closed-form expression for the delay jitter in terms of $M/G/1$ parameters.

3.3 Performance Metric: Delay Jitter

Delay jitter is measured as the variation in the delays of two consecutively received IP packets. Using the definition given in [11], we define the per-packet delay jitter for the j^{th} received packet as,

$$J_j = \left|(R_j - R_{j-1}) - (S_j - S_{j-1})\right|, \tag{1}$$

where R_j is the receive timestamp of the j^{th} packet received on an interface and S_j is the sender timestamp associated with it. The average delay jitter is given by,

$$J = \lim_{l \to \infty} \frac{1}{l} \sum_{j=1}^{l} J_j. \tag{2}$$

3.4 Delay Jitter Modelling for an interface with the CBR traffic

In this section, we propose a packet-level scheduling model for the generated traffic under the $SILA$ regime and provide the delay jitter analysis for a single network interface in the multi-access system.

In the multi-access system, given source traffic is to be split among N available interfaces where each interface converges to different L values (shown in Figure 2). However, in the presence of multiple network interfaces, initially, a random amount of packets are generated over an interface. A typical model in Figure 3 shows a random burst of IP packets (bursts) generation on an interface at every T_s and the service given to them on the per-packet level. Let L_i be the random amount of packets (burst) generated on i^{th} arrival and gets served before the next burst arrival under the $SILA$ regime. Figure 3 also shows the sender and receive timestamps of the IP packets and different delays involved while serving the i^{th} arrived burst. Let $R_{j,i}$ be the receive timestamp of the j^{th} received packet on i^{th} burst arrival and $S_{j,i}$ is the sender timestamp associated with it. Please note that the sender timestamp of all the generated packets belonging to L_i is the same.

Let $\{d_i\}$ be the scheduling delay experienced by the i^{th} arrived burst. Now, while transmission in the uplink, all the packets in L_i will not be received at

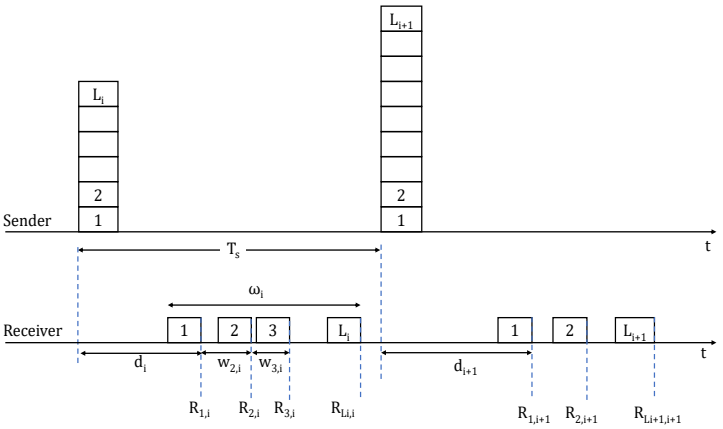

Fig. 3: *SILA* Model with delays and timestamps for an interface.

the same time but with a delay of d_i (measured by the delay experienced by the first IP packet served in the burst L_i) and $w_{j,i} + d_i$ from j ranging from 2 *to* L_i. Here, $\{w_{j,i}\}$ is the random delay experienced by $j = 2$ *to* L_i packets after the first packet scheduled in the burst L_i. Then, from the Figure 3, we deduce the average jitter in the i^{th} burst arrival as,

$$J_i = \frac{1}{L_i}\Big(\big|R_{2,i} - R_{1,i}\big| + \big|R_{3,i} - R_{2,i}\big| + \ldots + \big|R_{L_i,i} - R_{L_i-1,i}\big| + \big|(R_{1,i+1} - R_{L_i,i}) - T_s\big|\Big), \tag{3}$$

here, difference in sender timestamp of all the received packets belonging to burst L_i is $S_{j,i} - S_{j-1,i} = 0$, $\forall\ j = 2$ *to* L_i except for the first received packet in the arrival and the last received packet of previous arrival, that is $S_{1,i+1} - S_{L_i,i} = T_s$, *for* $j = 1$, which will contribute in the jitter of first served packet from the arrived burst.

$$\begin{aligned} J_i &= \frac{1}{L_i}\left[\sum_{j=2}^{L_i}\big|(R_{j,i} - R_{j-1,i})\big| + \big|(R_{1,i+1} - R_{L_i,i}) - T_s\big|\right], \\ &= \frac{1}{L_i}\left[\sum_{j=2}^{L_i} w_{j,i} + \left|d_{i+1} - d_i - \sum_{j=2}^{L_i} w_{j,i}\right|\right]. \end{aligned} \tag{4}$$

In Equation 4, the distribution of $\{d_i\}$ depends upon the scheduling scheme and load in the network in terms of the number of users connected to it and actively using the network, as well as $\{w_{j,i}\}$ depends upon the network condition and possibly on the value L_i also. Taking $\sum_{j=2}^{L_i} w_{j,i} = \omega_i$, we modify

Equation 4 and get,

$$J_i = \frac{1}{L_i}[\omega_i + |d_{i+1} - d_i - \omega_i|], \tag{5}$$
$$= \begin{cases} \frac{1}{L_i}(d_{i+1} - d_i), & \omega_i < d_{i+1} - d_i \\ \frac{1}{L_i}(2\omega_i + d_i - d_{i+1}), & \omega_i > d_{i+1} - d_i \end{cases}$$

