Wasiur R. KhudaBukhsh, Bastian Alt, Sounak Kar, Amr Rizk, and Heinz Koeppl: Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies. To be published in Proceedings of IEEE International Conference on Computer Communications (INFOCOM), 2018.

Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies

Wasiur R. KhudaBukhsh*, Bastian Alt*, Sounak Kar†, Amr Rizk†, and Heinz Koeppl*
*Bioinspired Communication Systems Lab (BCS), {wasiur.khudabukhsh | bastian.alt | heinz.koeppl}@bcs.tu-darmstadt.de,
†Multimedia Communications Lab (KOM), {sounak.kar | amr.rizk}@kom.tu-darmstadt.de,
Technische Universität Darmstadt, Germany

***Abstract*—Collaborative uploading describes a type of crowdsourcing scenario in networked environments where a device utilizes multiple paths over neighboring devices to upload content to a centralized processing entity such as a cloud service. Intermediate devices may aggregate and preprocess this data stream. Such scenarios arise in the composition and aggregation of information, *e.g.*, from smart phones or sensors. We use a queuing theoretic description of the collaborative uploading scenario, capturing the ability to split data into chunks that are then transmitted over multiple paths, and finally merged at the destination. We analyze replication and allocation strategies that control the mapping of data to paths and provide closed-form expressions that pinpoint the optimal strategy given a description of the paths' service distributions. Finally, we provide an online path-aware adaptation of the allocation strategy that uses statistical inference to sequentially minimize the expected waiting time for the uploaded data. Numerical results show the effectiveness of the adaptive approach compared to the proportional allocation and a variant of the join-the-shortest-queue allocation, especially for bursty path conditions.**

I. Introduction

Internet of Things (IoT) describes a world of heterogeneous devices, such as sensors and actuators that are connected through various communication technologies while carrying out everyday tasks. Crowdsourcing in the context of IoT often refers to interconnected devices that ubiquitously *exchange* and *aggregate* information to achieve complex goals. For example, live events can be covered by composing many information streams originating from various mobile and fixed sources such as smart phones, audio/visual, and ambient sensors. Live events include not only entertainment events but also emergency situations, such as security breaches and attacks on civilians.

A common feature of many of these crowdsourcing devices is the ability to simultaneously utilize different sets of wireless and wired communication technologies, such as WiFi, cellular, Ethernet and powerline communication, and to further recognize and simultaneously interact with surrounding devices. Fig. 1 shows different examples of collaborative uploading scenarios. As depicted, it is crucial to understand how a primary device can best utilize the parallel paths provided by secondary devices for uploading its data.

Modeling the collaborative uploading problem intrinsically includes scenario-specific challenges as shown by the heterogeneous examples in Fig. 1. Nevertheless, we are interested in the uploading performance, *e.g.*, the time required to transfer a piece of data from a source device to a processing unit in an edge-cloud. An abstraction that enables powerful results on such performance measures is provided through queuing theory, as sketched in the bottom of Fig. 1. This connection-layer abstraction enables the source to make intelligent decisions as to how to utilize the available and possibly heterogeneous paths by only considering their latencies.

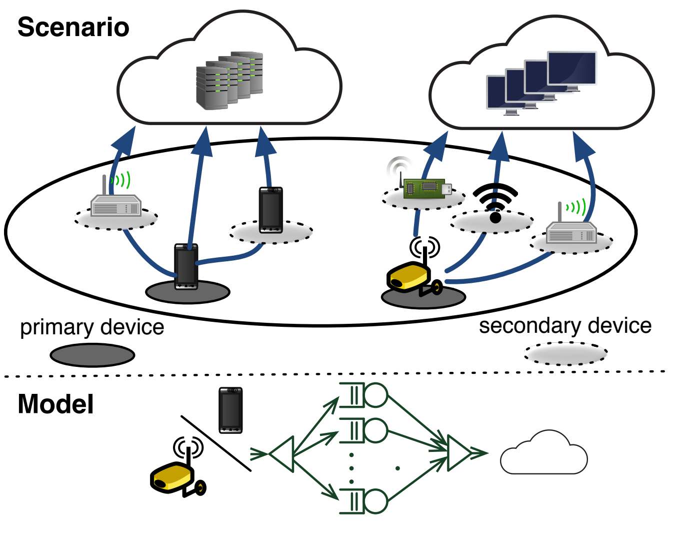

Fig. 1: Collaborative uploading: A device uses neighboring devices and different paths to upload a data stream. The scenario is naturally captured by a Fork-Join queuing system.

Our goal in this work is to find optimal collaborative uploading strategies in crowdsourcing scenarios. We differentiate between intermittent (devices such as sensors sending data on a coarse time scale) and continuous collaborative uploading (devices continuously streaming video footage, *e.g.*, using Facebook Live). In optimizing performance metrics such as the uploading time, we also make a distinction between the cases when devices possess full knowledge of the different path characteristics, and when they perform statistical inference.

In this paper, we analyze replication and allocation strategies for collaborative uploading scenarios. We use a Fork-Join (FJ) queuing system (see Fig. 1) that captures the ability to split data into chunks that are transmitted over multiple paths, and finally merged when all chunks are received. Our contributions are summarized as: **1)** closed-form expressions for the mean upload latency in the intermittent uploading case, allowing a comparison between a replication and an allocation (splitting) strategy. We find optimal strategies for given path latencies. In doing so, we also show numerical results suggesting near-optimality of the proportional allocation. **2)** Online path-aware adaptation of the allocation strategy based on statistical inference and stochastic gradient descent to sequentially minimize the expected waiting times in the continuous uploading case. We evaluate and compare the performance of our proposed adaptive strategy under various levels of path service burstiness.

The rest of this paper is organized as follows: in Sec. II, we outline our modeling approach for the intermittent as well as the continuous uploading case. The intermittent case is then considered in detail in Sec. III. In Sec. IV, we pose the continuous uploading system as a queuing theoretic one and use stochastic gradient methods to address the optimization of the allocation. In Sec. V, we furnish an evaluation study of our proposed online algorithm. Finally, we discuss related work in Sec. VI and summarize our findings in Sec. VII.

II. Modeling approach

Here, we present an overview of our approach, which consists of *(i)* defining an appropriate performance metric and *(ii)* framing an appropriate optimization problem thereafter.

We characterize the intermittent case as one where the time intervals between two successive uploads are so large that there is *no* self-induced queuing. Then, aspects such as cross-traffic can be described by means of the statistical properties of the path latencies alone. A primary device uploading data intermittently aims to minimize the upload latency, *i.e.*, the time until the data reaches the cloud. Given multiple paths over secondary devices, the primary device may split the data into chunks that are transmitted or replicated over the available paths. The upload latency being a stochastic quantity, it is natural to consider its mean as a performance metric and optimize it over all possible splitting/replication configurations. In Sec. III, we express the upload latency as an order statistic of the individual upload times over the different paths, making the theory of order statistics a useful tool in our analysis.

In the case of continuous upload of a data stream, *e.g.*, a primary device uploading a live video to the cloud, there is a notion of waiting before each data chunk can be uploaded and hence, that of queuing. We call the event of new data generation and passing by the application to the lower layers on the primary device, an arrival of a new data batch. Each data batch is split into chunks of various sizes that are transported over several paths. Paths are characterized by a random service time required to transport the assigned chunks. Finally, the data batch reaches the cloud when all of its chunks are received. Such systems are known as FJ queuing systems [1]–[3].

