Reliability, timeliness and load reduction at the edge for cloud gaming

Franco, Antonio; Fitzgerald, Emma; Landfeldt, Björn; Körner, Ulf

Published in:
International Performance Computing and Communications Conference

DOI:
10.1109/IPCCC47392.2019.8958728

2020

Reliability, timeliness and load reduction at the edge for cloud gaming

Antonio Franco*, Emma Fitzgerald*†, Björn Landfeldt*, Ulf Körner*
†Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
*Department of Electrical and Information Technology, Lund University, Lund, Sweden
{antonio.franco, emma.fitzgerald, bjorn.landfeldt, ulf.korner}@eit.lth.se

***Abstract*—In this paper we study reliability, timeliness and load reduction in an hybrid Mobile Edge Computing (MEC)/Cloud game streaming infrastructure. In our scenario, a user plays a game streamed by the producer to their handheld device – or User Equipment (UE). The UE communicates user actions to the edge/cloud via the mobile communication infrastructure; the object is to retrieve the latest game status, of which the most important information is the rendered frame. Particularly, we study reliability through replication in a number of MEC-servers, timeliness through Age of Information (AoI) and load reduction by leveraging the X2 interface at the edge, in order to abort useless frame rendering computations. We translate it as a scenario where a sink – representing the UE – is interested in the freshest possible update from distributed nodes. Each node sends updates following a Last Come First Served (LCFS) policy with preemption. We consider two scenarios; the first is n parallel LCFS systems sending updates, and the second adds a feedback loop aimed at decreasing the number of jobs sent per second by the nodes, thus decreasing the load per node. We analyze the number of jobs sent per second and average peak Age of Information at the sink, showing that the second scheme achieves a significantly lower rate of jobs compared with the first, while maintaining constant AoI, thus reducing the load at the edge. We also find that using the feedback loop, we achieve the maximum saving in transmitted jobs per second when the average arrival rate per system is equal to the inverse of the average busy time in every node.**

***Index Terms*—Age of information; Queuing theory; Cloud gaming; Edge computing.**

I. Introduction

A gaming industry trend is the shift from the traditional paradigm of locally installed software rendered directly on the device hardware, to the streaming cloud gaming paradigm [1]. In this context, the game is streamed to the device, thus allowing it to run in a platform independent manner. Processing is offloaded to the cloud/edge, leveraging greater computational power than that of the single device. Also, the shift is pushed along since it helps the game producers to control piracy [2].

On the other hand, video game producers have to provide the necessary cloud infrastructure to accommodate the increased computational needs and increased bandwidth, which brings high costs that may be difficult to cover fully [3]. Also, latency issues must be addressed, since a user is going to experience poor Quality of Experience when delays are larger than 100 ms from the moment of interaction with the game to the rendered frame being displayed on their device [4]. In this context, edge computing could be a solution to the aforementioned issues, having lower latency, bringing the computational core closer to the user, and being easy to deploy.

The rise of Massively Multiplayer Online Game (MMOG) demand added strain to the infrastructure, being affected not only by the single user interaction, but also by other users playing the same scenario, as opponents or allies. A user is usually interested more in the *current* state of the game, being it the scenario in which their character interacts, determining score, victory or other rewards.

A suitable metric for the "timeliness" of updates is the Age of Information (AoI) [5]. In the AoI view, the 'freshness' of data is considered, which may not follow network delay directly. In a sense, the cross layer nature of AoI, stemming from it being a characteristic of the end-to-end information flow, represents a broader view of information freshness than delay does. To illustrate the difference between network delay and AoI, consider a scenario with sensor data traversing a single First Come First Served (FCFS) queueing system. If the sensor data generation rate increases, the queueing delay increases which in turn increases the delay through the system. Decreasing the data generation rate leads to shorter delay through the system, but at the same time, the time between samples increase, which leads to larger AoI.

AoI has been considered in the contest of cloud gaming. In [6] the authors study a system comprised of a device sending user commands to a remote game server via the network (e.g. in the cloud or at the edge). Then the command is passed through a buffer to the game server that processes it and creates the appropriate responses. Afterwards it goes to a buffer that ends in a frame renderer, that finally renders the frame and sends it to the device via the network. They use a preemption with a time threshold τ (defined by the system designers), in which a frame is considered obsolete and replaced by a new frame if its age is greater than a function of τ. They study the system and find a model describing the average AoI at the user end.

On the other hand we consider an hybrid Mobile Edge Computing (MEC) scenario/cloud based scenario (Fig. 1), with replicated renderings being computed in parallel, for reliability purposes when users are mobile. As the authors of [6] observe, user commands are usually small in size and it is very unlikely that a human will send more than 100 updates per second to

978-1-7281-1025-7/19/$31.00

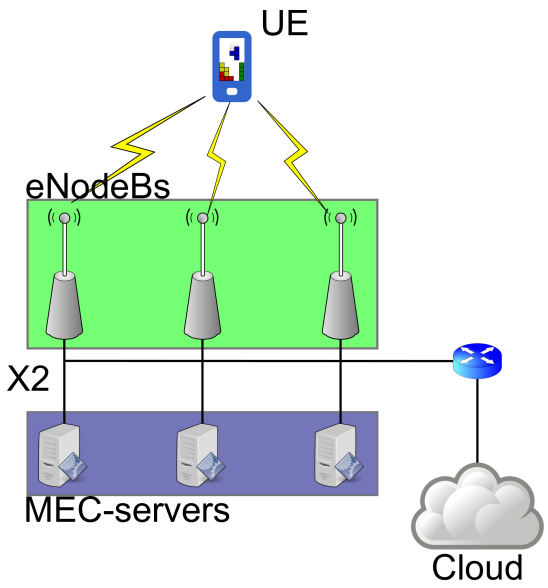

Fig. 1. Mobile edge computing scenario.

the server. We then concentrate on the more computationally intense part of frame rendering. We consider a user playing a MMOG on a mobile device – i.e. their User Equipment (UE). As the UE is moving, it could be within range of one or more Base Stations (BSs), or eNodeBs in the LTE ecosystem. eNodeBs are all interconnected via the X2 interface (or an equivalent reliable link), in order to exchange data. Also the interface is directly connected to a series of MEC-servers able to perform calculations. A connection to the cloud is also present, and can be considered as an additional MEC-server for our purposes.

