mathematics

MDPI

Article

The Delay Time Profile of Multistage Networks with Synchronization

Yonit Barron

Industrial Engineering and Management, Ariel University, Ariel 40700, Israel; ybarron@ariel.ac.il

Abstract: The interaction between projects and servers has grown significantly in complexity; thus, applying parallel calculations increases dramatically. However, it should not be ignored that parallel processing gives rise to synchronization constraints and delays, generating penalty costs that may overshadow the savings obtained from parallel processing. Motivated by this trade-off, this study investigates two special and symmetric systems of split–join structures: (i) *parallel* structure and (ii) *serial* structure. In a parallel structure, the project arrives, splits into m parallel groups (subprojects), each comprising n subsequent stages, and ends after all groups are completed. In the serial structure, the project requires synchronization after each stage. Employing a numerical study, we investigates the time profile of the project by focusing on two types of delays: delay due to synchronization overhead (occurring due to the parallel structure), and delay due to overloaded servers (occurring due to the serial structure). In particular, the author studies the effect of the number of stages, the number of groups, and the utilization of the servers on the time profile and performance of the system. Further, this study shows the efficiency of lower and upper bounds for the mean sojourn time. The results show that the added time grows logarithmically with m (parallelism) and linearly with n (seriality) in both structures. However, comparing the two types of split–join structures shows that the synchronization overhead grows logarithmically undr both parallelism and seriality; this yields an unexpected *duality* property of the added time to the serial system.

Keywords: parallelism; synchronization overhead; sojourn time; queueing; split–join networks

MSC: 90B15; 90B22; 60J28

Citation: Barron, Y. The Delay Time Profile of Multistage Networks with Synchronization. *Mathematics* **2023**, *11*, 3232. https://doi.org/10.3390/math11143232

Academic Editors: Maria Do Castelo Gouveia and Carla Oliveira Henriques

Received: 23 June 2023
Accepted: 11 July 2023
Published: 23 July 2023

1. Introduction

Nowadays, the concept of a "project" is becoming much more general and multidimensional. The interaction between projects and servers has grown in complexity, originating from today's practice that projects are predominantly parallel subprojects that involve parallel calculations. Examples of such departures from the traditional one-server-per-project model include data centers at Google, Microsoft, and Facebook, where parallelism can occur both at the hardware and software levels [1–3], process mining and business activities evaluations [4–6].

Split–join networks are a key modeling tool for parallelism in operations research, queueing models, and supply chain management. The basic split–join network (also known as the *fork–join* network), is a one-stage network (see Figure 1). A stream of projects arrives at the split node (the first pink triangle), where each project is split into m tasks that are allocated to m parallel servers (the gray circles). A task may have to wait in a queue (the black rectangle) until its server finishes all previous tasks. After the task is completed, it waits in a join-type queue (the second pink triangle) until all the other $m-1$ tasks of the same project are completed. When all m tasks of the same project are completed, they rejoin (are synchronized) at the join node. The *sojourn time* is the time from the arrival of the project until its completion. (i.e., until its departure from the join node). Here, the terminology "server" is used when talking about something that processes tasks, such as

a machine, a single CPU core, or a single thread. Additionally, the label "queue" is used to represent the place where the task waits. Accordingly, there are two types of queues: a queue before each server where the task is delayed due to a busy server, and a queue where a group waits until all the parallel groups are completed. In the remainder of the paper, the author refers to these two types of queues as *operational* queues and *synchronization* queues, respectively.

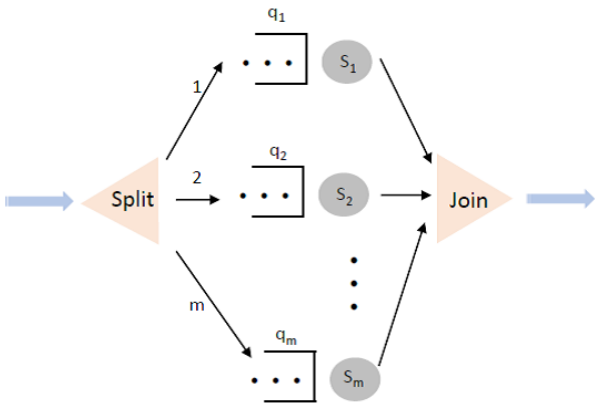


Figure 1. The basic split–join network.

A split–join network with a more general topological structure is illustrated in Figure 2. Here, the project starts with task 1, and then splits into two parallel groups, starting with tasks 2 and 8, respectively. After finishing task 2, the project continues to three parallel groups, the first of which consists of two consecutive tasks (tasks 3 and 6). Finishing all these three groups, the project continues with task 7. After finishing tasks 7 and 8, the project is completed. The operational queues are marked by striped black rectangles, and the join (synchronization) queues are marked by blue half-ellipses. Note that the splitting action takes no time; similarly, the joining action takes no time (after all the relevant groups have arrived).

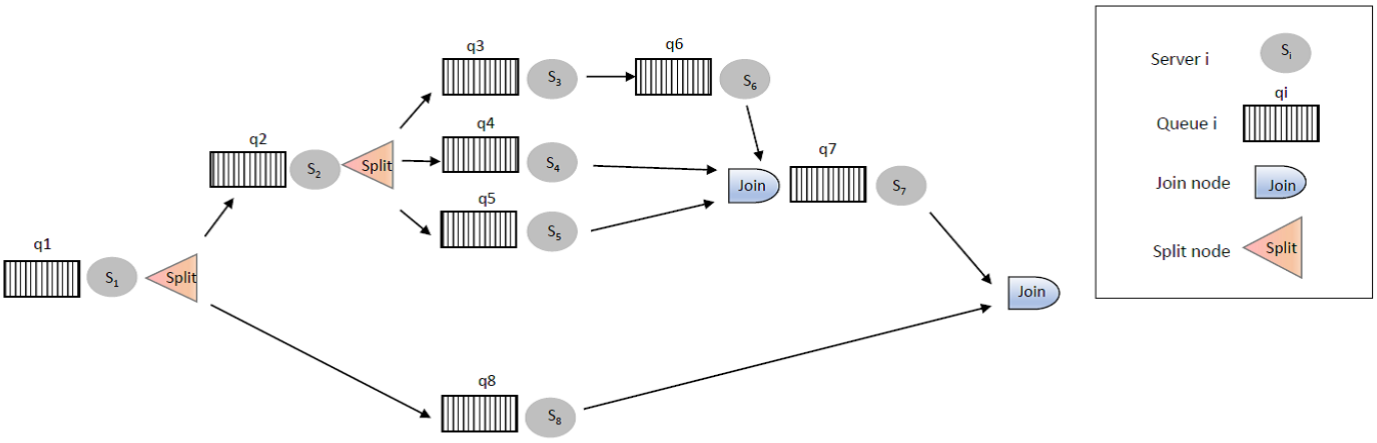


Figure 2. A general split–join network.

The scope of applications that use the split–join structure in reality is huge. It includes data center services, web searching, social networking and big data analysis, systems with a wide range of queue policies and service time distributions, systems with multiple servers per split node for failure finding and load balancing, and systems with a varying number of parallel tasks or sharing services due to limited resources [3,7,8]. In manufacturing, the split–join structure is used in assembly systems of high-tech equipment manufacturers (OEMs) that require several parts to be processed simultaneously at separate work stations or plant locations [9–11]. In project management and supply chain management, the

split–join structure is typically used for representing the arrival of an order composed of several different items or products from a vendor, or for synchronization between arriving and departing vehicles at the transshipment location [12,13]. The split–join structure is also prevalent in healthcare and, in particular, in emergency departments, where multiple customer classes share multiple processing resources [14,15], in supply chain networks of high-tech manufacturers with multiple suppliers that each produce a unique component of the product [16–19], and in ocean transport for managing container terminals [20].

This study investigates two special split–join networks with a symmetric topological structure. The first is the *parallel* split–join network (see Figure 3). The project arrives, and splits into m parallel groups, each including n subsequent stages allocated to n different servers (in total, $m \times n$ tasks). The project leaves after all its groups are completed. It is assumed that projects arrive according to a renewal process and are served in a first-come-first-served (FCFS) order. Importantly, note that the FCFS policy is used in many settings, such as the Google Borg task scheduler and the management of multiserver jobs in the cloud-computing industry [8]. However, note that other scheduling rules such as FIFO or $c\mu$ rules are also examined in the literature (see, e.g., [14,15,21]).