III. Intermittent Collaborative Uploading

In the following we consider the intermittent uploading case of data of size K over N possibly heterogeneous paths (*e.g.*, sensor or monitoring devices uploading data on a coarse time scale). Assume that the data can be divided into N smaller chunks consisting of packets. Then, every $\mathbf{k} = (k_1, k_2, \ldots, k_N) \in \Lambda(N,K)$ is a valid allocation vector, where k_i denotes the number of packets to be sent via path i and $\Lambda(N,K)$ denotes the set of all non-negative integer solutions of the Diophantine equation $\sum_{i=1}^{N} k_i = K$, for $N, K \in \mathbb{N}$. We denote the random amount of time taken to transport the j-th packet out of the k_i packets allocated to path i by $D_{i,j}$. Here, $D_{i,j}$ may capture different phenomena that impact the transmission time over a path, such as resource allocation, transmission collisions, and retransmissions. Assume that for each $i \in [N]$, with $[N] := \{1, 2, \ldots, N\}$, the random variables $D_{i,j}$'s are mutually independently distributed[1]. Recall that the data consisting of K packets can be reconstructed only after *all* the packets have arrived. Therefore, the upload latency can be expressed as $D := \max(D_1^{(k_1)}, D_2^{(k_2)}, \ldots, D_N^{(k_N)})$ where $D_i^{(k_i)} := \sum_{j=1}^{k_i} D_{i,j}$ for $k_i > 0$ denotes the amount of time taken by path i to transport k_i packets, and by convention, $D_i^{(0)} := 0 \,\forall i \in [N]$. The random variable D measures the total amount of time taken to transport *all* the packets over N different paths. In this work, we consider

$$\psi(\mathbf{k}) := \mathrm{E}[D] = \mathrm{E}\left[\max(D_1^{(k_1)}, D_2^{(k_N)}, \ldots, D_N^{(k_N)})\right],$$

the expected upload time given an allocation $\mathbf{k}$, as our performance metric. The density function of $D_i^{(k_i)}$ is given by the k_i-fold self-convolution of the density function of $D_{i,j}$ due to independence. Let us denote the cumulative distribution function (CDF) of $D_i^{(k_i)}$ by $F_i^{(k_i)}$. Stacking into a column vector $\mathbf{F}^{(\mathbf{k})} := (F_1^{(k_1)}, F_2^{(k_2)}, \ldots, F_N^{(k_N)})^\mathsf{T}$, we express the expected values of the order statistics of $D_1^{(k_1)}, D_2^{(k_2)}, \ldots, D_N^{(k_N)}$ as an operator μ on $\mathbf{F}^{(\mathbf{k})}$ (see Remark 1 in Appendix I-A). Since $\psi(\mathbf{k})$ is the first moment of the N-th order statistic, we get

$$\psi(\mathbf{k}) = \mu_N \, \mathbf{F}^{(\mathbf{k})} = \sum_{j \in [N]} (-1)^{j+1} \mathrm{M}_j \, \mathbf{F}^{(\mathbf{k})}, \tag{1}$$

where μ_N and M_j are operators defined in Appendix I-A. The optimal allocation is found by minimizing ψ, *i.e.*,

$$\mathbf{k}_{\text{opt}} := \underset{\mathbf{k} \in \Lambda(N,K)}{\arg\min} \; \psi(\mathbf{k}). \tag{2}$$

Note that when the path characteristics are unknown, we can perform statistical inference[2]. In the following, we show some illustrative examples with computable $\mathbf{k}_{\text{opt}}$ before generalizing this allocation scheme to include replication strategies.

A. The canonical two-path case

We consider the problem of finding the optimal allocation over two heterogeneous paths. Let $\mathbf{k} \in \Lambda(2,K)$ denote our

[1]Mutual independence, although not necessary for the subsequent analysis, is assumed for the sake of simplicity. In order to account for possible dependencies observed in real-world applications, one needs to additionally specify a correlation structure for these variables. This step is application-specific and is not easy in general. We do not attempt that in this paper.

[2]This is particularly important from an engineering perspective. The issue of statistical inference becomes more interesting in the context of continuous upload. We show examples in Sec. V.

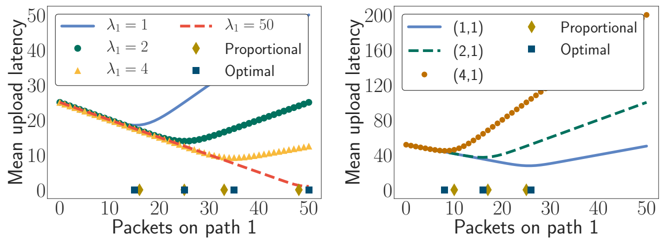

Fig. 2: **Canonical two-path case:** We plot the mean upload latency as a function of the number of packets allocated to path 1 out of overall 50 packets. **(Left)** Both paths have exponentially distributed delays. The rate of the first path λ_1 is increased from 1 to 50, while that of the second path is fixed at $\lambda_2 = 2$. Note the shift in the optimal allocation as λ_1 increases. **(Right)** Path 1 has Weibull delay with (scale, shape) parameters given in the legend while path 2 has lognormal delay with parameters 0 and 0.25. Observe that the optimal allocation is indeed close to the proportional allocation.

allocation. The corresponding upload latency is given by $D := \max(D_1^{(k_1)}, D_2^{(k_2)})$ and its mean is $\psi(\mathbf{k}) = \mu_2\, \mathbf{F}^{(\mathbf{k})} = \mathrm{E}\left[D_1^{(k_1)}\right] + \mathrm{E}\left[D_2^{(k_2)}\right] - \int_0^\infty (1-F_1^{(k_1)}(x))(1-F_2^{(k_2)}(x))\,\mathrm{d}x$. Suppose the packet latencies $D_{1,j}$ and $D_{2,j}$ are exponentially distributed with rates λ_1 and λ_2. Then, setting $p = \frac{\lambda_1}{\lambda_1+\lambda_2}$, $q = 1-p$, and $r = \frac{1}{\lambda_1+\lambda_2}$, the expected upload time is

$$\psi(\mathbf{k}) = \frac{k_1}{\lambda_1} + \frac{k_2}{\lambda_2} - r \sum_{n_1=0}^{k_1-1} \sum_{n_2=0}^{k_2-1} \binom{n_1+n_2}{n_1} p^{n_1} q^{n_2}.$$

To minimize the above, we derive the following relation through algebraic manipulation in [4]

$$\psi(k_1, k_2) \gtreqless \psi(k_1+1, k_2-1) \iff \frac{I_p(k_1, k_2)}{I_{1-p}(k_2-1, k_1+1)} \gtreqless \frac{\lambda_2}{\lambda_1},$$

where $I_x(a, b)$ is the regularized β-function. This allows finding the optimal allocation $\mathbf{k}_{\text{opt}}$ (see [4, Appendix B]).

When K is large, the optimal strategy can be found by numerically solving the following nonlinear equation

$$\frac{I_p(x, K-x)}{I_{1-p}(K-x-1, x+1)} - (\frac{1}{p} - 1) = 0.$$

In this case, the optimal allocation on path 1 is one of the two nearest integers producing a lower mean upload latency.

In Fig. 2, we consider the canonical two-path scenario for different choices of path-specific delay distributions and show the mean upload latency as a function of the number of packets on path 1. For distributions not admitting a closed-form expression for the mean upload latency, *e.g.*, Weibull and lognormal, we performed numerical integration.