Since the user is interested in the latest status of the game – that is, our piece of information – we want to send them the freshest update. We then allow our MEC-servers to preempt, i.e. substitute a staler task from the same user with a fresher one. We can consider them as Last Come First Served (LCFS) queueing systems with preemption. Also, for reliability purposes, the calculation is offloaded to n MEC-servers in parallel. MEC-servers could be loaded with a number of tasks (jobs) that we can consider independent from the calculation requested by our UE; we can then assume the service time – or, more correctly, busy time, as they are different in preemptive systems, see Section II – triggered by the calculation as being exponentially distributed.

Given that the job is replicated in different MEC-servers, upon the completion of a task by one of them, the staler jobs in execution in the other $n-1$ servers are aborted, since a staler frame refresh is useless to the end user; also we want to reduce the load on the MEC-servers, since computation is the most expensive part of this infrastructure, especially if leased from telecommunication/cloud resources providers and paid per clock. We achieve this objective by broadcasting the time stamp of the executed jobs to the other MEC-servers via the X2 interface, thus letting them decide whether to abort or continue the user task. Then the completed task is sent to the eNodeB closest to the UE for transmission.

In order to model this scenario, we decouple the source of commands and the receiver of the job (both being the UE in Fig. 1), having different sources for different LCFS M/M/1 systems with preemption, and a central sink collecting the pieces of information. The task substitution per MEC-server is modeled by the preemptive nature of the single queueing system. The task deletion on the other MEC-servers is modeled via a feedback loop from the sink to every MEC-server involved in the calculation, communicating the timestamp of the freshest piece of information received. Also, we consider exponentially distributed inter arrival times to account for network delays between the UE and the MEC-servers. A more detailed description of the model is in Section II.

LCFS systems with preemption have been studied, in terms of average AoI, for single queuing systems [7]–[9]. Due to the nature of our scenario, jobs arrive at the sink out of order; AoI in conjunction with preemptive disciplines for multi-class systems and jobs arriving out of order at the destination were studied in [10]–[13].

In general, this paper considers a case in which a central monitor (sink) is interested in the freshest possible update from some remote systems, either performing a calculation, or measuring a phenomenon. We have only a single information flow. Optimal policies for systems with n transmitters and n information flows sent through a shared resource were studied in [14], [15], while in [16], [17] there is a central system updating remote devices with different flows.

Specifically, we study two scenarios: first we find the number of jobs sent per second of several LCFS M/M/1 queuing systems with preemption in parallel, all sending updates from the same information stream to a sink. Then we add a feedback loop in order to decrease the number of jobs sent to the sink, while maintaining the same AoI at the receiver end. In the latter, a back channel is used by the sink to communicate directly with the queues. We find that by using the feedback loop we can achieve a saving in sent jobs close to 50 % for 20 parallel servers.

Summarizing, the main contributions of this paper are:

- we study a MEC/cloud hybrid game streaming scenario where rendering tasks are replicated in n MEC-servers for reliability purposes; we add a mechanism to reduce the overall load in the system; we model it with a series of LCFS M/M/1 systems with preemption sending updates to a central sink with a feedback loop in place;
- we consider AoI as a metric for timeliness, being users more interested in the current state of the game;
- we find the expression of the number of jobs sent by a number n of LCFS M/M/1 systems with preemption with or without a feedback loop from the sink and show the advantage in terms of saved job transmissions by using the feedback, while maintaining the AoI constant;
- we find that the maximum saving in transmitted jobs per second is achieved when the average arrival rate per system, is equal to the inverse of the average busy time

period per system (as we will see in Section II service time and busy time are different in preemptive systems);
- we also find the expression of the average peak Age of Information (pAoI) at the sink side.

The rest of this paper is subdivided as follows. In Section II the scenario is described in detail. In Section III an expression for the number of jobs sent per second for both scenarios is derived. In Section IV an expression for the average pAoI is derived. In Section V the previous expressions are tested against simulations and numerical results analyzed. Finally in Section VI conclusions and future work are discussed.

II. Model description

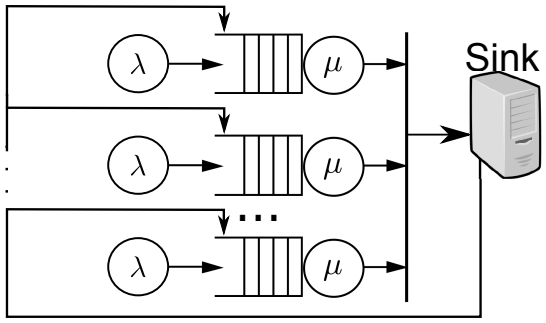

Fig. 2. Our scenario.

Our model is depicted in Fig. 2. Each source generates pieces of information with an average rate of λ jobs per second, modeling, for example, an exponentially distributed delay between the UE and the MEC-servers. The servers all serve jobs with an average rate of μ jobs per second, modeling the fact that the MEC-servers are loaded with tasks independent from our rendering. All the sources are about the same information/calculation. We assume the communication with the queues is instantaneous and without errors.

We will refer to the time generated when a job arrives to the server without finding any other job in service as **busy time**, while the time from the arrival of a job to the server and its departure from the system will be referred as **service time**. While in systems without preemption they are the same, in preemptive systems they are different, as we will see in the next paragraphs.