In order to analyse further for a particular interface, we are assuming that the random variable $\{d_i\}$ is exponentially distributed with parameter λ (observed in [6]) and $\{w_{j,i}\}$ is uniformly distributed over the interval $[0, \theta]$. The $\{\omega_i\}$ is the sum of uniform random variable $\{w_{j,i}\}$ following Irwin-Hall distribution with the parameters 0, θ and $(L-1)$. Thus the Equation 5 for the average delay jitter becomes,

$$J_i = \frac{1}{L_i} \int_{\omega=0}^{(L_i-1)\theta} \int_{y=-\infty}^{\infty} (\omega + |y - \omega|) \frac{\lambda}{2} e^{-\lambda|y|} dy \cdot d\omega, \tag{6}$$

where $y = d_i - d_{i+1}$ (difference between two independent and identically distributed random variables) and the probability density function of y is $\frac{\lambda}{2}e^{-\lambda y}$. The final average delay jitter for an interface is obtained as,

$$J = \frac{1}{L}\left[(L-1)\theta + \frac{1}{\lambda}\left(\frac{1 - \exp^{-\lambda\theta}}{\lambda\theta}\right)^{L-1}\right]. \tag{7}$$

Equation 7 can be approximated by expanding $\exp^{-\lambda\theta}$, where $\lambda, \theta \geq 0$, we thus obtain,

$$J = \frac{1}{L}\left[(L-1)\theta + \frac{1}{\lambda}(1 - \frac{\lambda\theta}{2})^{L-1}\right].$$

On further approximation by using binomial expansion of $\left(1 - \frac{\lambda\theta}{2}\right)^{L-1}$ for $\left|\frac{\lambda\theta}{2}\right| < 1$, we simplify the above Equation as,

$$J(L) = \frac{1}{L}\left[\frac{1}{\lambda} + \frac{3}{2}(L-1)\theta\right], \tag{8}$$

for $0 \leq \lambda\theta \leq 2$.

Here the parameters λ and θ are estimated at the receiver using the receive timestamp of the stream of IP packets. The λ and θ are channel-dependent parameters and different for each network interface depending on the network condition. The higher value of λ and lower θ show better network conditions and vice versa for the bad network scenarios. Equation 8 is the closed-form expression obtained for the average delay jitter of an interface. It gives the relationship between the average delay jitter with the L values for a known λ

and θ on an interface. This signifies for known delay parameters, L is decided such that it improves the delay jitter performance of that interface.

On considering N multiple interfaces, for the average delay jitter of n^{th} interface Equation 8 becomes,

$$J_n(L_n) = \frac{1}{L_n}\left[\frac{1}{\lambda_n} + \frac{3}{2}(L_n - 1)\theta_n\right], \tag{9}$$

Here, each interface should converge to a CBR traffic with the best achievable L values that tend to reduce the overall delay jitter of the system.

Further, in the case of multiple interfaces, while splitting the traffic, we need a clear understanding of the present network scenario to distribute the adequate amount of traffic that the particular interface can handle at that time. Continuous learning is required so that even in the case of network-changing conditions, the optimal traffic distribution can be achieved by improving the overall delay jitter performance. Thus, we propose an online adaptation algorithm in the next section for traffic splitting in a multi-access system.

4 Traffic Splitting Algorithm

In this section, we propose a traffic splitting algorithm similar to a reinforcement learning mechanism to split the source traffic across multiple interfaces. Unlike heuristic approaches [47], [48] that uses fixed control rules based on simplified models of the network environment for which no theoretical guarantees are available. The proposed technique utilizes the delay jitter analysis for an interface and attempts to learn the traffic spitting from observations, which is more adaptive to the varying network conditions.

We consider N interfaces in the multi-access system and interface $n \in \{1, 2, \ldots, N\}$. Let, $\mathrm{p} = (p_1, p_2, \ldots, p_N)$ be the interface probability vector and $\mathrm{J} = (J_1, J_2, \ldots, J_N)$ be the delay jitter vector. Here, p_n is the probability to choose n^{th} interface for transmitting an incoming source IP packet and J_n is the average delay jitter measure associated with interface n.

In our scheme, the proposed algorithm is responsible for determining the traffic distribution on the sender side based on the average delay jitter obtained in feedback from the receiver, which would decide further the fraction of arrived source burst given to interface n i.e., L_n. Since the number of packets sent on each interface can be determined by interface probability, p_n, our approach would include *Stochastic Approximation Algorithm* where p_n gets updated at different instants at the sender.

We define the reward function as the inverse of the cost of the interfaces denoted as $T_n(p_n)$ given as follows:

$$T_n(p_n) = \frac{1}{J_n(p_n)}, \tag{10}$$

where J_n is the delay jitter of interface n which can be obtained either via Equation 8 or via direct estimation.

4.1 Proposed Online Adaptation Algorithm

In the multi-access system, based on the average delay jitter obtained in feedback, the interface probability of each interface is updated at the sender in the following manner,

$$p_n(k+1) = p_n(k) + \epsilon \cdot \mathcal{S}_n(\sigma(\mathrm{T}(k))_n - p_n(k)), \tag{11}$$

here, k denotes the update instants and interface update its interface probability such that $\sum_{n=1}^{N} p_n = 1$. The $\mathcal{S}_n(\cdot)$ is a direction function which provides the additive or subtractive correction to the interface probabilities and $\epsilon > 0$ is the learning rate. $\mathrm{T}(k) = (T_1(k), T_2(k), \ldots, T_N(k))$ is a reward vector formed by inverting the delay jitter vector obtained in feedback, where $T_n(k) = \frac{1}{J_n(k)}$ given in Equation 10.

Also, $\sigma(\cdot)$ is the soft-max function applied over the received reward, whose i^{th} component is obtained using the following expression,

$$\sigma(\mathrm{T}(k))_i = \frac{\exp^{T_i(k)}}{\sum_{n=1}^{N} \exp^{T_n(k)}},\ i = 1, 2, \ldots, N.$$

Here, the output of $\sigma(\mathrm{T}(k))$ also lie in the N-dimensional probability simplex.

It is to be noted that the convergence speed is controlled by ϵ that depends on $\left|\sigma(\mathrm{T}(k))_n - p_n(k)\right|$ and the update rate of each interface would be different but the convergence point would not change. Due to this probability vector, $\mathrm{p}(k+1)$ the random burst sizes on the interfaces are produced (as modeled in Section 3.4). But once the system converges then at equilibrium, we can say $L_n = \lceil p_n * L \rceil$ making traffic on each interface CBR.