Near-optimality of proportional allocation: A comparison with [5], [6]: The two-path scenario has been studied in [5], [6] for the exponential delay model. The authors, however, do not compute a closed-form expression for the mean upload latency and only provide the following upper bound, based on a Chernoff technique

$$\psi(k_1, k_2) \le \max\left\{\frac{k_1}{\lambda_1}, \frac{k_2}{\lambda_2}\right\} + \sqrt{2\pi}(\sqrt{\frac{k_1}{\lambda_1^2}} + \sqrt{\frac{k_2}{\lambda_2^2}}) \text{ (due to [5]).}$$

Based on the above bound, the authors characterize the optimal allocation as being either the proportional allocation, *i.e.*, $(k_1, k_2) = (Kp, K - Kp)$ or the winner-takes-it-all allocation, *i.e.*, $(k_1, k_2) = (K, 0)$. In contrast, we provide exact closed-form expression for the mean upload delay and find the optimal allocation $\mathbf{k_{opt}}$. Interestingly, we observe near-optimality of the proportional allocation, *e.g.*, as shown in Fig. 3 (left) for exponential path delays. In Fig. 2, we see that similar conclusions hold for Weibull and lognormal delays as well.

B. The N-path case with exponential delays

We next consider the general case of N paths available for uploading K packets of data as sketched in Fig. 1. Suppose the i-th path has exponential delay with rate λ_i. The mean upload latency of the allocation $\mathbf{k} \in \Lambda(N, K)$ is given by

$$\psi(\mathbf{k}) = \sum_{S \in \{A \subseteq [N] : A \neq \phi\}} (-1)^{|S|+1} \sum_{0 \le n_i \le k_i - 1 : i \in S} \left(\prod_{i \in S} \frac{\lambda_i^{n_i}}{n_i!} \right) \times \frac{\Gamma(\sum_{i \in S} n_i + 1)}{(\sum_{i \in S} \lambda_i)^{\sum_{i \in S} n_i + 1}}.$$

The outer summation is carried out over all non-empty subsets of $[N]$. The derivation is provided in [4]. The closed-form expression of $\psi(\mathbf{k})$ for more than two paths has not been provided before, to the best of our knowledge.

In [4], we provide a simplified expression for the mean upload latency for three heterogeneous paths with exponential delays and optimize it. The near-optimality of the proportional allocation is observed for this example also.

The allocation strategies discussed so far inherently impose a synchronization constraint at the destination. At a certain overhead, one way to circumvent this synchronization constraint is replication, which we consider next.

C. Replication strategies

A basic replication strategy is to send the entire data over all available paths and take the first chunk that arrives at the destination. Replication strategies are known to reduce latency in some regimes [7]. However, an apparent drawback is their overuse of resources, *e.g.*, higher energy consumption. Roughly put, replication replaces the max operation (requiring the last chunk to arrive to complete the data at the receiver) with a min operation (taking the first to arrive at the receiver). However, the min operation is taken over elements that stochastically dominate the elements over which the max operation is taken. This poses an interesting trade-off: *When should we replicate, and not allocate?*

In the basic replication case, the upload latency is $D := \min(D_1^{(K)}, D_2^{(K)}, \ldots, D_N^{(K)})$ where $D_i^{(K)} := \sum_{j=1}^{K} D_{i,j}$, as before. Our objective remains minimizing the mean upload latency

$$\phi(N, K) := \mathrm{E}[D] = \mathrm{E}\left[\min(D_1^{(K)}, D_2^{(K)}, \ldots, D_N^{(K)})\right] = \mu_1\, \mathbf{F}^{(K\upsilon)},$$

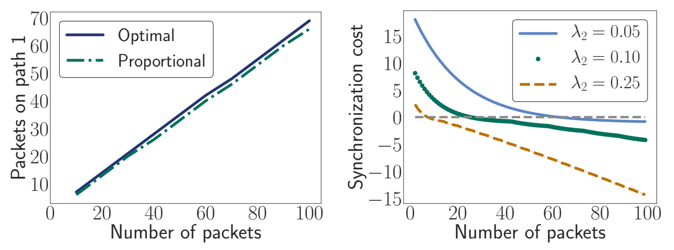

Fig. 3: **(Left) Near-optimality of proportional allocation:** The number of packets allocated to path 1 vs. the overall number of packets (data size) for two exponentially distributed path latencies with parameters $\lambda_1 = 4$ and $\lambda_2 = 2$. **(Right) The synchronization cost as a function of data size:** We consider two paths having exponential delays with rates $\lambda_1 = 1$ and λ_2 in the legend. Recall from (3) that positive (negative) synchronization cost implies superiority of replication (allocation). For large data sizes, it is better to allocate than to replicate. However, if one of the paths is much slower compared to the other one, the synchronization cost is high and consequently, replication may become profitable.

where υ is an N-dimensional vector of all ones and $\mathbf{F}^{(K\upsilon)} = (F_1^{(K)}, F_2^{(K)}, \ldots, F_N^{(K)})$. We favor the replication strategy if $\phi(N,k)$ is smaller than the mean upload latency of **any allocation** $\mathbf{k} \in \Lambda(N,K)$, *i.e.*, if $\phi(N,K) \le \min_{\mathbf{k}\in\Lambda(N,K)} \psi(\mathbf{k})$. In relation to the question of replication versus allocation, we introduce next the notion of synchronization cost.

Synchronization cost: Suppose all available paths are used for transmission and let $\Lambda^*(N,K) := \{\mathbf{k} = (k_1, k_2, \ldots, k_N) \in \Lambda(N,K) \mid k_i > 0 \quad \forall \quad i \in [N]\}$ denote the reduced set of valid allocations. Within Λ^*, an allocation can be worse than a replication essentially because of the synchronization at the destination, *i.e.*, because of some paths being much slower than others. To compare with a replication strategy, we define the synchronization cost given N paths and data size K as

$$\begin{aligned}\chi(N,K) &:= \min_{\mathbf{k}\in\Lambda^*(N,K)} \psi(\mathbf{k}) - \phi(N,K) \\ &= \min_{\mathbf{k}\in\Lambda^*(N,K)} \mu_N\, \mathbf{F}^{(\mathbf{k})} - \mu_1\, \mathbf{F}^{(K\upsilon)}. \qquad (3)\end{aligned}$$

If χ is positive, replication yields smaller mean upload latency and hence, is preferred. If χ is negative, we prefer allocation over replication. Intuitively, if the data size is large, we expect the cost of redundancy to be high and χ to be negative.

Consider the canonical two-path example with exponential delays from Sec. III-A. A straightforward computation of $\mu_1\, \mathbf{F}^{(K\upsilon)}$ yields the following closed-form expression of the synchronization cost defined in (3),

$$\chi(2,K) = \min_{(k_1,k_2)\in\Lambda^*(2,K)} \psi(k_1,k_2) - r \sum_{n_1=0}^{K-1}\sum_{n_2=0}^{K-1} \binom{n_1+n_2}{n_1} p^{n_1} q^{n_2}.$$

In Fig. 3, we show the synchronization cost as a function of the data size K. As the data size increases the cost of redundancy worsens the performance of replication. Consequently, an allocation strategy is preferred for large data. However, the *zero-crossing* data size seen in Fig. 3, which marks the regimes where replication and allocation are more beneficial, shifts depending on path heterogeneity.