The sink is only interested in the freshest piece of information, hence it is interested in minimizing the AoI at its end. Both the inter generation times and the busy times follow an exponential distribution i.e. their respective Probability Density Functions (PDFs) are:

$$f_A(t) = \lambda \mathrm{e}^{-\lambda t} \mathrm{H}(t), \tag{1}$$

and

$$f_S(t) = \mu \mathrm{e}^{-\mu t} \mathrm{H}(t), \tag{2}$$

where $\mathrm{H}(t)$ is the Heaviside step function defined as:

$$\mathrm{H}(t) = \begin{cases} 1 & , t \geq 0 \\ 0 & , t < 0 \end{cases}.$$

It is also worth mentioning that for the remainder of the paper the PDF of a random variable X will be expressed as $f_X(x)$ and its Cumulative Density Function (CDF) as $F_X(x) = \Pr\{X \leq x\}$. We will also assume that, unless specified, all the random variables have non negative support.

Each queuing system employs preemption, in which each time a fresher piece of information is generated, the new job takes the place in the server of the previous one already in service. Note however that since we consider exponentially distributed busy times, the residual time has the same distribution as the busy time. The service time experienced by the new job will be the residual service time of the preempted job. The substituted job is discarded. Since we consider only one information stream, and each system has only one source, there are no jobs in the queue; they can only be in service.

Having exponentially distributed inter arrival times and busy times in a single LCFS M/M/1 system with preemption has the property that the service time for a successful job (i.e. a job that is not preempted) is also exponentially distributed, as we will prove in Lemma 1. This means that, when preempting the staler job in the MEC-server, the residual service time is distributed in the same manner for all successfully rendered tasks, irregardless of how many tasks have been preempted before.

Lemma 1. *In a single LCFS M/M/1 system with preemption, with average arrival rate λ jobs/s and average busy time μ^{-1} s, the service time of a successful job (i.e. is not preempted) T is also exponentially distributed with rate $\lambda + \mu$ jobs per second.*

Proof. We call A the random variable describing the interarrival times, that is exponentially distributed with rate λ jobs per second Eq. (1). On the other hand, S is the random variable describing the busy time, also exponentially distributed with rate μ jobs per second Eq. (2).

The effective service time will be the service time experienced by a successful job. It means is the service time of a job given that the next arrival comes after the remaining busy period, i.e:

$$\begin{aligned} F_T(t) &= \Pr\{S < t | S < A\} = \frac{\Pr\{S < t, S < A\}}{\Pr\{S < A\}} \\ &= \frac{\int_0^\infty \Pr\{S < t, S < a\} f_A(a) \mathrm{d}a}{\int_0^\infty \Pr\{S < a\} f_A(a) \mathrm{d}a} \\ &= \frac{\left(\int_0^t F_S(a) f_A(a) \mathrm{d}a + F_S(t) \int_t^\infty f_A(a) \mathrm{d}a\right)}{\int_0^\infty F_S(a) f_A(a) da} \\ &= \frac{\lambda + \mu}{\mu} \left(\int_0^t F_S(a) f_A(a) \mathrm{d}a + F_S(t) \left[1 - F_A(t)\right]\right) \\ &= 1 - \mathrm{e}^{-(\lambda+\mu)t}. \end{aligned}$$

□

In the feedback case, the sink acknowledges each received piece of information by broadcasting the latest information generation timestamp to all the n queues. Upon receiving

the broadcast, each system checks whether it has a piece of information in service older than the broadcasted generation timestamp. If there is such a job in service, it is aborted, thus saving a useless transmission.

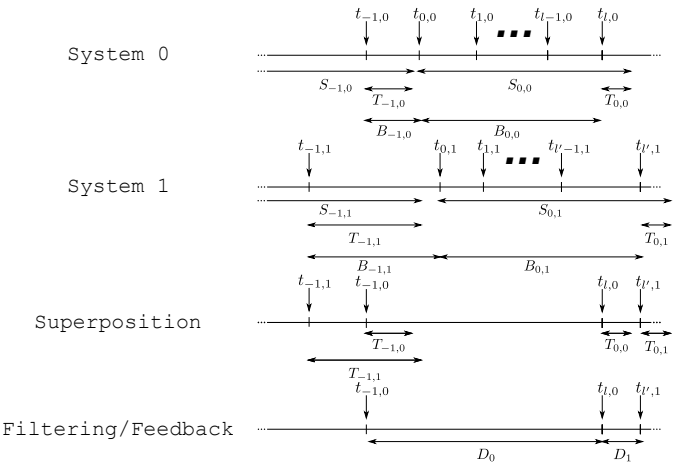

Fig. 3. Example of the time axis for two systems in parallel. Preemption, superposition, and finally filtering.

In order to formally introduce our calculations, we will first consider the behavior of jobs in our systems. Fig. 3 shows an example of the timeline with two systems in parallel. In each system a preemption happens independently from the other system. Generation times are denoted by $t_{i,j}$ where i is the number of the piece of information generated, and j is the system where it is generated. In system 0, job -1 completes service after a residual service time of $T_{-1,0}$. Then job 0 arrives, and is preempted by job 1 and so on, until job l completes service after a service time $S_{0,0}$. Similarly, in system 1 only job -1 and job l' survive. In each system the inter-generation times are denoted $B_{i,j}$.

The surviving jobs are then superimposed before filtering takes place. Since job $(-1, 1)$ is generated before job $(-1, 0)$, but arrives after the latter, job $(-1, 1)$ is filtered out in the AoI calculation. Then, the surviving inter-generation times are denoted D_i, where i is the job number in the resulting thinned process. In general we are dealing with n preemptions based on the generation times, a superposition, and filtering of staler jobs that have managed to survive the preemption.

The AoI, which is a measure of the freshness of the information stored at the sink, has a characteristic sawtooth behavior. When a job is received, the AoI takes the value of the time it took for the job to arrive, from generation to reception (in our case, the service time). Then the AoI grows with slope 1 until the next job is received. If the AoI of the job is less than the current AoI, the AoI drops to the age of the received job; otherwise it continues to grow with slope 1 until the next meaningful event.