4.2 System Equilibrium

At equilibrium, the update algorithm given in Equation 11 converges for all the interfaces and becomes,

$$p_n = \sigma(T)_n.$$

By using [49], we can say there exist a function $F_n(\cdot)$ such that $J_n = F_n(p_n)$ and we assume $F_n(\cdot)$ is an increasing function: higher delay jitter for the larger interface probability $(*)$. It is to be noted that the function $F_n(\cdot)$ depends only on the probability of the interface n and would not be affected by the other interfaces available in the system, like their network variations and conditions.

The equilibrium point achieved on the convergence of Equation 11 can be viewed as the solution of an optimization problem for traffic splitting in a multi-access system and the objective function of each interface would be,

$$U_n(p_n) - \frac{p_n^2}{2}, \tag{12}$$

where $U_n(p_n) = \int \sigma(T)_n dp_n$ is the utility function defined for an interface and the goal is to maximize the aggregate interface utility over their interface probabilities.

*_Remark:_ The assumption made for $F_n(\cdot)$ is based on the observation made from the obtained results in [6] for LTE, where we have observed that on increasing burst sizes more than the allocation (Transport Block Size) provided by the base station (eNodeB) on an interface increases the delay jitter. Therefore the paradigm proposed in our paper would work for all such $F_n(\cdot)$ and applicable to different objective functions.

We observe the objective function (Equation 12) is strictly concave when the following condition holds: $U_n''(p_n) - 1 < 0$, under the assumption $F_n(\cdot)$ exists such that T_n decreases with increase in p_n.

We consider the term $U_n''(p_n)$ and prove it comes < 0 and the second term $(-\frac{p_n^2}{2})$ is already < 0. We have,

$$U_n''(p_n) = \frac{\partial}{\partial T_n} \frac{\exp^{T_n}}{\sum_{i=1}^{N} \exp^{T_i}} \cdot \frac{\partial T_n}{\partial p_n} \tag{13}$$

$$= \frac{C \exp^{T_n}}{(C + \exp^{T_n})^2} \cdot \frac{\partial T_n}{\partial p_n}. \tag{14}$$

where $C > 0$ is the constant replacing all the terms except $i = n$ as F_n is not influenced by the other interface performances. Equation 14 would always come $< 0, \forall n$ as $\frac{C \exp^{T_n}}{(C+\exp^{T_n})^2}$ is positive and $\frac{\partial T_n}{\partial p_n} < 0$.

Now, we characterize interface n by {objective function (12), $0 < p_n \leq 1$}, where $0 < p_n \leq 1$ is the range of interface probability, respectively. We may assume without loss of generality that $p_n > 0$ for all n, because interface n with $p_n = 0$ can be omitted from consideration for transmission across it. Let $I_n = (0, 1]$ denote the range in which probability p_n must lie. Thus, the objective is to choose interface probabilities p_n so as to

$$\max_{p_n \in I_n} \quad \sum_{n=1}^{N} \left(U_n(p_n) - \frac{p_n^2}{2} \right), \tag{15}$$

$$\text{subject to} \quad \sum_{n=1}^{N} p_n = 1. \tag{16}$$

Though Equation 15 is separable in p_n but the interfaces are coupled by the probability constraint Equation 16.

4.3 A Fixed Point Formulation for $N = 2$ and Experimental Result

In this section, we consider the simplest model of $N = 2$ ($n = 1, 2$) to evaluate the performance of the system. We present the experimental results and validate our proposed algorithm with the fixed point obtained from the analytical model (Equation 19) using two interfaces.

Let the two interfaces be named IF1 and IF2. We utilize the results obtained in Section 3 for delay jitter and get the average delay jitter on IF1 by using Equation 8 as,

$$J_1 = \frac{1}{L_1}\left[\frac{1}{\lambda_1} + \frac{3}{2}(L_1 - 1)\theta_1\right], \tag{17}$$

and on IF2 as,

$$J_2 = \frac{1}{L - L_1}\left[\frac{1}{\lambda_2} + \frac{3}{2}(L - L_1 - 1)\theta_2\right], \tag{18}$$

where L_2 is replaced by $L - L_1$.
We have observed in Section 4, at the equilibrium we have $p_n = \sigma(T)_n$ and at this point we get $L_n = \lceil p_n * L \rceil$, this gives the following relation,

$$L_n = \frac{L}{1 + \exp^{\left(\sum_i \frac{1}{J_i} - \frac{1}{J_n}\right)}}, \forall\ n = 1, 2, \ldots, N$$

where i is all the other interfaces except n^{th} one i.e., $i = 1..n-1, n+1, ..N$.
Now, for $N = 2$, we get,

$$L_1 = \frac{L}{1 + \exp^{\left(\frac{1}{J_2} - \frac{1}{J_1}\right)}}, \tag{19}$$

We get the value of L_1 directly from Equation 19 and L_2 by $(L - L_1)$. The obtained (L_1, L_2) is the fixed point for the case $N = 2$. In Equation 19, J_1 and J_2 are given in Equation 17 and 18, where parameters $(\lambda_n, \theta_n, \forall\ n)$ are known and estimated at the receiver from the stream of received IP packets.