D. *Combined Allocation and Replication: An (N,r)-strategy*

Here, we present a variant of the replication strategy, called the (N,r)-strategy. An (N,r)-strategy splits data of size K into N smaller chunks so that the data batch can be reconstructed from any r out of the N chunks. One of the ways to achieve such a splitting is to use Erasure codes, *e.g.*, maximum distance separable (MDS) codes [8]. Note that an (N,N)-strategy corresponds to allocation and an $(N,1)$-strategy, to replication. To formulate an (N,r)-strategy, we define

$$\Upsilon(N,r,K) := \{\mathbf{k} \in [K]^N \mid \sum_{i\in S} k_i \ge K \,\forall\, S \subseteqq [N], \mid S \mid = r\}.$$

We call a $\mathbf{k} \in \Upsilon(N,r,K)$ an (N,r)-allocation for data of size K. The data is received as soon as the first r out of N chunks arrive at the destination. Let the order statistics corresponding to $D_1^{(k_1)}, D_2^{(k_2)}, \ldots, D_N^{(k_N)}$ be denoted by $C_1 \le C_2 \le \ldots \le C_N$. The mean upload latency for $\mathbf{k} \in \Upsilon(N,r,K)$ is $\eta_r(\mathbf{k}) := \mathrm{E}[C_r] = \mu_r\, \mathbf{F}^{(\mathbf{k})}$. In Appendix I, we provide an example of (N,r)-allocations given three heterogeneous paths with exponential delays. For a fixed r, the optimal (N,r)-allocation is given by $\mathbf{k}_{\text{opt}}^{(r)} := \arg\min_{\mathbf{k}\in\Upsilon(N,r,K)} \eta_r(\mathbf{k})$. We can, however, further improve the performance by optimizing over r. To measure the performance of an allocation $\mathbf{k}$ compared to the optimal one, we define the regret of an (N,r)-allocation $\mathbf{k}$ as

$$\gamma(\mathbf{k}) := \eta_r(\mathbf{k}) - \min_{r\in[N]} \eta_r(\mathbf{k}_{\text{opt}}^{(r)}). \qquad (4)$$

In Fig. 4, we consider three heterogeneous paths with exponential delays. We find the optimal allocation by minimizing the regret. Interestingly, the optimal allocation is neither a $(3,1)$ replication, nor a $(3,3)$ allocation, but rather a $(3,2)$-allocation.

IV. Adaptive Collaborative Stream Uploading

Now, we analyze collaborative uploading for continuous data streams using an FJ queuing model. An example scenario is the continous upload of video data using multiple paths, as depicted in Fig. 1. We first consider a rigid allocation strategy based on known probabilistic bounds on the steady-state waiting times before proposing an adaptive allocation scheme based on stochastic gradient descent.

A. *Rigid allocation based on steady-state bounds*

Following [1], we define the waiting time of an incoming data batch as the amount of time it waits until the last of its chunks starts getting uploaded. Consider the steady-state waiting time W (precise definition given in [1] and for the sake of completeness, also in [4]). It is hard to find out the distribution of the steady-state waiting times in closed form (see [1], [9]). One approach is to compute tight upper bounds on the tail probabilities. Following [2], for a given allocation $\mathbf{k} \in \Lambda(N,K)$ and independent service times, we get

$$\mathrm{P}(W \ge \sigma) \le \exp(-\tilde{\theta}\sigma) \sum_{i\in[N]} \exp\big(-(\theta_i - \tilde{\theta})\sigma\big), \qquad (5)$$

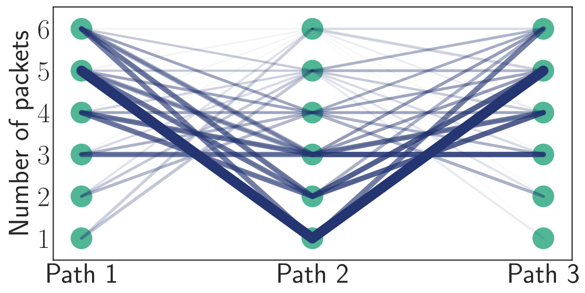

Fig. 4: **Optimal allocation by minimizing regret:** The lines specify different (3, 2)-allocations. For example, the line joining 1, 5, and 6 corresponds to the allocation (1, 5, 6). Valid (3, 2)-allocations require the combined size of any 2 chunks to be at least the data size, here, 6. The darkness/thickness of the shades is inversely proportional to the regrets defined in (4). The allocation (5, 1, 5) (the thickest line), achieves zero regret and hence, is the optimal one. We assume exponential delays with rates 1, 5 and 10 in order of increasing path indices.

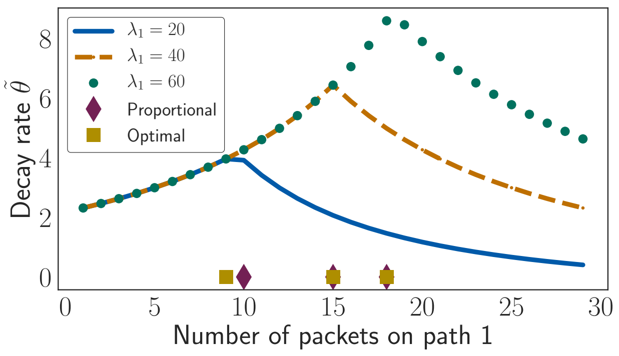

Fig. 5: **Canonical two-path scenario for collaborative stream uploading:** The effective decay rate $\tilde{\theta}$ from (5) as a function of the number of packets sent via path 1 out of overall 30 packets. Both paths have exponential delays. We vary the rate of the first path λ_1 as shown in the legend and keep the rate of the second path fixed at $\lambda_2 = 40$. The inter-arrival times are exponentially distributed with rate 0.5. Here too, we observe near-optimality of the proportional allocation.

where $\theta_i > 0$ is given by a condition involving the Laplace transforms of the inter-arrival times and the service times for k_i packets and $\tilde{\theta} := \min_{i \in [N]} \theta_i$. Here, $\tilde{\theta}$ is the effective decay rate of the tail probability in the sense of large deviations principle, and assesses the quality of a given allocation (the higher the decay rate, the better).

Reducing the waiting times is equivalent to maximizing the effective decay rate. Treating $\tilde{\theta}$ as a function of the allocation, the optimal allocation is given by

$$\mathbf{k}_{\text{opt}} := \underset{\mathbf{k} \in \Lambda(N,K)}{\arg\min}\ \tilde{\theta}(\mathbf{k}). \tag{6}$$

In Fig. 5, we revisit the canonical two-path scenario with exponential delays (derivation in [4, Example 3]). Plotting the effective decay rate $\tilde{\theta}$ as a function of the number of packets on path 1, we find the optimal allocation (yielding the largest decay rate). We also observe the near-optimality of the proportional allocation.

The approach in (6) is convenient because of its simplicity. However, it has a number of drawbacks. Apart from the exponentially growing search space for the optimal allocation, the approach is valid for the steady-state waiting time only. In many applications, the transient behavior is important. The approach in (6) does not allow for adaptation as it ignores the current state of the system (the number of chunks already on each path). In a realistic setup with changing environment (*e.g.*, Markov-modulated paths' services), the ability to adapt is crucial. Keeping this in mind, we propose an adaptive allocation scheme in the next section.

B. Adaptive Allocation

We consider the problem of sequentially optimizing allocations for collaborative uploading of incoming data batches. The procedure is sketched in Fig. 6. Our adaptive allocation strategy seeks to minimize a sequence of cost functions $(c_j)_{j \in \mathbb{N}}$ by choosing a sequence of optimal allocation vectors $(\mathbf{k}_j)_{j \in \mathbb{N}}$, with $\mathbf{k}_j := (k_{1,j}, \ldots, k_{N,j})$. An allocation vector $\mathbf{k}_j$ is a vector of integers, where the n-th entry corresponds to the number of packets (chunk size) transported over path n and $\|\mathbf{k}_j\|_1 = K_j$ is the size of the j-th data batch. Since optimization over integers is hard, we adopt the following standard relaxation: instead of an allocation vector, we optimize a proportion vector $\mathbf{x}_j := (x_{1,j}, \ldots, x_{N,j}) \in C$ corresponding to the j-th data batch, where $C := \{\mathbf{y} \in [0,1]^N : \|y\|_1 = 1\}$ is the set of valid proportions. The allocation vector $\mathbf{k}_j$ is found by taking the floor of first $N-1$ entries of $\mathbf{x}_j K_j$ and subtracting their sum from K_j to get the N-entry such that $\mathbf{k}_j \in \Lambda(N, K_j)$ (denote it as $\mathbf{k}_j := \langle \mathbf{x}_j K_j \rangle_{\Lambda(N,K_j)}$). Adhering to our modeling approach in Sec. II, we choose the mean waiting time as our cost function.