In our scheme, jobs do not arrive in generation order (Fig. 4). In the example, job 2 is received after job 3, then is discarded by the sink. The average AoI is given by the average area of a generic trapezoid Q_j scaled by the effective

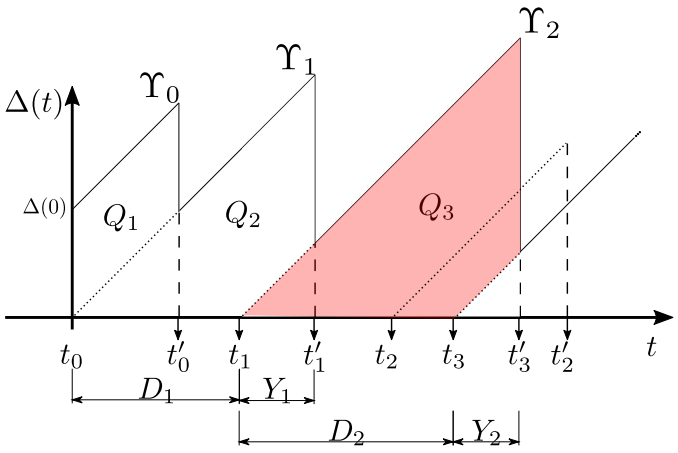

Fig. 4. AoI for a sink receiving jobs out of generation order. t_i is the i-th arrival time, while t'_i is the corresponding (expected) departure time.

rate; an example of a trapezoid is highlighted in Fig. 4. We call D_q the inter-generation times between jobs that are not filtered out, and Y_q the relative service times.

By using reasoning similar to that in [12] for out of order update arrivals, and looking at Fig. 4, we can derive the average pAoI $\overline{\Upsilon}$. It is defined as the average of the AoI just before a job carrying fresh information is received (Υ_q in Fig. 4). It gives the average "worst case scenario" for the freshness of the updates and is useful in case there is an upper limit on the AoI at the sink. It can be computed as the sum of the q-th inter-generation time and the q-th service time:

$$\overline{\Upsilon} = \mathrm{E}[\Upsilon] = \mathrm{E}[\Upsilon_q] = \mathrm{E}[D_q + Y_q] = \mathrm{E}[D_q] + \mathrm{E}[Y_q]\,. \quad (3)$$

III. Number of jobs sent per second

A. LCFS systems with preemption in parallel without feedback loop

Since we have only one source per system, a freshly generated job in a generic system can only find either no jobs in the system, or one job in service. In the former case, the job simply goes into service. In the latter case, it substitutes the existing job in the server, with a service time that is the residual service time of the job it has substituted. Since both the interarrival times and the busy times are generated from a memoryless process, the residual service time will also be exponential. If we call A the random variable associated with the interarrival times, and S the random variable associated with the busy times, then the probability that the generated job will substitute a job in service is given by the probability that the interarrival time is less than the residual service time. Conversely, the probability p_1 of not being filtered out is the probability that the interarrival time is greater than the residual service time, i.e.:

$$p_1 = \Pr\{S < A\} = \int_0^\infty \Pr\{S < t\} f_A(t)\mathrm{d}t$$
$$= \int_0^\infty F_S(t) f_A(t)\mathrm{d}t = \frac{\mu}{\lambda + \mu} = \frac{1}{\rho + 1}.$$

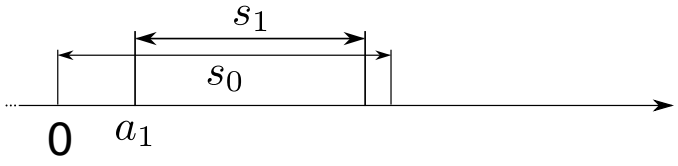

Fig. 5. Time axis for two queuing systems.

Then the effective rate $\lambda_{e,1}$ for n systems in parallel will be:

$$\lambda_{e,1} = p_1 n\lambda = n\frac{\lambda\mu}{\lambda+\mu}. \tag{4}$$

B. LCFS systems with preemption in parallel and feedback loop

We will now examine the second scenario, where feedback from the sink is included. As a first example, we consider only two systems in parallel, discarding the effect of preemption (Fig. 5). If job 0 is generated at time 0, and job 1 is generated after a time a_1, then job 0 will be kept (that is, not discarded) if and only if its service time s_0 is shorter than the sum of the arrival time a_1 of job 1 and its service time s_1. In the case shown in Fig. 5, job 0 is discarded.

In general, if we take the generation time of job 0 as a reference time, this job will not be discarded only if its service time s_0 is less than the minimum of

- the sum of all the interarrival times of subsequent jobs $a_i\,, i = 1, \ldots, n-1$, plus their relative service times $s_i\,, i = 1, \ldots, n-1$, and
- the next interarrival time in system 0, a_0 (job not preempted).

Therefore, the total probability p_2 for a job to not be discarded due to preemption or filtering is given by:

$$\begin{aligned} p_2 &= \Pr\left\{S_0 < \min_{i=1\ldots n-1}\{A_i + S_i\} \wedge S_0 < A_0\right\} \\ &= \Pr\left\{S_0 < \min\left\{\{A_i + S_i \,|\, i = 1\ldots n-1\} \cup A_0\right\}\right\} \\ &= \int_0^\infty \left[1 - F_Q(t)\right] f_S(t)\mathrm{d}t, \end{aligned} \tag{5}$$

where $Q = \min\left\{\{A_i + S_i \,|\, i = 1\ldots n-1\} \cup A_0\right\}$. Since all the $Z_i = A_i + S_i$ are independent and identically distributed, we may drop the index i and write:

$$F_Z(t) = \begin{cases} 1 + \frac{\lambda}{\mu+\lambda}\mathrm{e}^{-\mu t} - \frac{\mu}{\mu+\lambda}\mathrm{e}^{-\lambda t}, & \lambda \neq \mu \\ 1 - (\lambda t + 1)\mathrm{e}^{-\lambda t}, & \lambda = \mu. \end{cases}$$