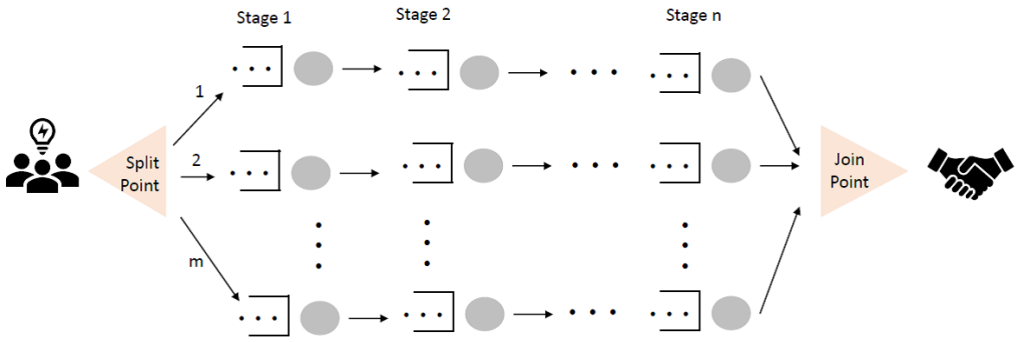


Figure 3. A typical parallel split–join network.

In reality, we are indeed witnessing that projects must be processed in parallel. However, in practice, we cannot ignore that parallel processing needs to be more synchronized. Characterizing this constraint, this study introduces the second type of split–join network, namely, the *serial split–join* network. The serial network has similar features to the parallel split–join network, except that it has, in addition, multiple synchronization queues. Specifically, a project arrives and splits into m parallel groups; each of which is composed of n stages. When all groups of the same project complete stage j, they are obliged to reunite, after which they can continue to the next $(j+1)$-th stage; this procedure is the same for all stages. The project can exit the system when all groups in the last stage (the n-th stage) are completed. Here, there are $n \times m$ operational queues, but n synchronization queues.

Although the basic split–join network has been studied intensively in the literature, e.g., [22–28], no analytical expressions are known for the joint steady-state number of projects in the queues, nor for the mean sojourn time. There is agreement that split–join networks are another of those infamous queueing systems for which the stationary distribution is intractable; the complexity of the analysis is explained by the dependence of the times of the projects, due to their common moments of arrival [3,29]. Most exact results on split–join networks are limited to systems with two parallel servers. For example, Nelson and Tantawi [30] and Nelson et al. [31] obtain an exact expression for a two-node M/M/1 homogeneous split–join network where the jobs arrive according to a Poisson process and the service times are i.i.d exponentially distributed random variables (r.v.s). For split–join networks with more than two parallel servers, only approximations for the performance measures are obtained. For example, Nelson and Tantawi [30] used a scaling approximation technique to approximate the mean sojourn time in an M/M/1 split–join system. Ko and Serfozo [27] provided results on G/M/1 queues, and Fiorini [32] and Wang et al. [8] studied M/G/1 queues. Networks with different types of products at different machines

were discussed in Ding [9]. For the general GI/G/1 split–join networks, upper and lower bounds on the mean sojourn time were derived by Baccelli and Makowski [22,23], Baccelli et al. [24,25], Lebrecht and Knottenbelt [33], Kemper and Mandjes [34], and, more recently, Enganti et al. [7] and Grobunova and Vishnevsky [2]. Ko and Serfozo [26] developed bounds and approximate expressions for evaluating the mean sojourn time and queue length distribution at join nodes. Takahashi et al. [35] used matrix analytic methods and assumed finite queue sizes, called buffers. Using the latter method, Qiu et al. [36] developed an efficient algorithm to approximate the distribution of response time in homogeneous multi-MAP/PH/1 split–join networks. Nelson et al. [30] compared four different structures of split–join networks: with/without central queueing and splitting/not splitting projects. Most of the above papers focus on systems where the number of servers is finite. Work on the heavy-traffic processing limit has been performed by, e.g., Varma and Makowski [37], Tan and Knessl [38], Knessl [39], Kushner [40], Atar et al. [14], Wang et al. [8], Schol et al. [19], Nguyen et al. [3], Zeng et al. [41] and Meijer et al. [42].

The above literature review shows that a great deal of effort has been devoted to the understanding and analysis of split–join queueing networks. This effort mainly focuses on deriving the sojourn time of a project, usually when there is only one type of project and two parallel servers. Indeed, the sojourn time is known to be intractable in more complex split–join networks; thus, only (upper and lower) bounds and approximations have been provided for them.

As far as the author knows, no explicit results are available for networks with more than two parallel groups or with more synchronization points. Despite their simple and symmetric structure, parallel and serial split–join networks have hardly been investigated and, at present, very little is known about their performance and time profile. Note that, in practice, parallel processing is subject to synchronization constraints and delays generating penalty costs that can offset the gains obtained from such processing. These include the cost of intermediate storage of uncompleted products, or the cost of memory in computer networks where subprojects are processed by different servers and then wait in the synchronization buffer. Thus, a thorough study of the performance and time profile of these systems is lacking but necessary. This study aims to take a first step toward filling this research gap.

To this end, the author focuses on the mean sojourn time of the parallel and the serial systems. This starts with the derivation of lower and upper bounds for the mean sojourn time; each bound emphasizes a different aspect of the time profile. Using extensive simulations, the efficiency of these bounds are evaluated, providing some insights into the interplay between operational delays (due to overloaded servers or limited resources) and synchronization delays. Then, a sensitivity analysis is carried out to study how the time profile is affected by different parameters of the networks, i.e., by the number of groups m, the number of stages n, and the utilization of the servers. Clearly, operational and synchronization delays are interdependent. Thus, accurately cataloging the time profile according to the different types of delays is not applicable and, hence, not possible. However, it is reasonable to attribute the number of groups to the synchronization delays, and the number of stages to the operational delays. In the serial split–join network, the overhead incurred due to additional synchronizations is significant; thus, comparing the serial network with the parallel network can improve our understanding of the time profile, the types of delays, and the impact of the various parameters on the mean sojourn time.

The contributions of this paper are summarized as follows. Two multistage split–join networks are introduced: the parallel network and the serial network. To the best of the author's knowledge, such networks have hardly been discussed in the literature (with the exception of the two-stage serial split–join network discussed in Ko [29]). These networks can serve as applied models (or as effective approximations) in many fields, such as industry and manufacturing, computer modelling, reliability systems, and supply chain management. Moreover, while most existing studies on split–join networks deal with the sojourn times, this study differs by focusing on the time profile and distinguishing between

two types of delays: synchronization delay (due to parallelism) and operational delay (due to seriality). Diagnosing the time profile and studying the effect of the parameters on the different types of delays can serve as a practical tool for decision makers in setting the optimal schedule for the project stages and allocating optimal resources at each stage in order to maximize its economic profitability. Furthermore, motivated by real-world examples, this study compares, numerically, the two networks and studies the effect of imposing synchronization constraints on the system's performance and time profile. Doing so provides a better understanding of the interplay between parallelism and seriality, and addresses how to deal with situations of adding constraints or a change in resource allocation.

The main results can be summarized as follows: (i) Synchronization delays have a logarithmic impact $O(\ln m)$ on the mean sojourn time, while operational delays have a linear impact $O(n)$; (ii) When increasing the number of stages n and the number of groups m, the impact of these delays diminishes; (iii) The impact of parallelism and seriality is (almost) independent of the utilization; (iv) Contrary to expectations, the serial network is less sensitive to changes (and sometimes even robust to them), compared to the parallel one; (v) Comparing the two networks shows that the ratio of the mean sojourn time is increasing logarithmically in both n and m. As a result, this further reveals a kind of duality property, implying that the extra time in the serial network is relatively similar due to parallelism and seriality; (vi) Finally, the results show that, in most cases, increasing servers' utilization slightly obscures the differences between the systems.

In summary, this analysis may shed some light on the time profile of multistage networks and, in particular, on the interplay between different types of delays: delays due to overloaded servers (seriality) and delays due to synchronization constraints (parallelism). As such, the analysis presented here can be used for estimation purposes when designing optimal multistage networks and allocating resources when more constraints are needed or, alternatively, in situations when reducing unnecessary synchronizations is possible.

The rest of the paper is organized as follows. Section 2 describes the parallel split–join network. Section 3 is devoted to preliminary results. Analyses of parallel and serial split–join networks are presented in Sections 4 and 5, respectively. Finally, Section 6 concludes and discusses potential avenues of future research.