Denote the service times of data batch j on path n and the inter-arrival times between data batch j and $j+1$ by $S_{n,j}$ and t_j, respectively. From a control theoretic perspective, we treat $\mathbf{x}_{j-1}$ as our control variable to optimize the mean of the following output W_j

$$W_j := \max\{0, \max_{n \in [N], k \in [j-1]} \{\sum_{i \in [k]} (x_{n,j-i} S_{n,j-i} - t_{j-i})\}\}. \tag{7}$$

The quantity in (7) mimics the waiting time of j-th data batch in a Fork-Join system [1], [2] with a diminishing rounding error for increasing batch sizes. The j-th cost function is

$$c_j(\mathbf{x}_j) := \mathrm{E}\left[W_{j+1}\right], \tag{8}$$

We minimize the cost functions sequentially in an online fashion using gradient descent methods as they achieve a bounded regret in an online convex programming scenario [10].

As data batches are passed from the application on the primary device to be split over multiple secondary devices (Fig. 1), we assume the j-th inter-arrival time t_j is known to the scheduler before employing the next proportion $\mathbf{x}_{j+1}$, given $\{\mathbf{x}_i\}_{i\leq j}$. Using Monte Carlo (MC) methods we can calculate an unbiased estimate of the cost function for each data batch j. Since (7) is a piecewise linear function, which is non-smooth, we use an unbiased estimate of a subgradient $\hat{\mathbf{g}}_j$ of the j-th cost function c_j in (8), to perform gradient descent. The definition of a subgradient is given in [4] and [11].

Gradient descent for data allocation: As shown in Fig. 6, we update the proportion $\mathbf{x}_j$ using gradient descent by

$$\mathbf{x}_{j+1} = \mathsf{P}_C\big(\mathbf{x}_j + \eta\hat{\mathbf{g}}_j(\mathbf{x}_j)\big) \tag{9}$$

where $\hat{\mathbf{g}}_j(\mathbf{x}_j)$ is an unbiased estimate of the subgradient of c_j in (8) evaluated at the current $\mathbf{x}_j$. Here, η is the static learning rate controlling the step size of the subgradient, and $\mathsf{P}_C(.)$ denotes the Euclidean projection operator [12], projecting the gradient update onto the set of feasible proportions C.

The update equation (9) ensures bounded regret with an unbiased estimate of the subgradient $\hat{\mathbf{g}}_j$ [13], which we obtain using Monte Carlo methods. We resample service times M times to estimate the subgradient. The m-th sample for the service times up to data batch j at each path is denoted by $\{s^{(m)}_{n,i}\}_{n\in[N],i\leq j}$. Then, the MC estimate of the subgradient is

$$\begin{aligned}\hat{\mathbf{g}}_j(\mathbf{x}_j) = \frac{1}{M}\sum_{m\in[M]} \Big(s^{(m)}_{n,j}\mathbf{e}_n \mathbb{1}\big(s^{(m)}_{n^*_m,k^*_m}\mathbf{e}^{\mathsf{T}}_{n^*_m}\mathbf{x}_j \\ + \sum_{i=1}^{k^*_m-1} x_{n^*_m,j-i}s^{(m)}_{n^*_m,j-i} - \sum_{i=0}^{k^*_m-1} t_{j-i} > 0\big)\Big),\end{aligned} \tag{10}$$

where $\mathbf{e}_n$ is the n-th unit vector, and $\mathbb{1}(.)$ is the indicator function. The maximizers n^*_m and k^*_m can be found by

$$(n^*_m, k^*_m) = \underset{\substack{n\in[N],\\ k\in[j]}}{\arg\max}\left\{ s^{(m)}_{n,j}\mathbf{e}^{\mathsf{T}}_n + \sum_{i=1}^{k-1} x_{n,j-i}s^{(m)}_{n,j-i} - \sum_{i=0}^{k-1} t_{j-i} \right\}. \tag{11}$$

Algorithm 1 describes the adaptive allocation method. In the next section, we describe the inference procedure required for our adaptive allocation.

Algorithm 1 Adaptive Allocation

```
1: $j = 1$, $\mathbf{x}_1 \leftarrow \frac{1}{N}$ and $\mathbf{k}_1 \leftarrow \langle \mathbf{x}_1 K_1\rangle_{\Lambda(N,K_1)}$ //Initialization
2: Draw M samples of $S_{n,j}$ at each batch arrival
3: for all $m \in [M]$ do
4:     Calculate $(n^*_m, k^*_m)$ using (11)
5: end for
6: Estimate the subgradient $\hat{\mathbf{g}}_j(\mathbf{x}_j)$ using (10)
7: $\mathbf{x}_{j+1} \leftarrow \mathsf{P}_C\big(\mathbf{x}_j + \eta\hat{\mathbf{g}}_j(\mathbf{x}_j)\big)$, $\mathbf{k}_{j+1} \leftarrow \langle \mathbf{x}_{j+1}K_{j+1}\rangle_{\Lambda(N,K_{j+1})}$
8: $j \leftarrow j + 1$
```

C. Inference for service time processes

Since the adaptation of allocations requires samples of the service times (Step 2 of Algorithm 1), we provide here illustrative resampling schemes for independent and identically distributed (i.i.d.) and Markov-modulated service times.

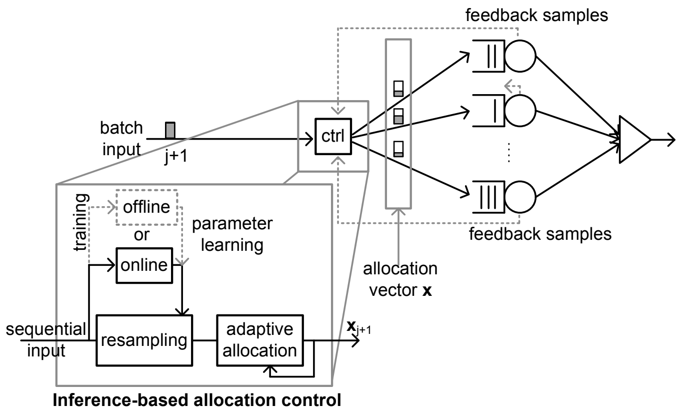

Fig. 6: At the arrival of each data batch, the controller first computes an MC estimate of the subgradient using (10) before finding the next allocation vector using (9). In order to get the MC estimate, we resample the service times M times. If unknown, we infer the latent state of the Markov chains that describe the paths' service in an online fashion, while other model parameters are learned offline using training samples. Observed service times serve as feedback to the controller.