The CDF of Q is simply:

$$F_Q(t) = 1 - \left[1 - F_Z(t)\right]^{n-1}\left[1 - F_A(t)\right]. \tag{6}$$

By using Eq. (6) in Eq. (5) and integrating over the support of S_0 we can finally write:

$$p_2 = \int_0^\infty \left[1 - F_Z(t)\right]^{n-1}\left[1 - F_A(t)\right] f_S(t)\mathrm{d}t.$$

By manipulating the above for $\lambda \neq \mu$, by changing the integration variable t to $\mathrm{e}^{-t} = q$, and using [18, (3.259.1)], we get:

$$\begin{aligned} p_2 &= \frac{\mu^n}{(\lambda+\mu)^{n-1}}\int_0^\infty \left[1 - \rho\mathrm{e}^{-(\lambda-\mu)t}\right]^{n-1}\mathrm{e}^{-(\lambda n+\mu)t}\mathrm{d}t \\ &= \frac{{}_2F_1\left(\frac{\rho n+1}{1-\rho}, 1-n; \frac{\rho(n-1)+2}{1-\rho}; \rho\right)}{(\rho n+1)(\rho+1)^{n-1}}, \end{aligned} \tag{7}$$

where $\rho = \lambda/\mu$ and ${}_2F_1\left(a, b; c; d\right)$ is the hyper-geometric function [19, (15.1.1)]. For $\lambda = \mu$, by changing the integration variable t to $\lambda t + 1 = q$ we find:

$$p_2 = \lambda\int_0^\infty (\lambda t+1)^{n-1}\mathrm{e}^{-\lambda(n+1)t}\mathrm{d}t = \frac{\Gamma(n, n+1)}{(n+1)^n}\mathrm{e}^{n+1}, \tag{8}$$

where $\Gamma(a, x)$ is the upper incomplete gamma function defined as:

$$\Gamma(a, x) = \int_x^\infty t^{a-1}\mathrm{e}^{-t}dt. \tag{9}$$

Finally, by combining Eq. (7) and Eq. (8), we have:

$$p_2 = \begin{cases} \frac{{}_2F_1\left(\frac{\rho n+1}{1-\rho}, 1-n; \frac{\rho(n-1)+2}{1-\rho}, \rho\right)}{(\rho n+1)(\rho+1)^{n-1}}, & \lambda \neq \mu \\ \frac{\Gamma(n, n+1)}{(n+1)^n}\mathrm{e}^{n+1}, & \lambda = \mu. \end{cases} \tag{10}$$

Then the effective rate $\lambda_{e,2}$ for n LCFS queues with preemption in parallel and feedback loop will be:

$$\lambda_{e,2} = p_2 n\lambda\,. \tag{11}$$

IV. Peak Age of Information

Since the sink discards jobs that are staler than the freshest piece of information it has previously received, the AoI and the pAoI at the sink do not depend on whether preemption is used alone or in conjunction with feedback. The feedback simply moves the filtering from the sink directly to the systems and the only difference will be the number of jobs sent.

The formula to calculate the average pAoI was given in Eq. (3). The average inter-generation time for jobs not preempted and not filtered out is (by using Eq. (11)) simply:

$$\mathrm{E}[D_q] = \mathrm{E}[D] = \left(\lambda_{e,2}\right)^{-1} = \left(p_2 n\lambda\right)^{-1}. \tag{12}$$

The average service time for jobs not preempted and not filtered out is:

$$\mathrm{E}\left[Y_q\right] = \mathrm{E}\left[Y\right] = \mathrm{E}\left[S_0 | S_0 < Q\right], \tag{13}$$

where Q is the random variable defined in Eq. (6). From the definition of conditional expectation we have:

$$\mathrm{E}\left[Y\right] = \int_0^\infty 1 - F_{S_0|S_0<Q}(t)\mathrm{d}t = \int_0^\infty 1 - F_Y(t)\mathrm{d}t. \tag{14}$$

We then find the CDF of Y as:

$$\begin{aligned} F_Y(t) &= \frac{\Pr\left\{S_0 < t \wedge S_0 < Q\right\}}{\Pr\left\{S_0 < Q\right\}} \\ &= p_2^{-1}\Pr\left\{S_0 < t \wedge S_0 < Q\right\} \\ &= p_2^{-1}\int_0^\infty \Pr\left\{S_0 < t \wedge S_0 < q\right\} f_Q(q)\mathrm{d}q \end{aligned}$$

$$= p_2^{-1}\left(\int_0^t F_S(q)f_Q(q)\mathrm{d}q + F_S(t)\int_t^\infty f_Q(q)\mathrm{d}q\right)$$
$$= p_2^{-1}\left(\int_0^t F_S(q)f_Q(q)\mathrm{d}q + F_S(t)\left[1-F_Q(t)\right]\right). \tag{15}$$

For $\lambda \neq \mu$:

$$\begin{aligned}\int_0^t F_S(q)f_Q(q)\mathrm{d}q &= F_Q(t)\\ &- \left[\lambda+(n-1)\mu\right](-1)^{n+1}(1-p_1)^{n-1}\\ &\times \int_0^t \left(1-\rho^{-1}\mathrm{e}^{-(\lambda-\mu)q}\right)^{n-2}\mathrm{e}^{-(\lambda+n\mu)q}\mathrm{d}q\\ &+ n\mu(-1)^{n+1}(1-p_1)^{n-1}\int_0^t \left(1-\rho^{-1}\mathrm{e}^{-(\lambda-\mu)q}\right)^{n-2}\\ &\times \mathrm{e}^{-(2\lambda+(n-1)\mu)q}\mathrm{d}q\,.\end{aligned}$$