Now, to validate the analytical model, we experiment on real networks by using the set-up shown in Figure 6 for which the brief explanation is given in Section 5.1. We generate the CBR traffic from the source with $L \times P = 20 \times 300\ Bytes$ at every $T_s = 80\ ms$. We use two cellular modems that are USB tethered to the sender machine. We implement the adaptation algorithm using Equation 11 on real networks (with $\epsilon = 0.05$) where source CBR traffic is split over the two interfaces. At the receiver, We estimate the required parameters for both the interfaces (L_n information is attached in the packet header indicating the burst number packet belongs to while transmitting from the sender, and we extract this at the receiver, parameters λ_n and

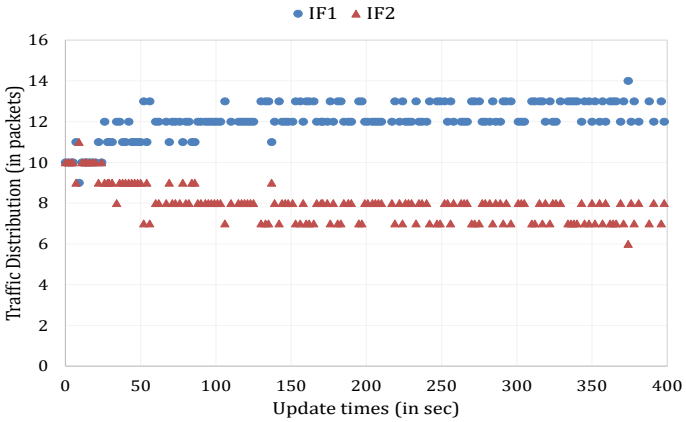

Fig. 4: Convergence of traffic distribution (in packets) with time on applying the adaptation algorithm at the sender.

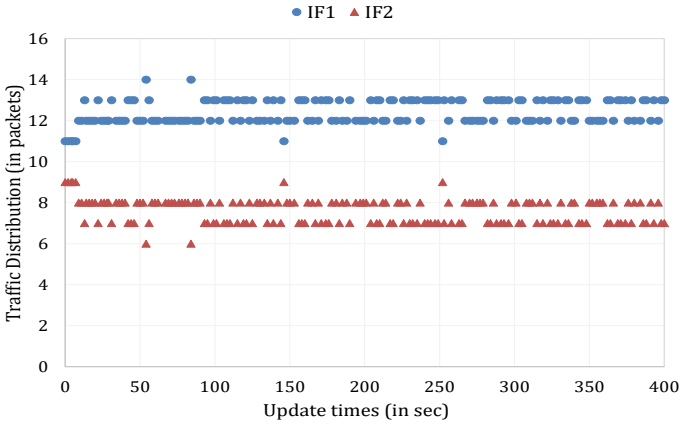

Fig. 5: Traffic distribution (in packets) with time calculated using the Equation 19.

θ_n are obtained using the receive timestamp of the packets). Then the J_n, $\forall\, n$ is calculated using Equation 8 and sent to the sender for adaptation of the interface probabilities that decide, in turn, the splitting of traffic i.e., L_n values.

Figure 4 shows the convergence of the traffic distribution with time (parameter update times) at the sender on using the adaptation algorithm and Figure 5 shows the traffic distribution calculated by directly using Equation 19 after getting the average delay jitter for each interface using Equation 17 and 18 at the receiver. It is to be noted that the adaptation of the L in the experiment (shown in Figure 4) is done using the algorithm not by using Equation 19.

We observe from Figure 4, IF2 performs badly, and thus the burst sent across it decreases and converges at around $L_2 = 8$ *packets*, and IF1 is better

compared to IF2 and converges to $L_1 = 12$ *packets*. This implies interface probability converges to $p_1 \approx 0.6, p_2 \approx 0.4$ based on the interface performance.

We observe from Figure 5, the L_1 and L_2 slowly converges to the 12 and 8 packets as obtained in Figure 4 on using the online adaptive algorithm. This shows that the fixed point achieved from the model and convergence using the online adaptive algorithm is the same. This validates our model against the experimental results obtained in the real networks.

5 Performance Evaluation

In this section, we give the real-world implementation of the proposed algorithm for the CBR and RTP traffic with $N = 4$ interfaces and share the experimental results that illustrate its convergence in a slowly time varying environment.

5.1 Experimental Set-up

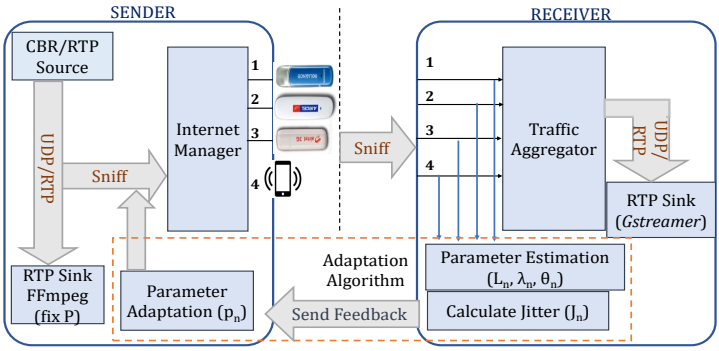

Fig. 6: Experimental Set-up.

In the basic experimental set-up shown in Figure 6, we have two end-machines for Sender and Receiver, where the sender has total $N = 4$ network interfaces connected to it (three USB-tethered to a cellular modem, and one is connected through the Wi-Fi to the smartphone). The receiver has a fixed public IP address. We generate CBR/RTP traffic from the traffic source and then use Python's sniff command of the Scapy library to extract these CBR/RTP packets. The captured packets are then processed by the Interface Manager and encapsulated into well-formed UDP packets. The UDP packets are transmitted over the selected interface from the sender machine to the assigned destination port. At the receiver, we again use Python's sniff to capture all the CBR/RTP streams arriving at different receiving ports. In the case of RTP traffic, the source video is played out internally at the sender machine using *FFmpeg* (video stream handling free software). We fix the packet size while generating from *FFmpeg* and simultaneously sniff the packets to send over the

interfaces. We estimate the parameters at the receiver used to calculate the average delay jitter required to give in feedback to the sender end. Once the feedback is received we adapt the interface probabilities at the sender. While the process is going on, simultaneously at the receiver the RTP streams are aggregated and played out using the *Gstreamer* (media player). A detailed description of the set-up used can be found in [50].

5.2 Learning with Delayed Reward and Traffic adaptation

Algorithm 1: Proposed Traffic Adaptation Algorithm

Connect N interfaces to the sender system.