Exponential i.i.d. service times: Assume i.i.d service times at the different paths. Suppose the service times at one of the paths are exponentially distributed with rate parameter λ so that the distribution of $S_{n,j}$ is gamma with shape parameter K_j, and rate λ. The predictive distribution of a new sample of service time $S^*_{n,j}$ based on a set of observed data $\{s'_{n,i}\}_{i\leq j}$ is found by integrating out the parameter λ, *i.e.*, by $\int p(S^*_{n,j}|K_j,\lambda)p(\lambda|\{s'_{n,i}\}_{i\leq j})\mathrm{d}\lambda$. If we assume a conjugate prior $p(\lambda)$ on λ, namely, a gamma distribution with (hyper)-parameters K_0 and λ_0, the posterior $p(\lambda|\{s'_{n,i}\}_{i\leq j})$ is a gamma distribution with (posterior) parameters $K_{\mathrm{p}} = K_0 + \sum_{i=1}^{j} K_{n,i}$ and $\lambda_{\mathrm{p}} = \lambda_0 + \sum_{i=1}^{j} s'_{n,i}$. The derivation steps are provided in [4]. In order to sample from the predictive distribution, we first sample λ from the posterior distribution and then sample from $p(S^*_{n,i}|K_i,\lambda)$ conditioned on λ (step 2 in Algorithm 1). Repeating this procedure iteratively, we get the following update equations for the posterior parameters $K^{(i)}_{\mathrm{p}} = K^{(i-1)}_{\mathrm{p}} + K_{n,i}$, $\lambda^{(i)}_{\mathrm{p}} = \lambda^{(j-1)}_{\mathrm{p}} + s'_{n,i}$, with $K^{(0)}_{\mathrm{p}} = K_0$ and $\lambda^{(0)}_{\mathrm{p}} = \lambda_0$.

Markov modulated service times: In this case, we assume that the service times are instances of a Markov modulated exponentially distributed variable. For a sequence of service times, we first find maximum-likelihood estimates (MLE) of the underlying parameters, *e.g.*, the initial distribution, transition matrix and the rates. Since the MLE computation is expensive, this estimation process is executed offline. The derivation of the MLE is provided in [4]. Next, we condition on these parameters to calculate the *maximum a posteriori* (MAP) estimate of the current hidden state of the Markov chain. We do this online using the Viterbi algorithm [14]. Having inferred the hidden state, we resample the service times (step 2 in Algorithm 1) online by conditioning on the hidden state and the parameters estimated in the first step.

V. Numerical Evaluation

We consider the setup in Fig. 1 with arrivals at the primary device being modeled as a Markov modulated Poisson process (MMPP) to allow for burstiness in batch arrivals. This model was shown to be a good candidate for some network traffic [15]. We assume inter-arrival times are modulated by a three-state Markov chain. Further, the batch sizes are sampled from a Poisson distribution with mean 10^2 to account for varying data sizes to be uploaded. We assume five heterogeneous paths are available. For the service process, we observe just one sample of the service times for each path and each data batch.

We evaluate the performance of our adaptive allocation using two experiments. In the first experiment, we vary the service time distributions to reflect different regimes of stress on the paths, assuming full knowledge of the model parameters. In the second experiment, we do not assume knowledge of the model parameters, but instead infer them. The MC estimate of the subgradient is always based on 10^2 samples, except for the one sample estimate (OSE) method where only the observed sample is used. We use 10^4 and $5 \cdot 10^3$ simulations for the first and the second experiment, respectively.

We compare our adaptive allocation method with the proportional allocation because of its observed near-optimality in the intermittent case and in the bound approach for continuous stream uploading. Note that finding the optimal allocation is computationally expensive, while the proportional allocation is readily found. We also consider a variant of a queue-aware "join the shortest queue" (JSQ) schedule where the batch is assigned to the path with the shortest queue. Such schedules are known to have good performance. However, for many applications, obtaining the queue length information is hard.

Experiment 1: Here, we assume Markov modulated exponentially distributed service times. This captures the service time correlations, *e.g.*, in time varying wireless channels [16]. We use five independent three-state Markov chains to modulate the means of the service times for each chunk on each path. For the proportional allocation, we calculate the mean service rate weighted by the stationary probabilities.

We consider two complimentary situations: one, called the *low stress* regime, where the service rates are high such that one path is sufficient to serve incoming batches (ensuring stability), and the other, called the *high stress* regime, where the service rates are low that utilizing all the five paths is *necessary* to ensure stability. In Fig. 7, we show the complementary cumulative distribution function (CCDF) of the waiting times comparing different allocation strategies. This figure shows the benefit of adaptive allocation. Note that the proportional allocation is not adaptive. On the other hand, the batch JSQ under-utilizes parallelization. Our allocation scheme described in Sec. IV-B ensures adaptiveness while being allocative. This benefit of adaptation is prominent, especially in the high stress regime as seen in Fig. 7 (Middle). Fig. 7 (Right) also shows how our allocation adapts to the service rate changes. Here, the latent state of the Markov chain and the parameters for the service times are assumed to be known.

Experiment 2: In this experiment, we infer the model parameters. In keeping with the setup in Fig. 6, we first assume i.i.d. exponentially distributed service times. Fig. 8 (Left) shows the CCDFs of the waiting times for different allocation strategies. To estimate the subgradient, the oracle uses the true parameters to draw samples, while the Bayesian inference draws samples from the predictive distribution.

We further consider the setup in Fig. 6 with Markov modulated exponentially distributed service times as described in *Experiment 1*. The inference method draws samples from the emission distribution using a MAP estimate of the current latent state of the Markov chain for each allocation. The parameters of the Markov modulation of the service times are learned offline using a training sequence. In Fig. 8 (Right), we compare our inference-based method with the OSE, and an oracle that draws samples from the emission distribution with the true parameters and the true latent states.

From Fig. 8, we see that increasing the number of samples for the subgradient estimation leads to smaller tail probabilities, since the subgradient noise decreases. Interestingly, the lack of knowledge of the model parameters does not affect the performance of our adaptive allocation, as the inference method achieves results comparable to that of the oracle.

VI. Related Work

The work in [17] anticipated the emergence of crowdsourcing systems. A recent discussion introduced crowdsourced live event coverage in [18], where the authors propose adaptive strategies for collaboratively uploading the most relevant streams. Our analytical treatment is complimentary to [18].

The intermittent uploading scenario is analyzed in [5], where the authors provide an upper bound on the mean delay for the canonical two-path scenario with exponential path delays. In contrast, we obtain closed-form expressions for the mean delay for $N \geq 2$ paths. We also provide tools and examples of the optimization thereof.

A segment of related work is concerned with the analysis of controlled and uncontrolled Fork-Join systems. For uncontrolled, *i.e.*, non-adaptive FJ systems, it is known that exact results are hard to obtain [9]. Exact results are known for the joint workload distribution for only two parallel queues with Poisson arrivals and i.i.d exponential service times [19]. For more general scenarios we resort, *e.g.*, to bounds on the tail probabilities of the steady-state waiting times for single-stage systems [1], [2], [9] or multistage systems [20]. They also highlight the benefits of parallelization under high utilization regimes, in agreement with what we observe in *Experiment 1* in Sec. V. The work in [21] considers controlling FJ systems using a gradient descent approach to minimize queue lengths. They use changes in queue lengths to find an unbiased estimate of the gradient. Note that obtaining the queue lengths is not straightforward in many applications. Therefore we infer the service time distributions to adapt our allocation using gradient descent to minimize the expected waiting time. In [22], the authors investigate scheduling of batch jobs in systems with multiple servers. As opposed to our setup, they assume a