We notice that, by changing the integration variable q to $z = \rho^{-1}\mathrm{e}^{-(\lambda+\mu)q}$:

$$\begin{aligned}\psi(\nu;t) &= \int_0^t \left(1-\rho^{-1}\mathrm{e}^{-(\lambda-\mu)q}\right)^{n-2}\mathrm{e}^{-\nu q}\mathrm{d}q\\ &= \frac{\rho^{\frac{\nu}{\lambda-\mu}}}{\lambda-\mu}\int_{\rho^{-1}\mathrm{e}^{-(\lambda+\mu)t}}^{\rho^{-1}} (1-z)^{n-2}z^{\frac{\nu}{\lambda-\mu}-1}\mathrm{d}z\\ &= \frac{\rho^{\frac{\nu}{\lambda-\mu}}}{\lambda-\mu}\left[B\left(\rho^{-1};\frac{\nu}{\lambda-\mu},n-1\right)\right.\\ &\left.- B\left(\rho^{-1}\mathrm{e}^{-(\lambda+\mu)t};\frac{\nu}{\lambda-\mu},n-1\right)\right],\end{aligned} \tag{16}$$

where $B(x;a,b)$ is the incomplete beta function [(6.6.1)][20]. By calling $q_1 = 1-p_1$ and using the previous in Eq. (15), for $\lambda \neq \mu$ we obtain:

$$\begin{aligned}F_Y(t)|_{\lambda\neq\mu} = p_2^{-1}\Bigg(&F_Q(t) - \left[\lambda+(n-1)\mu\right](-1)^{n+1}q_1^{n-1}\\ &\times \frac{\rho^{\frac{\lambda+n\mu}{\lambda-\mu}}}{\lambda-\mu}\left[B\left(\rho^{-1};\frac{\lambda+n\mu}{\lambda-\mu},n-1\right)\right.\\ &\left.- B\left(\rho^{-1}\mathrm{e}^{-(\lambda+\mu)t};\frac{\lambda+n\mu}{\lambda-\mu},n-1\right)\right]\\ &+ n\mu(-1)^{n+1}q_1^{n-1}\frac{\rho^{\frac{2\lambda+(n-1)\mu}{\lambda-\mu}}}{\lambda-\mu}\\ &\times\left[B\left(\rho^{-1};\frac{2\lambda+(n-1)\mu}{\lambda-\mu},n-1\right)\right.\\ &\left.- B\left(\rho^{-1}\mathrm{e}^{-(\lambda+\mu)t};\frac{2\lambda+(n-1)\mu}{\lambda-\mu},n-1\right)\right]\\ &+ F_S(t)\left[1-F_Q(t)\right]\Bigg).\end{aligned}$$

We then plug the previous in Eq. (14), using a similar procedure as for the previous, finally having:

$$\begin{aligned}\mathrm{E}\left[Y\right]|_{\lambda\neq\mu} = p_2^{-1}&\\ &\times\left(\frac{p_1^{n-1}}{\lambda n}{}_2F_1\left(1-n,\frac{\rho n}{\rho-1};\frac{\rho(n+1)-1}{\rho-1};\rho\right)\right.\\ &+ \left[\lambda+(n-1)\mu\right](-1)^{n+1}q_1^{n-1}\int_0^\infty \psi(\lambda+n\mu;t)\mathrm{d}t\\ &- n\mu(-1)^{n+1}q_1^{n-1}\int_0^\infty \psi(2\lambda+(n-1)\mu;t)\mathrm{d}t\\ &\left.- \frac{{}_2F_1\left(\frac{\rho n+1}{1-\rho},1-n;\frac{\rho(n-1)+2}{1-\rho};\rho\right)}{\mu(\rho n+1)(\rho+1)^{n-1}}\right),\end{aligned} \tag{17}$$

where the integrals of $\psi(\nu;t)$ (as defined in Eq. (16)) can be solved numerically. For $\lambda = \mu$:

$$\begin{aligned}\int_0^t F_S(q)f_Q(q)dq &= F_Q(t)\\ &- \frac{\mathrm{e}^{-n}}{n^{n-1}}\Big[(n-1)\left(\Gamma\left(n-1,n(\lambda t+1)\right)-\Gamma\left(n-1,n\right)\right)\\ &- \Gamma\left(n,n(\lambda t+1)\right)+\Gamma\left(n,n\right)\Big].\end{aligned}$$

By inserting the previous in Eq. (15) we obtain:

$$\begin{aligned}F_Y(t)|_{\lambda=\mu} = p_2^{-1}\Bigg(&F_Q(t)\\ &- \frac{\mathrm{e}^{-n}}{n^{n-1}}\Big[(n-1)\left(\Gamma\left(n-1,n(\lambda t+1)\right)-\Gamma\left(n-1,n\right)\right)\\ &- \Gamma\left(n,n(\lambda t+1)\right)+\Gamma\left(n,n\right)\Big] + F_S(t)\left[1-F_Q(t)\right]\Bigg).\end{aligned} \tag{18}$$

Finally by substituting the previous in Eq. (14) we obtain:

$$\begin{aligned}\mathrm{E}\left[Y\right]|_{\lambda=\mu} = p_2^{-1}&\\ &\times\left[\left(1-\frac{1}{\mu}\right)p_2 + \frac{\mathrm{e}^{-n}}{n^{n-1}}(n-1)\int_0^\infty \phi(n-1,t)\mathrm{d}t\right.\\ &\left.- \frac{\mathrm{e}^{-n}}{n^{n-1}}\int_0^\infty \phi(n,t)\mathrm{d}t\right],\end{aligned} \tag{19}$$

where

$$\phi(a,t) = \Gamma\left(a,n(\lambda t+1)\right) - \Gamma\left(a,n\right).$$

Finally, by using Eq. (17) and Eq. (19) we obtain:

$$\mathrm{E}\left[Y\right] = \begin{cases}\mathrm{E}\left[Y\right]|_{\lambda\neq\mu}, & \lambda\neq\mu\\ \mathrm{E}\left[Y\right]|_{\lambda=\mu}, & \lambda=\mu\end{cases}. \tag{20}$$