Inputs:

Generate RTP stream from $FFmpeg$ with P ;

Send to Local host: (IP,Port);

Initialize:

ϵ, W ;

$k \leftarrow 0$ (update instant) ;

$\mathrm{p}(k) \leftarrow [1/\mathrm{N}, \forall\ \mathrm{N}]$ at update instant k ;

$n(m) \leftarrow 0$ (packet count on interface n);

SENDER:

Capture a packet from Local host ;

Select an interface $n \in [1, N]$ using $\mathrm{p}(k)$;

Send packet to the Receiver system via n ;

RECEIVER:

{ Receive packet ;

Extract n from destination port information;

Increment $n(m)$;

while $n(m) > 0, \forall\ n$ **do**

Calculate per packet Jitter from Eq.(1),

$J_{n(m)} = \left|(R_{n(m)} - R_{n(m-1)}) - (S_{n(m)} - S_{n(m-1)})\right|$;

Store receive timestamp.;

if $n(m) \geq W, \forall\ n$ **then**

Estimate λ_n and θ_n using receive timestamp ;

Calculate average jitter using Eq. (9) ;

Send average Jitter (Eq. (9)) to the Sender;

$n(m) \leftarrow 0, \forall\ n$;

}

if *Jitter received* **then**

Calculate reward using Eq. (10) ;

Update $\mathrm{p}(k)$ using Eq. (11) ;

Increment k ;

In our experiments, the sender does not receive immediate evaluative feedback for each action it performs. This is a more realistic scenario because, in the real world, the delay jitter has significant variations after a certain duration. Thus the reward (or jitter) is fed back to the sender system after receiving at least W packets on each interface. In our experiments, we apply the adaptation rule Equation 11 and correction is applied on each interface to update the interface probabilities at update instant $k+1$. We compare the reward $(T_n(k))$ (obtained by reciprocating $J_n(k)$ received in feedback) and current probability $(p_n(k))$ of that interface, i.e., if $(T_n(k) > p_n(k))$, then we increase probability with $\|\epsilon \cdot (\sigma(\mathrm{T}(k))_n - p_n(k))\|$ value otherwise decrease with $\|\epsilon \cdot (\sigma(\mathrm{T}(k))_n - p_n(k))\|$. Here, we have decided the ϵ at the beginning of the experiment. The increments and decrements are not static step size of ϵ but we have taken $\|\epsilon \cdot (\sigma(\mathrm{T}(k))_n - p_n(k))\|$ amount (becomes dynamic). The Algorithm is summarized for the RTP stream in Algorithm 1.

5.3 Performance Analysis

All the experimental results included in this section are real-world implementation with $N = 4$ $interfaces$. We name the interfaces as {IF1, IF2, IF3, IF4}. Here, IF1 is the Wi-Fi hotspot of the mobile phone, and IF2, IF3, IF4 are USB tethered to the sender system. We do socket programming in Python. We include the results for the *adaptation* using the algorithm and *no adaptation* cases for a better comparison of the proposed scheme. We also change the application source rate at the middle of the run to see how our algorithm is adaptable to such change. The experimental results in Figure 9, 10, and 13 are plotted at the receiver whereas Figures 8-11 are at the sender machine. The experiments are performed with the CBR and RTP traffic. The details of the experimental parameters set are given in Table 2.

The following are the results and the observations made:

5.3.1 CBR Traffic with $N = 4$ interfaces

The CBR traffic is generated from the source with $L \times P = 50 \times 300 = 15\ KBytes$ at every $T_s = 300\ ms$. For the convergence the learning rate (ϵ) used is 0.05.

Figure 7 and 8 represent the variation in the interface probability and traffic (on packet-level) distribution on each interface with time on applying the adaptive rule for the CBR traffic. Initially, the channel conditions are unknown, and thus, the probabilities are set equal $(0.25, \forall\ n\ = 1\ to\ 4)$ and traffic is evenly distributed as shown in Figure 7 and 8 at the start of the experiment. At this point, we observe that $L = 50$ is evenly distributed with burst around $12 - 13$ in Figure 8. After 750 *secs* since the start of experiment, we observe $IF3$ and $IF4$ start performing better, and $IF1$ and $IF2$ started deteriorating. This influence is also reflected in Figure 8 when we plot the traffic distribution at the sender side, the $IF3$ and $IF4$ start getting more traffic on them.

Table 2: Experimental Parameters.

Traffic type	**CBR**
P, L, T_s	300 *Bytes*, 50 *Packets*, 300 *ms*
N, ϵ	4 interfaces, 0.05
Initialize: p	$[0.25, 0.25, 0.25, 0.25]$ (as $N = 4$)
Experiment Duration	$\approx$ 26 *mins*
Traffic type	**RTP**
Traffic source	Stored video file
Video resolution	$480P$
Sampling rate	30 *fps*
Average frame size	20,000 *Bytes*
P	200 *Bytes* (fixed while generation from *FFmpeg*)
N, ϵ	4 interfaces, 0.1
Initialize: p	$[0.25, 0.25, 0.25, 0.25]$ (as $N = 4$)
Experiment duration	$\approx$ 21 *mins*

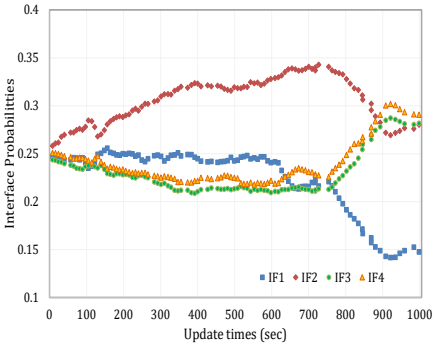

Fig. 7: Interface Probabilities on applying adaptation.(Traffic: CBR, $N = 4$)