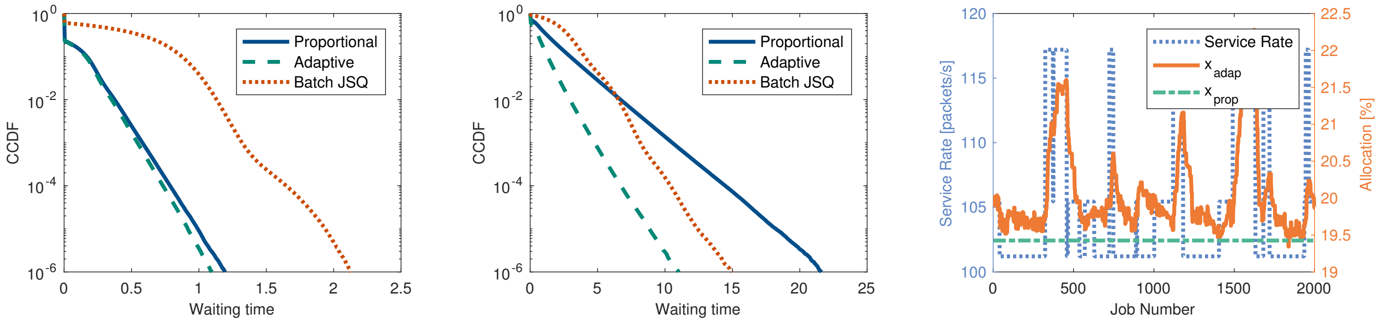

Fig. 7: **Waiting times for *Experiment 1*:** We compare a low stress regime **(Left)** and a high stress regime **(Middle)**. In both regimes, our adaptive allocation outperforms proportional allocation and batch JSQ. Notably the rigidity of the proportional allocation causes large waiting times in the high stress regime. **(Right)** Allocation adaptation on path 1. The dotted line shows the service rate, while the dashed line shows the proportional allocation and the solid line shows the adaptation.

queuing model without synchronization constraint and only consider a special class of i.i.d. distributed service times. For our adaptive allocation method, we do not make any independence assumption.

Redundancy techniques have grown in popularity over the years as a means to decrease latency. In [7], the authors study the trade-off between the latency reduction attained by redundancy and the corresponding overhead. Based on empirical results, they argue that redundancy can be effective in a large class of applications. The work in [8] is close to ours. The authors model a cloud computing scenario as a Fork-Join system with identical servers and analyze different redundancy techniques, which are akin to our (N, r)-allocations in the intermittent uploading case, with a view to reducing latency in a cost-efficient manner. The authors find that the log-concavity of the task service times decides the success of redundancy techniques. Their approach is complimentary to our adaptive allocation with heterogeneous Markov modulated servers.

The allocation problem in the continuous stream uploading case can be seen as a type of load-balancing problem, however, with the mean waiting time as the objective function. A programming model for the allocation of continous streams over multiple paths was introduced in [23]. In a recent work [24], the authors consider storage and delivery of large files in data-centers, where files are first erasure-coded and then

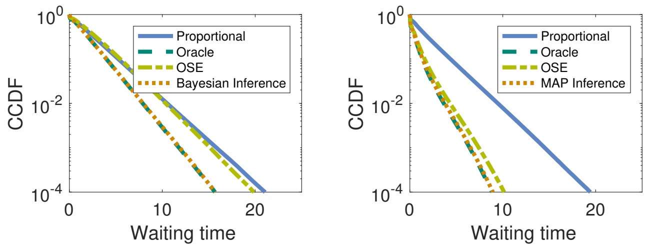

Fig. 8: Waiting time distributions for *Experiment 2*. We consider i.i.d **(Left)** and Markov modulated **(Right)** exponentially distributed service times. Evidently the lack of knowledge of the model parameters can be overcome by means of inference, which performs almost as good as the oracle method.

stored in a subset of the available servers. They compare the performance of water-filling and batch sampling as dynamic load-balancing policies and provide computable performance bounds. In contrast, we do not restrict ourselves to a rigid allocation strategy and adapt our allocation dynamically.

VII. Conclusion

In this work, we optimize allocation and replication strategies in collaborative uploading scenarios. We differentiate between intermittent and continuous stream uploading, based on the system's queuing behavior. In the first case (no queuing), we unify the notions of allocation and replication, and provide closed-form expressions for the mean upload latency. We use our exact formulation for the intermittent uploading case to derive optimal allocation and replication strategies.

We pose the continuous stream uploading case as a Fork-Join queuing model with varying burstiness of the data traffic to be uploaded, and of the paths' service. Thereby we propose an adaptive allocation scheme, based on statistical inference of the properties of the paths' latencies. We sequentially minimize a notion of the expected waiting time, ensuring a bounded regret. We show the effectiveness of our adaptive approach compared to proportional allocation and batch JSQ allocation. The lack of knowledge of the model parameters does not affect the performance of our adaptive allocation, as the inference methods are able to achieve results comparable to those of an oracle with full system knowledge.

Appendix I

A. *Moments of order statistics*

Let $X_1, X_2, \dots, X_N$ be independent positive-valued random variables with *absolutely continuous* CDFs $F_1, F_2, \dots, F_N$. Let the corresponding order statistics be $Y_1 \leq Y_2 \leq \dots \leq Y_N$. Write $\mathbf{F} := (F_1, F_2, \dots, F_N)^\intercal$ and $\mathbf{1}-\mathbf{F} := (1-F_1, 1-F_2, \dots, 1-F_N)^\intercal$. The distribution of the r-th order statistic can be elegantly written in terms of certain permanents as [25, Theorem 4.1],

$$\mathrm{P}(Y_r \leq y) = \sum_{i=r}^{N} \frac{1}{i!(N-i)!} \operatorname{per} \left[\underset{i}{\mathbf{F}(y)} \; \underset{N-i}{\mathbf{1} - \mathbf{F}(y)} \right], \tag{12}$$

where $\left[\frac{\mathbf{F}(y)}{i}\ \frac{\mathbf{1}-\mathbf{F}(y)}{N-i}\right]$ denotes the matrix whose first i columns are $\mathbf{F}(y)$ and the last $N-i$ columns are $\mathbf{1}-\mathbf{F}(y)$, per $A := \sum_{\sigma\in\Theta(N)}\prod_{i=1}^{N} a_{i,\sigma(i)}$ denotes the permanent of an $N\times N$ real matrix $A = ((a_{i,j}))_{i,j\in[N]}$, and $\Theta(N)$ denote the class of all permutations of $[N]$. Using (12), we derive the expected values of the order statistics [25], [26] (proof given in [4]).

Remark 1. For $r \in [N]$, the mean of Y_r can be conveniently written in terms of μ-operators given by

$$\mathrm{E}\left[Y_r\right] = \mu_r\,\mathbf{F} := \sum_{j=N-r+1}^{N} (-1)^{j-(N-r-1)}\binom{j-1}{N-r}\mathrm{M}_j\,\mathbf{F},$$

where the M_j-operators, for $j\in[N]$, are defined as

$$\mathrm{M}_j\,\mathbf{F} := \sum_{S\in\{A\subseteq[N]:|A|=j\}}\int_0^\infty \Big(\prod_{i\in S}(1-F_i(x))\Big)\,\mathrm{d}x\,. \qquad (13)$$

B. General (N,r)-strategy

Example 1 (Example of (N,r)-strategy)**.** Suppose we have three paths with exponential delays with parameters λ_1,λ_2 and λ_3. Define, for $i=1,2,3$, $p_{ij}=\frac{\lambda_i}{\lambda_i+\lambda_j}$, $q_{ij}=1-p_{i,j}$, $r_{ij}=\frac{1}{\lambda_i+\lambda_j}$ and $p_{123}^{(i)}=\frac{\lambda_i}{\lambda_1+\lambda_2+\lambda_3}$, $r_{123}=\frac{1}{\lambda_1+\lambda_2+\lambda_3}$. The mean upload latency corresponding to a $(3,1)$-allocation (replication) $\mathbf{k}=(k_1,k_2,k_3)\in\Upsilon(3,1,K)$ is $\mu_1\,\mathbf{F}^{(\mathbf{k})}=\eta_1(\mathbf{k})$, and is given by

$$r_{123}\sum_{n_1=0}^{k_1-1}\sum_{n_2=0}^{k_2-1}\sum_{n_3=0}^{k_3-1}\frac{(n_1+n_2+n_3)!}{n_1!n_2!n_3!}\left(p_{123}^{(1)}\right)^{n_1}\left(p_{123}^{(2)}\right)^{n_2}\left(p_{123}^{(3)}\right)^{n_3}.$$