By summing up Eq. (12) and Eq. (20), we finally find the average pAoI $\overline{\Upsilon}$:

$$\overline{\Upsilon} = \mathrm{E}[D] + \mathrm{E}\left[Y\right]. \tag{21}$$

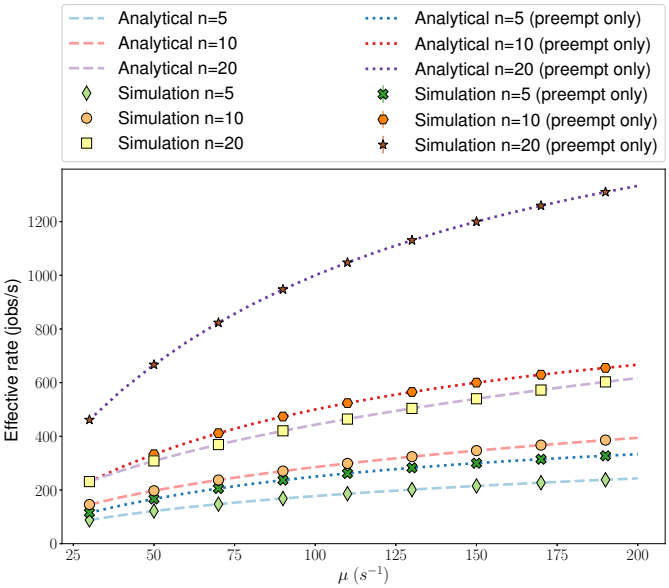

Fig. 6. Effective rate, simulation vs analytical.

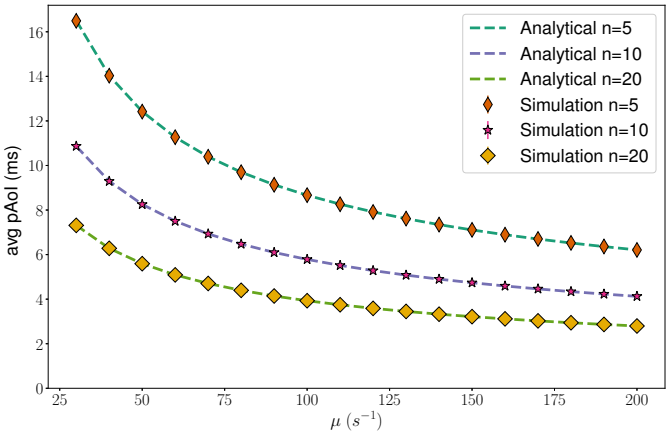

Fig. 7. Average pAoI, simulation vs analytical.

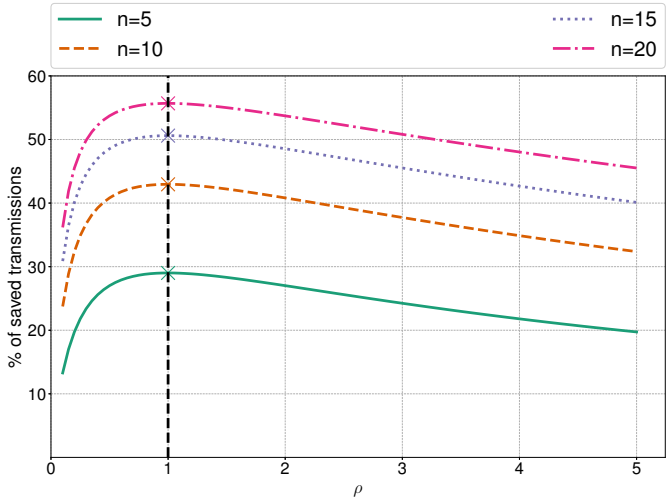

Fig. 8. Percentage of saved jobs transmissions per second.

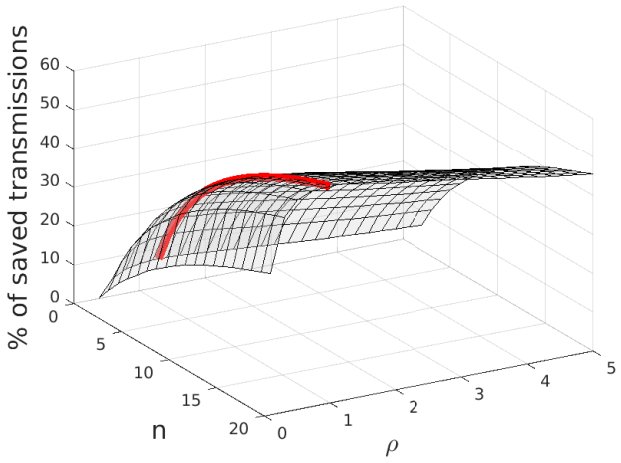

Fig. 9. Percentage of saved jobs transmissions per second, isometric view. The red line is the line where $\rho = 1$.

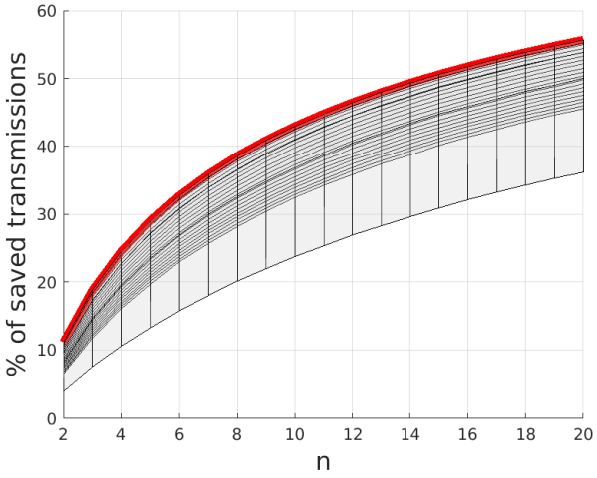

Fig. 10. Percentage of saved jobs transmissions per second, view from a plane parallel to the axis representing n and the one representing the percentage of saved job transmissions per second. The red line is the line where $\rho = 1$.