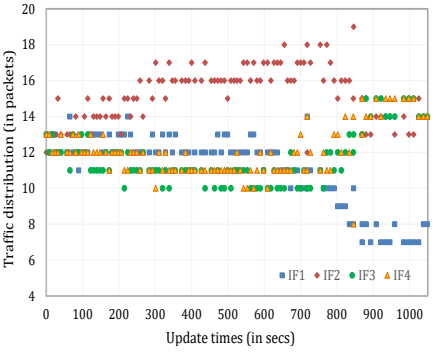

Fig. 8: Traffic distribution on applying adaptation.(Traffic: CBR, $N = 4$)

5.3.2 RTP Traffic with $N = 4$ interfaces

We generate the RTP stream from the stored video file. The video stream is first sent internally to the local port on the sender system using *FFmpeg* having $P = 200$ *Bytes* at sampling rate 30 *fps*. The stream is first saved in the queue and then sent across the interfaces to the assigned destination port. Here, we have increased the learning rate to $\epsilon = 0.1$ to speed up the convergence rate as the video duration (or experiment) is less. Please note that improving/controlling the convergence rate is not part of this work. During implementation, the frame arrival times are calculated by using the time difference between the two consecutive *FFmpeg* generated video packets where packet counts are done within the frames.

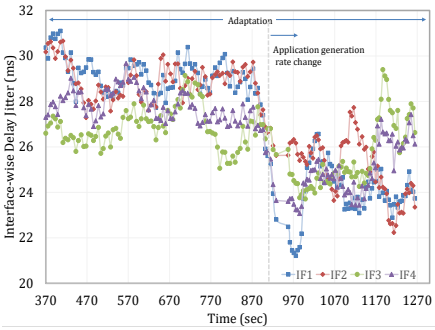

Fig. 9: Interface-wise Delay jitter performance using Model (Equation 9). (Traffic: RTP, $N = 4$)

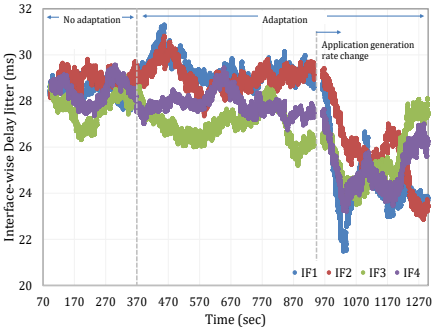

Fig. 10: Interface-wise Delay jitter performance in Experiment (Equation 1). (Traffic: RTP, $N = 4$)

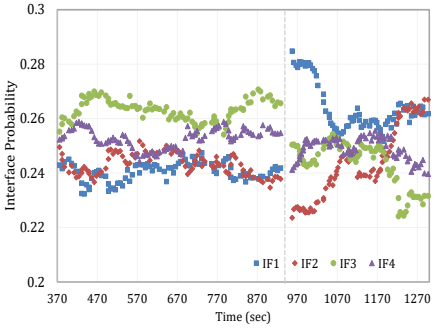

Fig. 11: Interface probabilities on applying adaptation. (Traffic: RTP, $N = 4$)

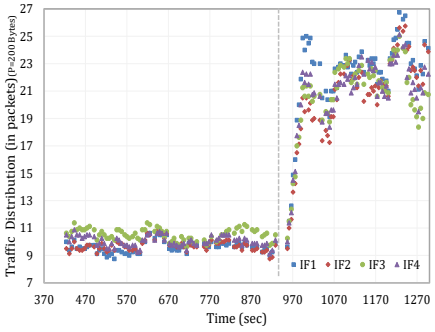

Fig. 12: Traffic distribution on applying adaptation. (Traffic:RTP, $N = 4$)

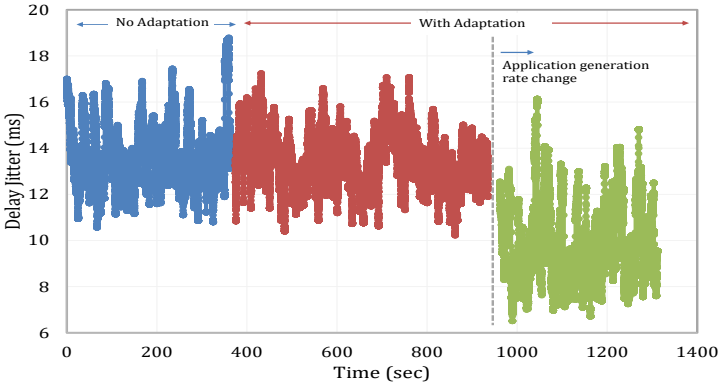

Fig. 13: Overall delay jitter variation with time (Without adaptations and adaptation in the same run). (Traffic: RTP, $N = 4$)

We experiment with $N = 4$ interfaces for the RTP traffic. We start the experiment with $R = 300$ $Kbps$ without implementing the adaptation rule for 300 $secs$ and then apply the adaptation mechanism. We start the run using adaptation for the 300 $Kbps$ video rate for 12 $mins$ then, we change the R to 500 $Kbps$. Figure 9 shows the delay jitter performance on each interface estimated using the model (Equation 9), which will be given in the feedback to the sender system. In our experiment, we simultaneously validate the model against the per-packet jitter measurement (Equation 1) shown in Figure 10. At the sender, Figure 11 and 12 give the interface probability and traffic distribution on each interface on applying the adaptation algorithm. Figure 13 gives the overall delay jitter of the output stream. We observe the following:

- From Figure 9 and 10, we observe the delay jitter from the model is validated against the jitter measured in the experiment using Equation 1.
- In the case of *no adaptation* region in Figure 10 and 13, the jitter on the interfaces are within $26 - 30$ ms and varies closely. On implementing *adaptation* algorithm, interface-wise jitter behaves differently based on the varying traffic on each interface due to the adaptation mechanism that tries to stabilize the overall delay jitter even if the jitter on IF1 and IF2 is high.
- At the start of *adaptation*, in Figure 9 and 10, we see the jitter in order $J_3 < J_4 < J_2 < J_1$ that shows the interface probabilities in order $p_3 > p_4 > p_2 > p_1$ (Figure 11) that in turn affects the traffic distribution on each interface in the same order as of probabilities (Figure 12).
- On increasing the video rate to 500 $kbps$, the delay jitter on each interface starts decreasing and comes within $23 - 27$ ms (Figure 10) which also reduces the overall delay jitter performance shown in Figure 13. Slowly, the order of jitter performance on each interface seems to change in a manner $J_2 < J_1 < J_4 < J_3$ (seen at the end of run). Also, the traffic on each interface almost gets doubled as shown in Figure 12. We can apply the adaptation algorithm in such changes in application rate and try to achieve optimal traffic splitting such that the overall system performance improves.