2. Description of the Parallel Split–Join Network

A project arrives and splits into m parallel groups, each including n subsequent stages allocated to n different servers (in total, $m \times n$ tasks). When group i, $i \in \{1, \ldots, m\}$, completes its n stages, it enters the synchronization queue, waiting for the other $(m-1)$ groups of the same project to be completed. The project leaves after all its groups are completed. A typical parallel split–join network is presented in Figure 3.

Note that, after splitting, each group goes through all n stages/tasks; thereafter, it waits at the synchronization node (join point). Thus, the system has one synchronization queue and $n \times m$ operational queues. The definitions and notations to be used throughput this paper are now introduced:

- Let A_k, $k = 1, 2, \ldots,$, with $A_0 = 0$, be the arrival time of the k-th project. Assume that the arrival process is a renewal process with rate λ_k (specifically, this paper focuses on the Poisson process with rate λ). Let $\nu_k = A_k - A_{k-1}$, $k = 1, 2, \ldots$ be the inter-arrival time between the k-th project and the $(k-1)$-th project. Thus, $\nu_k, k = 1, 2, \ldots$ are identical independent (i.i.d) random variables (r.v.s) with average $1/\lambda_k$;
- Upon arrival, each project k is split into m parallel groups. Alternatively, one can think of this as m parallel treatments that the project may simultaneously go through. Each group i, $i = 1, \ldots, m$ includes n sequential stages/tasks. Completion of stage j, $j = 1, \ldots, n-1$ enables the group to continue to stage $j+1$. In what follows, the index i to group $i = 1, \ldots, m$, the index j to stage $j = 1, \ldots, n$, and the double-index (i, j) to task j in group i are used;
- Each task is allocated independently to its own station. The station is characterized by an infinite queue and a single server. The queue is managed according to FCFS

discipline. As a result, a task (i,j) of project k cannot enter the server before its predecessor tasks (i,j) of projects $1,2,\ldots,k-1$;

- Let $S_k(i,j)$ be the service time of task (i,j) of project k. Assume that the times $S_k(i,j)$ are independent r.v.s in i, j, and k, having exponential distribution with rate $\mu_k(i,j)$;
- A project k is completed when all its groups finish their n stages. Assume that the time of that final synchronization is negligible. Thus, when all groups are finished, they are reunited with no time, and the project immediately leaves the system. Clearly, the FCFS discipline implies that a project cannot leave before its predecessors;
- A sufficient and necessary condition for stability of the station (i,j) is $E(S_k(i,j)E(\nu_k) = \mu_k(i,j)/\lambda_k < 1$, $k = 1,2,\ldots$The system is stable if $\rho_k(i,j) = \mu_k(i,j)/\lambda_k < 1$ for all i,j,k [21,23];
- Let $W_k(i,j)$ be the waiting time of the project k at station (i,j) (i.e., the operational delay), and denote by $T_k(i,j) = S_k(i,j) + W_k(i,j)$ as the sojourn time (i.e., the waiting time plus the service time).

A snapshot of the system at time $t \geq 0$ can be modeled by the m-column vector $\mathbb{L}(t) = (L_1(t), L_2(t), \ldots, L_m(t))^T$, where $L_i(t)$ is an n-row vector $L_i(t) = (L_{i,1}(t), L_{i,2}(t), \ldots, L_{i,n}(t))$. The component $L_{i,j}(t)$ indicates the number of tasks at station (i,j) at time t (i.e., $L_{i,j}(t) - 1$ tasks are waiting at the queue, along with one task at the server). Recall that the synchronization action at the end join point does not take any time. In addition, let $N_{i,P}(t)$, $i = 1,\ldots,m$ be the number of groups waiting at the joint point at time t. It is easy to verify that $N_{i,P}(t)$ satisfies

$$N_{i,P}(t) = \underset{1\leq i\leq m}{Max}\left\{\sum_{j=1}^{n} L_{i,j}(t)\right\} - \sum_{j=1}^{n} L_{i,j}(t).$$

Example 1. *Let $m = 5, n = 4$, and assume that project 7 arrives. Figure 4 demonstrates the system as observed by project 7. Here, $\mathbb{L}(t^-) = ((0,0,1,1,2),(0,3,2,0,1),(1,0,1,1,3),(0,0,0,0,0))^T$. (Use t^- (t^+) for the time just before (after) time t.) Hence, project 7 enters immediately to servers $1,2$ and 3 and $\mathbb{L}(t) = ((1,0,1,1,2),(1,3,2,0,1),(2,0,1,1,3),(1,0,0,0,0))^T$. We also see that $\underset{1\leq i\leq m}{Max}\left\{\sum_{j=1}^{4} L_{1,j}(t)\right\} = Max\{5,7,7,1\} = 7$, leading to $N_{1,P}(t) = 2, N_{2,P}(t) = 0, N_{3,P}(3) = 0$, and $N_{4,P}(t) = 6$. Examples of possible transitions from $L_1(t) = (1,0,1,1,2)$ to $L_1'(t^+)$ are, e.g., at rate $\mu_7(1,1)$, we reach $L_1'(t^+) = (0,1,1,1,2)$, and at rate $\mu_6(1,3)$, we reach $L_1'(t^+) = (1,0,0,2,2)$. Similarly, from $L_2(t) = (1,3,2,0,1)$, we reach $L_2'(t^+) = (0,4,2,0,1)$ and $L_2'(t^+)$ at rates $\mu_7(2,1)$ and $\mu_4(2,2)$, respectively.*

As mentioned in Section 1, accurate analytical analysis of the parallel network is complicated and intractable. Thus, upper and lower bounds for the sojourn time of the parallel network are derived; then, a numerical analysis is used to evaluate the efficiency of these bounds and to study the time profile of the system. The author first presents some preliminary results to be used.

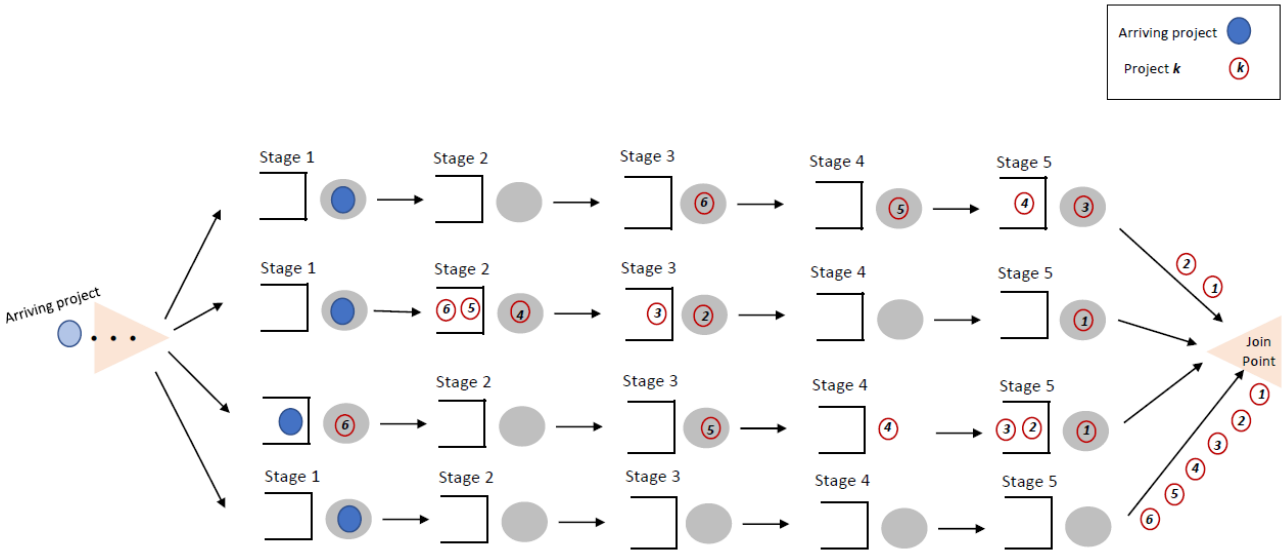


Figure 4. A snapshot of the parallel system upon arrival of project 7.

3. Preliminary Results

3.1. Tandem System

Assume an M/M/1 system with a Poisson arrival process with rate λ, exponentially distributed service time with rate μ, and utilization $\rho = \lambda/\mu < 1$. It is well known that the sojourn time T in such a system is an exponentially distributed r.v. with rate $(\mu - \lambda)$ and mean $1/(\mu - \lambda)$. Applying Burke's theorem [43], the departure process is also a Poisson process with rate λ.