V. Numerical Results

We conducted simulation studies using OMNeT++ [21]. We fixed $\lambda = 100$ s^{-1} – i.e. the maximum number of actions per second the user could send, as described in Section I –, $n = \{5, 10, 20\}$ and let μ vary between 30 and 200 fps (frames per second). All plots are presented with 95% confidence intervals, allowing for a sufficient warm-up period before taking measurements. All the plots make use of a black and white printer-friendly and accessible color scheme [22].

First we checked the effective rates (Eq. (4) and Eq. (11)) against the simulation (Fig. 6). As we can see the simulation agrees with the analytical results. Then we tested the pAoI (Eq. (21)) against the simulations (Fig. 7), again confirming our results. We can see that the pAoI never goes over the 100 ms threshold, after which, the user starts to perceive the lag.

We then plotted in Fig. 8 the percentage of saved jobs transmissions per second using the feedback mechanism, defined as:

$$\frac{\lambda_{e,1}(\lambda,n)-\lambda_{e,2}(\lambda,n)}{\lambda_{e,1}(\lambda,n)}\cdot 100=\left(1-\frac{p_2(\rho,n)}{p_1(\rho)}\right)\cdot 100. \quad (22)$$

As we can see, even at a low number of systems in parallel, there is a substantial saving in jobs sent per second. Also we can see that there is a maximum in $\rho = 1$ (dashed vertical line) independent from n. In order to confirm our finding we plotted the percentage of saved jobs against both n and ρ. In Fig. 9 the isometric view, where the red line is the value of the percentage when ρ = 1. We can appreciate the fact that effectively, for a fixed number of systems, the maximum job saving happens at ρ = 1, by taking the view from a plane parallel to the axis representing n and the one representing the percentage of saved job transmissions per second (Fig. 10).

VI. Conclusions and future work

In this paper we studied reliability, timeliness and load reduction in an hybrid MEC/cloud game streaming infrastructure by modeling it as a network of M/M/1 LCFS systems with preemption. In particular, reliability is given by task replication in n MEC-servers and timeliness is analyzed through the Age of Information metric. We addressed load reduction by leveraging the X2 interface at the edge, in order to abort staler computations, and translating the mechanism as a feedback loop from a sink to the various queueing systems.

We studied our model, with and without a feedback loop aimed at decreasing the number of transmissions from the nodes, while maintaining low average AoI at the sink. We have derived expressions both for the number of jobs sent per second for the two scenarios, and the peak age of information, and compared them to simulations.

We found that the use of the feedback loop significantly decreases the number of jobs sent, thus saving transmissions, that translates to reducing the load per MEC-server. We also found that in order to maximally reduce transmissions with the feedback loop in place, for a fixed number of systems, the optimal value is when the average arrival rate equals the inverse of the average busy time in all the systems.

Future work will involve the calculation for an expression for the average AoI, the addition of a delay in the feedback loop to study possible long transmission times from the UEs, the introduction of an error channel both from the UE to the edge and back, and the introduction of a limited resource channel involving collisions. Also, in order to account e.g. for different delays from the MEC-servers, different service rates and distributions for the busy time could be taken into account, modeling different routes the piece of information takes in order to reach the UE.

Acknowledgments

This work was supported by the "Excellence Center at Linköping-Lund in Information Technology (ELLIIT), Sweden".

References

[1] M. Manzano, J. A. Hernández, M. Urueña, and E. Calle, "An empirical study of cloud gaming," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, Nov 2012, pp. 1–2.

[2] W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung, and C. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[3] Y. Lin and H. Shen, "Leveraging fog to extend cloud gaming for thin-client mmog with high quality of experience," in *2015 IEEE 35th International Conference on Distributed Computing Systems*, June 2015, pp. 734–735.

[4] Z. Wen and H. Hsiao, "Qoe-driven performance analysis of cloud gaming services," in *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2014, pp. 1–6.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2731–2735.

[6] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[7] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*, March 2012, pp. 1–6.

[8] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 2574–2578.

[9] E. Najm and E. Telatar, "Status updates in a multi-stream M/G/1/1 preemptive queue," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 124–129.

[10] C. Kam, S. Kompella, and A. Ephremides, "Age of information under random updates," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 66–70.

[11] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of information of multiple sources with queue management," in *2015 IEEE International Conference on Communications (ICC)*, pp. 5935–5940.

[12] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1360–1374, March 2016.

[13] R. D. Yates, "Status updates through networks of parallel servers," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2281–2285.

[14] Q. He, D. Yuan, and A. Ephremides, "Optimizing freshness of information: On minimum age link scheduling in wireless systems," in *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2016, pp. 1–8.

[15] ——, "On optimal link scheduling with min-max peak age of information in wireless systems," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7.

[16] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2016, pp. 844–851.

[17] Y. P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 561–565.

[18] "3–4 - definite integrals of elementary functions," in *Table of Integrals, Series, and Products*, 7th ed., A. Jeffrey, D. Zwillinger, I. Gradshteyn, and I. Ryzhik, Eds. Boston: Academic Press, 2007, pp. 247 – 617.

[19] F. Oberhettinger, "Hypergeometric functions." in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York, NY, USA: Dover Publications, Inc., 1972, ch. 15, pp. 556–565.

[20] P. J. Davis, "Gamma function and related functions." in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York, NY, USA: Dover Publications, Inc., 1972, ch. 6, pp. 255–274.

[21] A. Varga, "The OMNET++ discrete event simulation system," in *ESM'01*, 2001.

[22] Cynthia Brewer, Mark Harrower and The Pennsylvania State University. (2013) ColorBrewer. http://colorbrewer2.org.