In all of the above experiments, the packet loss is within 1%.

6 Conclusion

In this paper, we have provided an analytical model for the delay jitter of an interface. We have proposed an online adaptive algorithm for the traffic splitting across the cellular and Wi-Fi heterogeneous networks to improve the delay jitter performance in the uplink. Furthermore, we have formulated the traffic splitting problem as an optimization problem. The performance of the proposed algorithm was evaluated using the real-world experimental setup reported to handle the video traffic in a multi-access environment. In the future, the proposed solution can be extended by incorporating the interface selection and splitting the traffic across the selected ones.

7 Declarations

8 Conflicts of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

[1] Sandvine Global Internet Phenomena Report. White Paper, Sandvine,Waterloo, ON, Canada (Jun. 2013)

[2] Index, C.V.N.: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast update, 2014–2019. Technical Report (2015)

[3] Ericsson Mobility Report. Technical Report (2016)

[4] Rehman, I.U., Nasralla, M.M., Ali, A., Philip, N.: Small Cell-based Ambulance Scenario for Medical Video Streaming: A 5G-health use case. In: International Conference on Smart Cities: Improving Quality of Life Using ICT IoT (HONET-ICT), pp. 29–32 (2018)

[5] The vision of 'Smart Ambulances' goes hand-in-hand with a data-empowered paramedic. The Journal of mHealth (2018)

[6] Sahu, M., Damle, S., Kherani, A.A.: Traffic Splitting for End-to-End Delay Jitter Control in Uplink Multi-Access Systems. International Conference on COMmunication Systems NETworkS (COMSNETS) (2019)

[7] Sahu, M., Rachuri, S.P., Ansari, A.A., Tandur, D., Kherani, A.A.: On limiting Delay and Jitter characteristics at application-layer of Multi-connected Systems. In: IEEE 5G World Forum (5GWF), pp. 36–41 (2020)

[8] Schmidt, P.S., Merz, R., Feldmann, A.: A First Look at Multi-access Connectivity for Mobile Networking. In: ACM Workshop on Capacity Sharing, pp. 9–14 (2012)

[9] TCP Extensions for Multipath operation with Multiple Addresses. RFC 6824 (2013)

[10] Iyengar, J.R., Shah, K.C., Amer, P.D., Stewart, R.: Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-end Paths. IEEE/ACM Transactions on Networking, 951–964 (2006)

[11] Singh, V., Ahsan, S., Ott, J.: MPRTP: Multipath Considerations for Real-time Media. In: Multimedia Systems Conference, MMSys '13, pp. 190–201 (2013)

[12] RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (2013)

[13] Mao, S., Bushmitch, D., Narayanan, S., Panwar, S.S.: MRTP: a multi-flow real-time transport protocol for ad HOC networks. IEEE Transactions on Multimedia **8**(2), 356–369 (2006)

[14] Pakulova, E., Miller, K., Wolisz, A.: Adaptive low-delay video streaming in heterogeneous wireless networks using MPRTP. International Wireless Communications and Mobile Computing Conference (IWCMC), 14–19 (2017)

[15] Streamcom. https://northcom.dk/streamcom-til-region-nordjylland/

[16] Deng, Z., Liu, Y., Liu, J., Argyriou, A.: Cross-layer DASH-based multipath video streaming over LTE and 802.11ac networks. Multim. Tools Appl. **80**, 16007–16026 (2021)

[17] Elgabli, A., Aggarwal, V.: SmartStreamer: Preference-Aware Multipath Video Streaming Over MPTCP. IEEE Transactions on Vehicular Technology **68**(7), 6975–6984 (2019)

[18] Elgabli, A., Liu, K., Aggarwal, V.: Optimized Preference-Aware Multi-Path Video Streaming with Scalable Video Coding. IEEE Transactions on Mobile Computing **19**(1), 159–172 (2020)

[19] Afzal, S., Rothenberg, C.E., Testoni, V., Kolan, P., Bouazizi, I.: Multipath MMT-based approach for streaming high quality video over multiple wireless access networks. Computer Networks **185**, 107638 (2021)

[20] Taha, Miran, Canovas, Alejandro, Lloret, Jaime, Ali, Aree: A QoE adaptive management system for high definition video streaming over wireless networks. Telecommunication Systems **77** (2021)

[21] Chiariotti, F., Kucera, S., Zanella, A., Claussen, H.: Analysis and Design of a Latency Control Protocol for Multi-Path Data Delivery With Pre-Defined QoS Guarantees. IEEE/ACM Transactions on Networking **27**(3), 1165–1178 (2019)

[22] Chen, Y., Towsley, D., Khalili, R.: MSPlayer: Multi-Source and Multi-Path Video Streaming. IEEE Journal on Selected Areas in Communications **34**(8), 2198–2206 (2016)

[23] Houzé, P., Mory, E., Texier, G., Simon, G.: Applicative-layer multipath for low-latency adaptive live streaming. In: IEEE International Conference on Communications (ICC), pp. 1–7 (2016)

[24] Wu, J., Yuen, C., Wang, M., Chen, J.L.: Content-Aware Concurrent

Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks. IEEE Transactions on Parallel and Distributed Systems **27**, 1–1 (2015)