Next, consider a tandem system that consists of n stations, each with one server and infinite queue. The service time of server j is an independent r.v. with distribution $exp(\mu_j)$. Customers join the first queue according to a Poisson process with rate λ, and, on completing service, immediately enter the next queue. We have the follwing:

- The departure process from the first station (server), which is now also the arrival process of the second station (queue), is a Poisson process with rate λ, provided that the queue is in equilibrium. This can be achieved if $\lambda < \mu_1$ [43];
- Recursively, it is easy to prove that the departure rate from station j $(j = 1, \ldots, n-1)$, which is now also the arrival process of the subsequent station $j+1$, is a Poisson process with rate λ, provided that $\lambda < \mu_j$;
- The sojourn times at station j, $j = 1, \ldots, n$ are mutually independent [44].

As a result, we obtain Corollary 1.

Corollary 1. *Under the condition that* $\lambda < \min_{j=1,\ldots,n} \{\mu_j\}$, *the sojourn time* T_j *at station* j *is an exponentially distributed r.v. with rate* $(\mu_j - \lambda)$; *the total sojourn time at the tandem system* $\sum_{j=1}^{n} T_j$ *has a hyperexponential distribution with average*

$$E\left[\sum_{j=1}^{n} T_j\right] = \sum_{j=1}^{n} \frac{1}{\mu_j - \lambda}.$$

Remark 1. *Note that the hyperexponential distribution is a special case of the phase-type (PH) distribution family with representation* $(\alpha, \mathbb{T})$ *of order* n, *where* α *is the initial probability* $(1 \times n)$ *vector and* $\mathbb{T}$ *is the* $(n \times n)$ *transition rate matrix among the transient states. More about the PH distribution can be found in Latouche and Ramaswami [45]. For the above tandem system, the* $(1 \times n)$ *vector* α *and* $(n \times n)$ *matrix* $\mathbb{T}$ *are given by*

$$\alpha = (1, 0, \ldots, 0), \quad \mathbb{T} = \begin{pmatrix} (\mu_1 - \lambda) & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & (\mu_n - \lambda) \end{pmatrix}.$$

3.2. Associated Random Variables

Definition 1. *The R-valued r.v.s* $\{X_1, \ldots, X_K\}$ *are said to be associated if and only if*

$$E[f(X)g(X)] \geq E[f(X)]E[g(X)]$$

for all monotonic non-decreasing mappings $f, g : R^K \to R^1$ *for which the expectations exist [24].*

Definition 2. *The R-valued r.v.s* $\{\overline{X}_1, \ldots, \overline{X}_K\}$ *form independent versions of the r.v.s* $\{X_1, \ldots, X_K\}$ *if*

(i) the r.v.s $\{\overline{X}_1, \ldots, \overline{X}_K\}$ *are mutually independent;*

(ii) for every $1 \leq k \leq K$*, the r.v.s* X_k *and* $\overline{X}_k$ *have the same probability function.*

The following properties are an easy consequence of Definitions 1 and 2.

Properties (Baccelli and Makowski [23]):

P1. Independent r.v.s are associated;

P2. The union of independent collections of associated r.v.s forms a set of associated r.v.s;

P3. Any subset of a family of associated r.v.s forms a set of associated r.v.s;

P4. Any monotonic non-decreasing function of associated r.v.s generates a set of associated r.v.s;

P5. If the r.v.s $\{X_1, \ldots, X_K\}$ are associated, then the inequality

$$p\left[\max_{1 \leq k \leq K} X_k \leq x\right] \geq p\left[\max_{1 \leq k \leq K} (\overline{X}_k \leq x)\right] = \prod_{i=1}^{K} p(\overline{X}_k \leq x)$$

holds true for all x in $\mathbb{R}$.

4. The Parallel Split–Join System: Analysis and Bounds

Let us start with some important results for the parallel system. Recall that projects arrive according to a renewal process and the service times are exponential.

Claim 1. ***Generalized Lindley equation.*** *The classic Lindley equation [46] for the GI/G/1 system can be expanded to the parallel system as follows. For* i, j*, and* k*, we have*

$$W_{k+1}(i,j) = \left[\sum_{l=1}^{j} T_k(i,l) - \sum_{l=1}^{j-i} T_{k+1}(i,l) - v_{k+1}\right]^+, k = 1, 2, \ldots, j = 1, 2, \ldots, n, i = 1, \ldots, m, \quad (1)$$

where $[x]^+ = \max(x, 0)$.

Proof. The proof is given in Appendix A. □

Assume project k arrives. Let $T_k(i)$ be the total sojourn time of group i, i.e., the time elapsed from stage 1 to n (not including a possible final synchronization delay), and let T_k

be the total sojourn time of project k in the system. Immediately after splitting, each group continues independently until the final synchronization; thus, we obtain

$$T_k(i) = \sum_{j=1}^{n} T_k(i,j), \;\; i = 1, \ldots, m.$$
$$T_k = \underset{i=1,\ldots,m}{Max} \{T_k(i)\}. \tag{2}$$

Claim 2. *(1). The set of random variables* $\{W_l(i,j), S_l(i,j), -\nu_l, i = 1, \ldots, m, j = 1, \ldots, n, l = 1, \ldots, k\}$ *are associated.*

(2). The set of random variables $\{T_k(i), i = 1, \ldots, m\}$ *are associated.*

Proof. The proof of Claim 2(1) is obtained by applying Claim 1 and a double induction on i and j, and is detailed in Appendix B. Applying **P3**, we obtain that any subset of a family of associated r.v.s forms a set of associated r.v.s. Now, consider the set $\{W_k(i,j), S_k(i,j)$, i=1, $\ldots, m, j = 1, \ldots, n\}$, and note that

$$T_k(i) = \sum_{j=1}^{n} (W_k(i,j) + S_k(i,j)), i = 1, \ldots, m. \tag{3}$$

The times $\{T_k(i), i = 1, \ldots, m\}$ are nondecreasing functions of associated r.v.s, and, thus, by applying **P4**, we obtain that they are associated, which completes the proof of Claim 2(2). □

Conclusion 1. *Applying **P5** yields that*

$$p\left[\max_{1\le i\le m} \{T_k(i)\} > t\right] \le 1 - \prod_{i=1}^{m} \left[p(\overline{T}_k(i) \le t)\right]. \tag{4}$$

Conclusion 1 enables us to compare two systems. The left side of Equation (4) refers to the parallel system. Here, applying (2) yields that $p\left[\max_{1\le i\le m} \{T_k(i)\} > t\right] = p(T_k > t)$. As for the right side of Equation (4), recall that the set $\{\overline{T}_k(i), i = 1, \ldots, m\}$ forms an *independent version* of $\{T_k(i), i = 1, \ldots, m\}$. Hence, an alternative and useful way to describe the right side of Equation (4) is to consider a similar parallel system except for the common arrival; i.e., there are m independent renewal arrivals of groups, each comprising n stages, that are synchronized at the end (we refer to this system as a *parallel–independent* system). Now, let $\overline{T}_k$ be the sojourn time of project k in such a system. We have

$$\begin{aligned} 1 - \prod_{i=1}^{m} \left[p(\overline{T}_k(i) \le t)\right] &= 1 - p(\overline{T}_k(1) \le t) \cdot \ldots \cdot p(\overline{T}_k(m) \le t) \\ = \;\; 1 - p(\overline{T}_k(1) \le t, \ldots \overline{T}_k(m) \le t) &= 1 - p\left(\max_{1\le i\le m} \overline{T}_k(i) \le t\right) \\ = \;\; p\left(\max_{1\le i\le m} \overline{T}_k(i) > t\right) &= p(\overline{T}_k > t). \end{aligned} \tag{5}$$

Integrating all yields

$$p(T_k > t) \le p(\overline{T}_k > t), t \ge 0. \tag{6}$$

Conclusion 2. *The parallel system has a lower probability of a long sojourn time compared to that of the parallel-independent system; i.e., an initial synchronization stochastically reduces the sojourn time.*

Let $T_{m,n} = \lim_{k\to\infty} T_k$ and $\overline{T}_{m,n} = \lim_{k\to\infty} \overline{T}_k$ be the sojourn time in steady state of the parallel system and the parallel–independent system, respectively. Equation (6) immediately implies that $p(T_{m,n} > t) \leq p(\overline{T}_{m,n} > t)$ for $t \geq 0$. Consequently,

$$E(T_{m,n}) = \int_0^\infty p(T_{m,n} > t)dt \leq \int_0^\infty p(\overline{T}_{m,n} > t)dt = E(\overline{T}_{m,n}). \tag{7}$$