For a $(3,2)$-allocation $\mathbf{k}\in\Upsilon(3,2,K)$, the mean upload latency, $\mu_2\,\mathbf{F}^{(\mathbf{k})}=\eta_2(\mathbf{k})=\mathrm{M}_2\,\mathbf{F}^{(\mathbf{k})}-2\mathrm{M}_3\,\mathbf{F}^{(\mathbf{k})}$, is given by

$$r_{12}\sum_{n_1=0}^{k_1-1}\sum_{n_2=0}^{k_2-1}\frac{(n_1+n_2)!}{n_1!n_2!}p_{12}^{n_1}q_{12}^{n_2}+r_{23}\sum_{n_2=0}^{k_2-1}\sum_{n_3=0}^{k_3-1}\frac{(n_2+n_3)!}{n_2!n_3!}p_{23}^{n_2}q_{23}^{n_3}$$
$$+\,r_{31}\sum_{n_3=0}^{k_3-1}\sum_{n_1=0}^{k_1-1}\frac{(n_3+n_1)!}{n_3!n_1!}p_{31}^{n_3}q_{31}^{n_1}-2\eta_1(\mathbf{k}).$$

Note that a $(3,3)$-allocation corresponds to simple allocation (see Sec. III-B and [4]). The derivations are provided in [4].

Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of projects C3 and B4 within the Collaborative Research Center (CRC) 1053 – MAKI. Computational facilities provided by the Lichtenberg - High Performance Computer at TU Darmstadt are gratefully acknowledged.

References

[1] A. Rizk, F. Poloczek, and F. Ciucu, “Computable bounds in fork-join queueing systems,” in *Proc. ACM Sigmetrics Perform. Eval. Rev.*, Jun. 2015, pp. 335–346.

[2] W. R. KhudaBukhsh, A. Rizk, A. Frömmgen, and H. Koeppl, “Optimizing stochastic scheduling in fork-join queueing models: bounds and applications,” in *Proc. IEEE Int. Conf. Comput. Commun.*, May 2017.

[3] F. Baccelli, A. M. Makowski, and A. Shwartz, “The Fork-Join Queue and Related Systems with Synchronization Constraints: Stochastic Ordering and Computable Bounds,” *Adv. Appl. Probab.* , pp. 629–660, 1989.

[4] W. R. KhudaBukhsh, B. Alt, S. Kar, A. Rizk, and H. Koeppl. (2017, July) Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies. Extended version http://arxiv.org/abs/1712.04175. [Online]. Available: http://arxiv.org/abs/1712.04175

[5] G. Zhang, Y. Wen, J. Zhu, and Q. Chen, “On file delay minimization for content uploading to media cloud via collaborative wireless network,” in *Proc. Int. Conf. Wireless Commun. and Signal Process.*, Nov 2011, pp. 1–6.

[6] Y. Wen and J. Sun, “On minimum-delay data block transport over two-connected mesh networks,” in *Proc. IEEE Wireless Commun. and Networking Conf.*, Mar 2007, pp. 4034–4039.

[7] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, “Low latency via redundancy,” in *Proc. 9th ACM Conf. Emerging Networking Experiments and Technol.*, 2013, pp. 283–294.

[8] G. Joshi, E. Soljanin, and G. Wornell, “Efficient redundancy techniques for latency reduction in cloud systems,” *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, pp. 12:1–12:30, May 2017.

[9] F. Baccelli, A. M. Makowski, and A. Shwartz, “The fork-join queue and related systems with synchronization constraints: stochastic ordering and computable bounds,” *Adv. Appl. Probab.*, vol. 21, pp. 629–660, Sep 1989.

[10] M. Zinkevich, “Online convex programming and generalized infinitesimal gradient ascent,” in *Proc. 20th Int. Conf. Mach. Learning*, 2003, pp. 928–936.

[11] B. T. Poljak, *Introduction to optimization*. Optimization Software, 1987.

[12] W. Wang and M. Á. Carreira-Perpiñán, “Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application,” *arXiv:1309.1541 [cs.LG]*, Sep 2013.

[13] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, “Online convex optimization in the bandit setting: gradient descent without a gradient,” in *Proc. 16th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan 2005, pp. 385–394.

[14] Y. Ephraim and N. Merhav, “Hidden Markov processes,” *IEEE Trans. Inf. Theory*, vol. 48, pp. 1518–1569, Jun 2002.

[15] A. Feldmann and W. Whitt, “Fitting mixtures of exponentials to long-tail distributions to analyze network performance models,” *Perform. Evaluation*, vol. 31, no. 3-4, pp. 245–279, 1998.

[16] K. Mahmood, A. Rizk, and Y. Jiang, “On the flow-level delay of a spatial multiplexing MIMO wireless channel,” in *Proc. IEEE Int. Conf. Commun.*, 2011, pp. 1–6.

[17] J. Howe, “The rise of crowdsourcing,” *Wired Mag.*, vol. 14, no. 6, pp. 1–4, 2006.

[18] B. Richerzhagen, J. Wulfheide, H. Koeppl, A. Mauthe, K. Nahrstedt, and R. Steinmetz, “Enabling crowdsourced live event coverage with adaptive collaborative upload strategies,” in *Proc. IEEE 17th Int. Symp. A World of Wireless, Mobile and Multimedia Networks*, June 2016, pp. 1–3.

[19] L. Flatto and S. Hahn, “Two Parallel Queues Created by Arrivals with Two Demands I,” *SIAM J. Appl. Math.*, vol. 44, no. 5, pp. 1041–1053, 1984.

[20] M. Fidler and Y. Jiang, “Non-Asymptotic Delay Bounds for (k, l) Fork-Join Systems and Multi-Stage Fork-Join Networks,” in *Proc. IEEE Int. Conf. Comput. Commun.*, April 2016, pp. 1–9.

[21] R. Pedarsani, J. Walrand, and Y. Zhong, “Robust scheduling in a flexible fork-join network,” in *Proc. IEEE 53rd Annu. Conf. Decision and Control*, 2014, pp. 3669–3676.

[22] Y. Sun, C. E. Koksal, and N. B. Shroff, “Near delay-optimal scheduling of batch jobs in multi-server systems,” Ohio State Univ., Tech. Rep., 2017.

[23] A. Frömmgen, A. Rizk, T. E. M. Weller, B. Koldehofe, A. Buchmann, and R. Steinmetz, “A Programming Model for Application-defined Multipath TCP Scheduling,” in *Proc. 18th ACM/IFIP/USENIX Middleware Conf.*, 2017, pp. 134–146.

[24] V. Shah, A. Bouillard, and F. Baccelli, “Delay comparison of delivery and coding policies in data clusters,” *arXiv:1706.02384 [cs.PF]*, Jun 2017.

[25] R. B. Bapat and M. I. Beg, “Order Statistics for Nonidentically Distributed Variables and Permanents,” *Sankhya Ser A*, vol. 51, no. 1, pp. 79–93, 1989.

[26] H. M. Barakat and Y. H. Abdelkader, “Computing the moments of order statistics from nonidentical random variables,” *Stat. Method. and Appl.*, vol. 13, no. 1, pp. 15–26, 2004.