[25] Yang, W., Dong, P., Cai, L., Tang, W.: Loss-Aware Throughput Estimation Scheduler for Multi-Path TCP in Heterogeneous Wireless Networks. IEEE Transactions on Wireless Communications **20**(5), 3336–3349 (2021)

[26] Wu, H., Alay, O., Brunstrom, A., Ferlin, S., Caso, G.: Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments. IEEE Journal on Selected Areas in Communications **38**(10), 2295–2310 (2020)

[27] Zhang, H., Li, W., Gao, S., Wang, X., Ye, B.: ReLeS: A Neural Adaptive Multipath Scheduler based on Deep Reinforcement Learning. In: IEEE INFOCOM - IEEE Conference on Computer Communications, pp. 1648–1656 (2019)

[28] Abbas, N., Hajj, H., Dawy, Z., Jahed, K., Sharafeddine, S.: An optimized approach to video traffic splitting in heterogeneous wireless networks with energy and QoE considerations. Journal of Network and Computer Applications **83**, 72–88 (2017)

[29] Zhang, X., Yang, F.: Joint Bandwidth and Power Allocation for Energy Efficiency Optimization over Heterogeneous LTE/WiFi Multi-Homing Networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2017)

[30] Harutyunyan, D., Herle, S., Maradin, D., Agapiu, G., Riggio, R.: Traffic-aware user association in heterogeneous LTE/WiFi radio access networks. In: NOMS 2018- IEEE/IFIP Network Operations and Management Symposium, pp. 1–8 (2018)

[31] Yang, R., Chang, Y., Sun, J., Yang, D.: Traffic Split Scheme Based on Common Radio Resource Management in an Integrated LTE and HSDPA Networks. In: IEEE Vehicular Technology Conference (VTC Fall), pp. 1–5 (2012)

[32] Wu, J., Yang, J., Chen, J.: Loss Tolerant Bandwidth Aggregation for multihomed video streaming over heterogeneous wireless networks. In: IEEE Global Communications Conference (GLOBECOM), pp. 2956–2962 (2013)

[33] Capela, N., Sargento, S.: Optimizing network performance with multi-homing and network coding. In: IEEE Globecom Workshops, pp. 210–215 (2012)

[34] Zhang, F.P., Yang, O.W.W., Cheng, B.K.-M.: Performance evaluation of jitter management algorithms. Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No.01TH8555) **2**, 1011–10162 (2001)

[35] Le, H., Nguyen, H., Pham Ngoc, N., Pham, A., Cong Thang, T.: A Novel Adaptation Method for HTTP Streaming of VBR Videos over Mobile Networks. Mobile Information Systems **2016** (2015)

[36] Okuyama, T., Yasukawa, K., Yamaoka, K.: Nearly Equal Delay Path Set Configuration (NEED-PC) for Multipath Delay Jitter Reduction. IEICE Trans. Commun. **91-B**(3), 722–732 (2008)

[37] Xu, C., Wang, P., Xiong, C., Wei, X., Muntean, G.: Pipeline Network Coding-Based Multipath Data Transfer in Heterogeneous Wireless Networks. IEEE Transactions on Broadcasting **63**(2), 376–390 (2017)

[38] Chan, M., Tseng, C., Yen, L.: Jitter-aware packet scheduler for concurrent multi-path transmission in heterogeneous wireless networks. In: IEEE Wireless Communications and Networking Conference, pp. 1–7 (2016)

[39] Kuhn, N., Lochin, E., Mifdaoui, A., Sarwar, G., Mehani, O., Boreli, R.: DAPS: Intelligent delay-aware packet scheduling for multi-path transport. In: IEEE International Conference on Communications (ICC), pp. 1222–1227 (2014)

[40] Devetak, F., Shin, J., Anjali, T., Kapoor, S.: Minimizing Path Delay in Multi-path Networks. In: 2011 IEEE International Conference on Communications (ICC), pp. 1–5 (2011)

[41] Shin, J., Devetak, F., Anjali, T., Kapoor, S.: Concurrent multi-path routing over bounded paths: Minimizing delay variance. In: IEEE Global Communications Conference (GLOBECOM), pp. 1483–1488 (2013)

[42] Leng, Q., Wei, Y.-H., Han, S., Mok, A., Zhang, W., Tomizuka, M.: Improving Control Performance by Minimizing Jitter in RT-WiFi Networks. Proceedings - Real-Time Systems Symposium, 63–73 (2015)

[43] Hammad, K., Moubayed, A., Shami, A., Primak, S.: Analytical approximation of packet delay jitter in simple queues. IEEE Wireless Communications Letters **5**, 1–1 (2016)

[44] Doggen, J., Van der Schueren, F.: Design and Simulation of a H.264 AVC Video Streaming Model. (2008)

[45] Sahu, M., Damle, S., Kherani, A.: End-to-end uplink delay jitter in LTE systems. Wireless Networks (2021)

[46] Sahu, M., Kherani, A.A.: End-to-End Delay Jitter in LTE Uplink: Simple Models, Empirical Validation Applications. In: International Conference on COMmunication Systems NETworkS (COMSNETS), pp. 221–228 (2020)

[47] Xing, M., Xiang, S., Cai, L.: A Real-Time Adaptive Algorithm for Video Streaming over Multiple Wireless Access Networks. Selected Areas in Communications, IEEE Journal on **32**, 795–805 (2014)

[48] Combes, R., Sidi, H.B.A., Elayoubi, S.-E.: Multi-path Streaming: Fundamental Limits and Efficient Algorithms. IEEE Journal on Selected Areas in Communications **35**(1), 188–199 (2017)

[49] Abraham, R., Marsden, J.E., Ratiu, T.: Manifolds, Tensors, Analysis, and Applications, 2nd ed. 1983 edn. New York: Springer-Verlag

[50] Damle, S., Sahu, M., Kherani, A.A.: An uplink multi-access system for high definition video transmission. International Conference on COMmunication Systems NETworkS (COMSNETS), 532–534 (2019)