Conclusion 3. *The interdependency created as a result of the joint arrival (in other words, the initial synchronization) reduces the mean sojourn time of the parallel system compared to the mean sojourn time of the parallel–independent system.*

Remark 2. *Consider constant inter-arrival times, i.e., $v_k = a$ for some fixed $a, k = 1, 2, \ldots$ Using Claim 1, it is easy to verify that the set of waiting times $\{W_k(i,j), j = 1, \ldots, n\}$ is independent of $i = 1, \ldots, m$. As a result, $T_k(i) = \sum_{j=1}^{n}(W_k(i,j) + S_k(i,j)), i = 1, 2, \ldots, m$, and the sojourn times of the groups are also independent of i. The immediate conclusion is that constant inter-arrival times lead to the parallel–independent system.*

5. A Special Case: The Poisson Arrival Process

5.1. Lower and Upper Bounds

This section studies how the system performance is affected by the parameters; it focuses on the impact of the number of stages (n), the number of groups (m), and the servers' utilization (ρ) on the mean sojourn time. It assumes a Poisson arrival process with rate $\lambda_k = \lambda$, i.i.d. exponential service times with rate $\mu_k(i,j) = \mu$ and utilization $\rho = \lambda/\mu < 1$. Due to the complexity of performing an exact analysis, three bounds for the mean sojourn time are presented and then a numerical analysis is applied. Start by presenting two lower bounds (marked by subscript 1 and 2, respectively) and one upper bound (marked by subscript 3), as follows:

1. The first lower bound is obtained by neglecting operational queues and assuming an empty system for all arrivals (System 1). In this case, the sojourn time is the maximum of m i.i.d. services, each composed of n exponential stages; i.e., we have m i.i.d. r.v.s $T^1(i) \sim Erlang(n, \mu)$. Let T^1 be the sojourn time in this system. Then,

$$T^1 = \max_{1\leq i\leq m}\{T^1(i)\}$$
$$E\left(T^1\right) = \int_{t=0}^{\infty}\left[1 - \left(1 - \sum_{k=0}^{n-1}\frac{e^{-\mu t}(\mu t)^k}{k!}\right)^m\right]dt; \tag{8}$$

2. The second lower bound is obtained by assuming no splitting into m groups; i.e., $m = 1$ (System 2). Let T^2 be the sojourn time. It is well known that $T^2 \sim Erlang(n, \mu - \lambda)$, and, thus, $E(T^2) = \dfrac{n}{\mu - \lambda}$.
   Comparing Systems 1 and 2 (the lower bounds) to the parallel system highlights the different types of delays. The difference between the performance of System 1 (without operational queues) and the parallel system gives an estimate of the waiting time for a server. Moreover, the difference between the performance of System 2 (without splitting) and the parallel system gives an estimate of the impact of the final synchronization. Obviously, the quality of the bounds depends on the servers' utilization. When ρ is low, neglecting operational delays is acceptable and, thus, the first bound performs better. However, as ρ increases, the operational delays become significant and the additional delay due to synchronization is negligible; thus, the second bound is preferred;
3. An upper bound is obtained by assuming m independent arrival processes of groups (System 3). Applying Burke's theorem [43] and Conclusions 2 and 3, we obtain that

the sojourn times of group $i, i = 1, \ldots, m$, are i.i.d. r.v.s. with $T^3(i) \sim Erlang(n, \mu - \lambda)$ distribution. Let $T^3 = \max_{1 \le i \le m} \{T^3(i)\}$ (the total sojourn time). We obtain

$$E\left(T^3\right) = \int_{t=0}^{\infty} \left[1 - \left(1 - \sum_{k=0}^{n-1} \frac{e^{-(\mu-\lambda)t}[(\mu-\lambda)t]^k}{k!}\right)^m\right] dt. \tag{9}$$

Clearly, System 3 highlights the impact of the initial interdependency of the m groups. Note that when $\rho \to 0$ (i.e., $\lambda \to 0$), $E(T^3)$ and $E(T^1)$ converge to the same limit (intuitively, when the frequency of arrivals is low, there are almost no operational delays, so the effect of joint arrival is negligible, and we see a similar behavior in systems 1 and 3). Since $E(T^1) \le E(T_{m,n}) \le E(T^3)$, by the Sandwich theorem (Squeeze theorem), these two bounds become tight.

5.2. Asymptotic Analysis for Large n and m

For a large n, Kang and Serfozo [47] prove that

$$\begin{aligned} \lim_{m \longrightarrow \infty} \left(\frac{T^3 - \overline{b}(m,n)}{\overline{a}(m,n)}\right) &\longrightarrow \overrightarrow{X}, \\ \lim_{m \longrightarrow \infty} \left(\frac{T^1 - \underline{b}(m,n)}{\underline{a}(m,n)}\right) &\longrightarrow \underrightarrow{X}, \end{aligned} \tag{10}$$

where the r.v.s $\overrightarrow{X}$ and $\underrightarrow{X}$ are asymptotically Gumbel with normalizing constants

$$\begin{aligned} \overline{b}(m,n) &= (\mu - \lambda)^{-1}(\ln m + (n-1)\ln\ln m - \ln(n-1)!), \ \overline{a}(m,n) = \frac{1}{(\mu - \lambda)}, \\ \underline{b}(m,n) &= \mu^{-1}(\ln m + (n-1)\ln\ln m - \ln(n-1)!), \ \underline{a}(m,n) = \frac{1}{\mu}. \end{aligned} \tag{11}$$

The Gumbel distribution function represents the asymptotic limit distribution of the maximum among m exponentially distributed variables. Furthermore, it is well known that, for an r.v.X_k satisfying

$$\lim_{k \longrightarrow \infty} p\left(\frac{X_k - b_k}{a_k} \le x\right) = G(x), \tag{12}$$

it follows that, under some conditions (such as X_k obtains positive values only), its expectation also converges:

$$\lim_{k \longrightarrow \infty} (a_k)^{-r} E(X_k - b_k)^r = \int_0^{\infty} x^r dGx. \tag{13}$$

Equations (10) and (11) show that both bounds grow at a linear convergence rate in n, and a logarithmic convergence rate in m. Applying the Sandwich theorem (Squeeze theorem) implies that the mean sojourn time of the parallel system $E(T_{m,n})$ has the same growth rate:

$$\begin{aligned} \lim_{m \longrightarrow \infty} \frac{E(T_{m,n})}{\ln m} &\longrightarrow 1, \ \forall n, \\ \lim_{n \longrightarrow \infty} \frac{E(T_{m,n})}{n} &\longrightarrow 1, \ \forall m. \end{aligned} \tag{14}$$

Equation (14) says that $E(T_{m,n})$ grows at logarithmic rate $O(\ln\ m)$ and linear rate $O(n)$ for large m and n, respectively.

5.3. The Efficiency of the Bounds

While the bounds $E(T^3)$ and $E(T^1)$ are tight for a low utilization $\rho \longrightarrow 0$, it is necessary to estimate the efficiency of the bounds for other cases. To do so, this study uses the Maple 2022 software tool to numerically obtain $E(T_{m,n})$. Let $\lambda = 1$ and vary μ in $\{5, 2, 1.25, 1.052\}$ so that $\rho \in \{0.2, 0.5, 0.8, 0.95\}$. For each pair (λ, μ), let $n \in [2, 100]$ and $m \in [2, 7]$. In addition, the bounds $E(T^1)$ and $E(T^3)$ are derived for each pair (λ, μ) by applying (8) and (9), respectively. To study the efficiency of the bounds, the following measures have been suggested:

$$E_U = \frac{E(T_{m,n})}{E(T^3)}, E_L = \frac{E(T^1)}{E(T_{m,n})}.$$

Clearly, we have $0 < E_U, E_L \leq 1$. The ratio E_U captures the amount of time shortened due to initial synchronization. The ratio E_L captures the additional time accumulated due to the stochastic nature of the arrival process and the resulting queues. Figures 5 and 6 plot E_U and E_L as a function of m and n for $\rho = \{0.2, 0.5, 0.8, 0.95\}$ (the black circles, red squares, blue crosses, and green stars, respectively). Observations and insights are summarized in Conclusion 3.

However, we see that both ratios are certainly influenced by n, m, and ρ.

Conclusion 4. (i) *The effect the number of groups (splits) m. Increasing m increases the dependency between the groups due to the joint arrival, the difference between $E(T_{m,n})$ and $E(T^3)$ increases, and, thus, E_U decreases. On the other hand, as m increases, the synchronization delay at the final join node becomes significant compared to the operational delays for servers. As a result, the relative weight of those operational delays decreases and E_L increases. Despite this increase, Figure 5 shows that, when comparing E_U and E_L, the ratio E_U is more efficient for estimating the mean sojourn time, and, therefore, is more recommended;*

(ii) *The effect of the number of stages n. Clearly, as n increases, the effect of the joint arrival fades. This is particularly evident in E_U, which is increasing in n. Regarding E_L, we would expect to see a decrease in E_L, since increasing n further yields more servers and queues. However, Figure 6 and additional results (that are not reported here) imply that the changes in E_L are inconsistent, probably since increasing n adds variability that blurs the differences between the groups and yields inconsistency;*

(iii) *In fact, Figures 5 and 6 show that the changes in E_U and E_L are relatively low in m and n. This can be explained by the fact that both T^3 and T^1 grow in m and n at the same rate (in orders $O(\ln\ m)$ and $O(n)$, respectively) and, thus, also $E(T_{m,n})$ changes at the same rate (which is consistent with the results in Section 5.2);*

(iv) *The effect of the utilization ρ. The lower bound assumes only one project with no operational queues. Clearly, when ρ is low, the lower bound becomes tighter (see the black circles in Figure 6). However, when projects arrive more frequently, and ρ increases, E_L drops sharply. By contrast, the changes in E_U are inconsistent, and, although E_U slightly decreases in ρ, its efficiency is quite high for most values of ρ.*

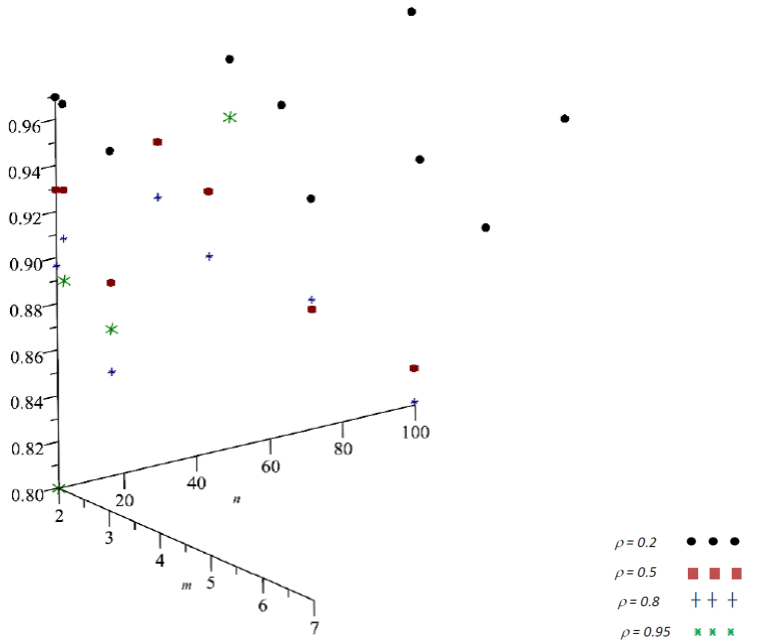


Figure 5. The ratio $E_U = \frac{E(T_{m,n})}{E(T^3)}$ as a function of m, n for $\rho \in \{0.2, 0.5, 0.8, 0.9\}$.

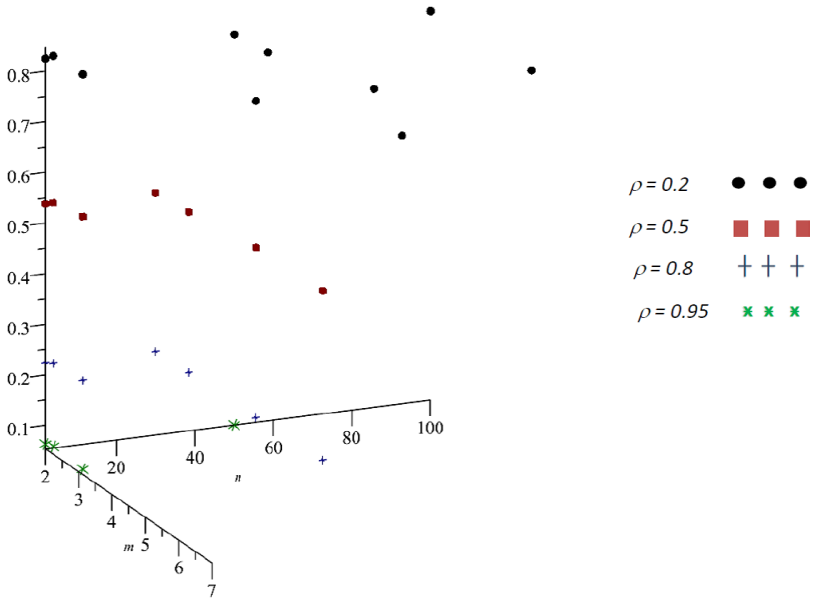


Figure 6. The ratio $E_L = \frac{E(T^1)}{E(T_{m,n})}$ as a function of m, n for $\rho \in \{0.2, 0.5, 0.8, 0.9\}$.

Overall, we may conclude that E_U is a good approximation, especially for large n. Table 1 summarizes the effect of increasing m, n, and ρ on the ratios E_U and E_L; " $\uparrow$ " and " $\downarrow$ " denote increasing and decreasing functions, respectively; " $\Downarrow$ " and " $\updownarrow$ " further denote a significant decrease and an inconsistent change, respectively.

Table 1. The efficiency of the bound as a function of m, n, and ρ.

The Efficiency of the Bound	$m \uparrow$	$n \uparrow$	$\rho \uparrow$
$E_U = \frac{E(T_{m,n})}{E(T^3)}$	$\downarrow$	$\uparrow$	$\downarrow$
$E_L = \frac{E(T^1)}{E(T_{m,n})}$	$\uparrow$	$\updownarrow$	$\Downarrow$

5.4. Simulation Study

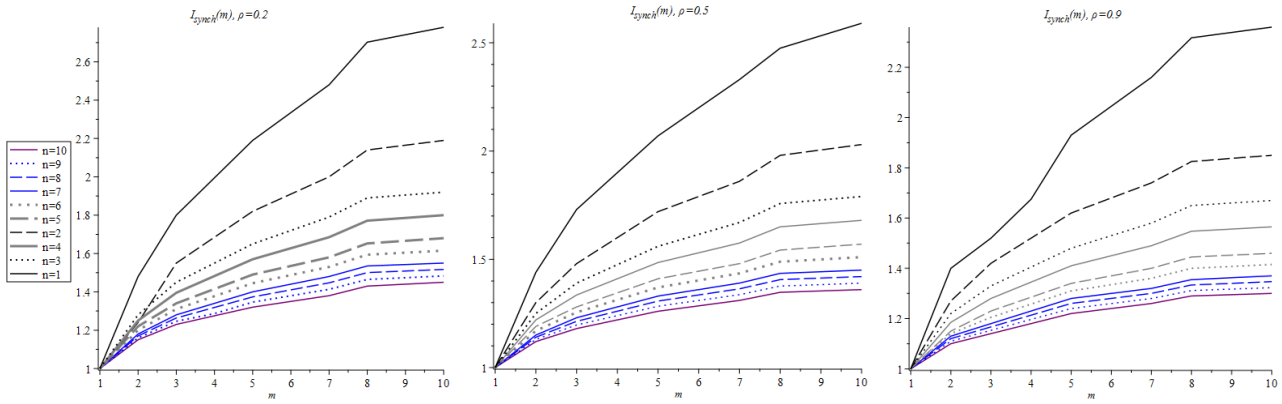
In this section, the aim is to investigate in depth the impact of the parameters (n, m, and ρ) on the parallel system's performance for relatively small n and m. To do so, a simulation study is used and, in each run, one of the parameters m, n, and ρ is varied while keeping the others fixed.

5.4.1. The Influence of the Synchronization Delay

Start by investigating the effect of the final synchronization on the mean sojourn time. First, fix n and ρ, and define the m-ratio $I_{synch}(m)$ as follows:

$$I_{synch}(m) = \frac{E(T_{m,n})}{E(T_{1,n})}. \tag{15}$$

The ratio $I_{synch}(m)$ compares the parallel system with $m \geq 1$ groups to a parallel system with a single group and no synchronization. In this way, $I_{synch}(m)$ captures the effect of parallelism intensified by the final synchronization and subsequent overhead. Clearly, $I_{synch}(m) \geq 1$ with $I_{synch}(m = 1) = 1$. Table 2 presents $I_{synch}(m)$ where m and n vary in $\{1, \ldots, 10\}$ and ρ varies in $\{0.2, 0.4, 0.5, 0.8, 0.9\}$. Figure 7a–c plots $I_{synch}(m)$ as a function of m for $n \in \{1, \ldots, 10\}$ and $\rho = 0.2, 0.5$, and 0.9, respectively.



(a) $\rho = 0.2$ (b) $\rho = 0.5$ (c) $\rho = 0.9$

Figure 7. The ratio $I_{synch}(m)$ as a function of m for $n = \{1, \ldots, 10\}$ and different values of ρ.

Conclusion 5. (i) *Figure 7a–c shows that $I_{synch}(m)$ increases logarithmically in m; this increase is consistent with the results of Section 5.2 for large m;*

(ii) *Increasing n decreases $I_{synch}(m)$ for fixed m, and decreases the growth rate of $I_{synch}(m)$ in m. This can be explained by the fact that increasing n increases the number of servers and the operational delays. In this case, the time required for the final synchronization takes a relatively small fraction of the total sojourn time. As a result, the difference between $E(T_{m,n})$ and $E(T_{1,n})$ is reduced, and $I_{synch}(m)$ decreases. We can further add that increasing n causes the system to be more deterministic, since each group behaves as an $Erlang(n, \mu - \lambda)$ distributed r.v. with coefficient of variation $c.v. = 1/\sqrt{n}$, which is decreasing in n. Thus, the final synchronization has less effect. Turning to the edge case, consider that $n \longrightarrow \infty$. In this case, the total sojourn time at each group is deterministic and equal to the sojourn time of other groups; statistically, there is no difference between the groups, and, thus, $I_{synch}(m) \longrightarrow 1$;*

(iii) *Comparing Figure 7a–c shows that $I_{synch}(m)$ is slightly decreasing in ρ. As discussed above, as the system becomes overloaded, more time is spent in waiting for servers rather than in synchronization. However, contrary to expectation, the results show that $I_{synch}(m)$ is hardly affected by ρ, and the changes are quite negligible. Here, too, consider the edge case $\rho \longrightarrow 0$, where almost no operational delays exist but only the final synchronization. In this case, the differences between the systems are significant, and $I_{synch}(m)$ increases to its maximum value.*

Table 2. $I_{synch}(m)$, $m, n \in \{1, \ldots, 10\}$, $\rho = \{0.2, 0.4, 0.5, 0.8, 0.9\}$.

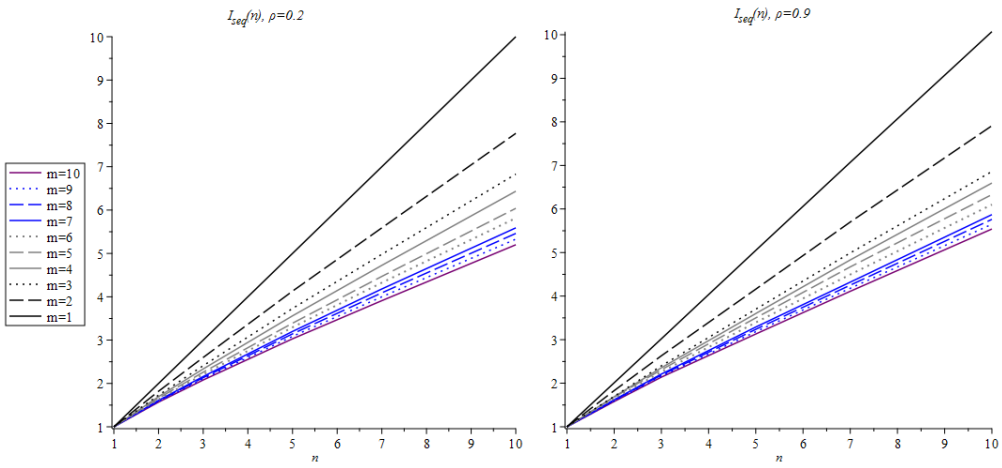
$I_{synch}(m)$	n	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$	$m=9$	$m=10$
	1	1.4800	1.8000	1.9950	2.1900	2.3350	2.4800	2.7025	2.7413	2.7800
	2	1.3400	1.5500	1.6850	1.8200	1.9100	2.0000	2.1400	2.1650	2.1900
	3	1.2800	1.4500	1.5500	1.6500	1.7200	1.7900	1.8900	1.9050	1.9200
	4	1.2500	1.3950	1.4825	1.5700	1.6275	1.6850	1.7713	1.7856	1.8000
$\rho = 0.2$	**5**	1.2200	1.3400	1.4150	1.4900	1.5350	1.5800	1.6525	1.6663	1.6800
	6	1.2000	1.3100	1.3775	1.4450	1.4875	1.5300	1.5938	1.6044	1.6150
	7	1.1800	1.2800	1.3400	1.4000	1.4400	1.4800	1.5350	1.5425	1.5500
	8	1.1701	1.2635	1.3186	1.3736	1.4103	1.4470	1.5004	1.5087	1.5170
	9	1.1599	1.2465	1.2965	1.3464	1.3797	1.4130	1.4647	1.4738	1.4830
	10	1.1500	1.2300	1.2750	1.3200	1.3500	1.3800	1.4300	1.4400	1.4500
	1	1.4500	1.7400	1.9300	2.1200	2.2450	2.3700	2.5775	2.6188	2.6600
	2	1.3200	1.5100	1.6300	1.7500	1.8300	1.9100	2.0300	2.0500	2.0700
	3	1.2500	1.4100	1.5000	1.5900	1.6450	1.7000	1.7925	1.8113	1.8300
	4	1.2150	1.3550	1.4300	1.5050	1.5550	1.6050	1.6850	1.7000	1.7150
$\rho = 0.4$	**5**	1.1800	1.3000	1.3600	1.4200	1.4650	1.5100	1.5775	1.5888	1.6000
	6	1.1700	1.2700	1.3275	1.3850	1.4225	1.4600	1.5188	1.5294	1.5400
	7	1.1600	1.2400	1.2950	1.3500	1.3800	1.4100	1.4600	1.4700	1.4800
	8	1.1468	1.2235	1.2752	1.3269	1.3553	1.3836	1.4295	1.4382	1.4470
	9	1.1332	1.2065	1.2548	1.3031	1.3298	1.3564	1.3980	1.4055	1.4130
	10	1.1200	1.1900	1.2350	1.2800	1.3050	1.3300	1.3675	1.3738	1.3800
	1	1.4400	1.7300	1.9000	2.0700	2.2000	2.3300	2.4750	2.5325	2.5900
	2	1.3000	1.4800	1.6000	1.7200	1.7900	1.8600	1.9800	2.0050	2.0300
	3	1.2500	1.3900	1.4750	1.5600	1.6150	1.6700	1.7575	1.7738	1.7900
	4	1.2200	1.3350	1.4100	1.4850	1.5300	1.5750	1.6500	1.6650	1.6800
$\rho = 0.5$	**5**	1.1900	1.2800	1.3450	1.4100	1.4450	1.4800	1.5425	1.5563	1.5700
	6	1.1700	1.2550	1.3125	1.3700	1.4025	1.4350	1.4888	1.4994	1.5100
	7	1.1500	1.2300	1.2800	1.3300	1.3600	1.3900	1.4350	1.4425	1.4500
	8	1.1401	1.2135	1.2602	1.3069	1.3353	1.3636	1.4061	1.4132	1.4203
	9	1.1299	1.1965	1.2398	1.2831	1.3098	1.3364	1.3764	1.3830	1.3897
	10	1.1200	1.1800	1.2200	1.2600	1.2850	1.3100	1.3475	1.3538	1.3600
	1	1.4100	1.6800	1.8200	1.9600	2.0900	2.2200	2.3950	2.4175	2.4400
	2	1.2700	1.4400	1.4400	1.4400	1.6000	1.7600	1.9100	1.9050	1.9000
	3	1.2100	1.3500	1.3500	1.3500	1.4750	1.6000	1.7225	1.7213	1.7200
	4	1.1800	1.3050	1.3050	1.3050	1.4075	1.5100	1.6088	1.6069	1.6050
$\rho = 0.8$	**5**	1.1500	1.2600	1.2600	1.2600	1.3400	1.4200	1.4950	1.4925	1.4900
	6	1.1350	1.2300	1.2300	1.2300	1.3025	1.3750	1.4438	1.4419	1.4400
	7	1.1200	1.2000	1.2000	1.2000	1.2650	1.3300	1.3925	1.3913	1.3900
	8	1.1134	1.1868	1.1868	1.1868	1.2485	1.3102	1.3661	1.3632	1.3603
	9	1.1066	1.1732	1.1732	1.1732	1.2315	1.2898	1.3389	1.3343	1.3297
	10	1.1000	1.1600	1.1600	1.1600	1.2150	1.2700	1.3125	1.3063	1.3000
	1	1.4000	1.4200	1.6750	1.9300	2.0450	2.1600	2.3175	2.3388	2.3600
	2	1.2700	1.4200	1.5200	1.6200	1.6800	1.7400	1.8250	1.8375	1.8500
	3	1.2200	1.3300	1.4050	1.4800	1.5300	1.5800	1.6500	1.6600	1.6700
	4	1.1850	1.2800	1.3450	1.4100	1.4500	1.4900	1.5475	1.5563	1.5650
$\rho = 0.9$	**5**	1.1500	1.2300	1.2850	1.3400	1.3700	1.4000	1.4450	1.4525	1.4600
	6	1.1400	1.2050	1.2575	1.3100	1.3350	1.3600	1.4000	1.4075	1.4150
	7	1.1300	1.1800	1.2300	1.2800	1.3000	1.3200	1.3550	1.3625	1.3700
	8	1.1201	1.1668	1.2135	1.2602	1.2802	1.3002	1.3336	1.3402	1.3469
	9	1.1099	1.1532	1.1965	1.2398	1.2598	1.2798	1.3115	1.3173	1.3231
	10	1.1000	1.1400	1.1800	1.2200	1.2400	1.2600	1.2900	1.2950	1.3000

5.4.2. The Influence of Multiple Stages

Next, the influence of the operational delays is studied, characterized by the number of stages n. Fix m and ρ and define the n-ratio $I_{\mathtt{seq}}(n)$ to be

$$I_{\mathtt{seq}}(n) = \frac{E(T_{m,n})}{E(T_{m,1})}. \tag{16}$$

The ratio $I_{\mathtt{seq}}(n)$ compares the parallel system with n stages to a parallel system with only one stage (the basic split–join network; see Figure 1). In this way, $I_{\mathtt{seq}}(n)$ captures the effect of operational delays intensified by the serial servers and queues; clearly, $I_{\mathtt{seq}}(n) \geq 1$ and $I_{\mathtt{seq}}(n = 1) = 1$. Table 3 tabulates $I_{\mathtt{seq}}(n)$, where n and m vary in $\{2, \ldots, 10\}$ and ρ varies in $\{0.2, 0.4, 0.5, 0.8, 0.9\}$. Figure 8a,b plots $I_{\mathtt{seq}}(n)$ as a function of n for $m \in \{1, \ldots, 10\}$ and $\rho = 0.2$ and 0.9, respectively.



(a) $\rho = 0.2$ (b) $\rho = 0.9$

Figure 8. The ratio $I_{\mathtt{seq}}(n)$ as a function of n for $m \in \{1, \ldots, 10\}$ and $\rho = 0.2$ and 0.9.

Conclusion 6. (i) *Figure 8a,b shows a statistically significant linear growth in $I_{\mathtt{seq}}(n)$ as a function of n. Consistent with the results of Section 5.2 for large n, the mean sojourn time increases proportionally with the number of added stages;*

(ii) *Increasing m both decreases $I_{\mathtt{seq}}(n)$ for fixed n and decreases the growth rate of $I_{\mathtt{seq}}(n)$ in n. Obviously, the reason lies in the fact that increasing m decreases the relative weight of the operational delays compared to the synchronization delay, and so $I_{\mathtt{seq}}(n)$ decreases;*

(iii) *We further see that $I_{\mathtt{seq}}(n)$ is hardly affected by ρ. This can be explained by the fact that changes in the utilization have a similar effect on all servers. Therefore, there is a negligible dependence between ρ and $I_{\mathtt{seq}}(n)$.*

Summarizing Conclusions 4 and 5, we see that parallelism (i.e., synchronization delays) has a logarithmic effect $O(\ln m)$ on the mean sojourn time, while seriality (i.e., operational delays) has a linear effect $O(n)$. However, these two effects slightly decrease as n and m increase, respectively. This highlights the interplay between the relative weights of the different delays. Accordingly, changes in the utilization have an effect mostly on the synchronization time, and hardly at all on the waiting times for servers; generally speaking, the effects of parallelism and seriality are (almost) independent in the utilization.

Table 3. $I_{\mathtt{seq}}(n)$, $m, n \in \{1, \ldots, 10\}$, $\rho = \{0.2, 0.4, 0.5, 0.8, 0.9\}$.

$I_{seq}(n)$	m	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$	$n=10$
	1	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	10.0000
	2	1.8200	2.6000	3.3650	4.1300	4.8600	5.5900	6.3167	7.0433	7.7700
	3	1.7300	2.4100	3.0700	3.7300	4.3550	4.9800	5.5967	6.2133	6.8300
	4	1.6950	2.3350	2.9475	3.5600	4.1425	4.7250	5.2950	5.8650	6.4350
$\rho = 0.2$	**5**	1.6600	2.2600	2.8250	3.3900	3.9300	4.4700	4.9933	5.5167	6.0400
	6	1.6350	2.2100	2.7525	3.2950	3.8100	4.3250	4.8217	5.3183	5.8150
	7	1.6100	2.1600	2.6800	3.2000	3.6900	4.1800	4.6500	5.1200	5.5900
	8	1.5968	2.1336	2.6388	3.1439	3.6174	4.0909	4.5477	5.0045	5.4613
	9	1.5832	2.1064	2.5963	3.0861	3.5426	3.9991	4.4423	4.8855	5.3287
	10	1.5700	2.0800	2.5550	3.0300	3.4700	3.9100	4.3400	4.7700	5.2000
	1	2.0000	3.0100	4.0150	5.0200	6.0200	7.0200	8.0200	9.0200	10.0200
	2	1.8200	2.5900	3.3300	4.0700	4.8300	5.5900	6.3100	7.0300	7.7500
	3	1.7300	2.4300	3.0850	3.7400	4.3750	5.0100	5.6267	6.2433	6.8600
	4	1.6900	2.3400	2.9450	3.5500	4.1400	4.7300	5.3017	5.8733	6.4450
$\rho = 0.4$	**5**	1.6500	2.2500	2.8050	3.3600	3.9050	4.4500	4.9767	5.5033	6.0300
	6	1.6300	2.2000	2.7375	3.2750	3.7950	4.3150	4.8167	5.3183	5.8200
	7	1.6100	2.1500	2.6700	3.1900	3.6850	4.1800	4.6567	5.1333	5.6100
	8	1.5935	2.1236	2.6288	3.1339	3.6124	4.0909	4.5544	5.0178	5.4813
	9	1.5765	2.0964	2.5863	3.0761	3.5376	3.9991	4.4490	4.8988	5.3487
	10	1.5600	2.0700	2.5450	3.0200	3.4650	3.9100	4.3467	4.7833	5.2200
	1	2.0000	3.0000	4.0050	5.0100	6.0100	7.0100	8.0133	9.0167	10.0200
	2	1.8100	2.6000	3.3600	4.1200	4.8550	5.5900	6.3233	7.0567	7.7900
	3	1.7100	2.4000	3.0550	3.7100	4.3400	4.9700	5.5867	6.2033	6.8200
	4	1.6900	2.3300	2.9450	3.5600	4.1450	4.7300	5.3050	5.8800	6.4550
$\rho = 0.5$	**5**	1.6700	2.2600	2.8350	3.4100	3.9500	4.4900	5.0233	5.5567	6.0900
	6	1.6350	2.2100	2.7550	3.3000	3.8175	4.3350	4.8450	5.3550	5.8650
	7	1.6000	2.1600	2.6750	3.1900	3.6850	4.1800	4.6667	5.1533	5.6400
	8	1.5901	2.1336	2.6354	3.1372	3.6190	4.1008	4.5710	5.0411	5.5113
	9	1.5799	2.1064	2.5946	3.0828	3.5510	4.0192	4.4724	4.9255	5.3787
	10	1.5700	2.0800	2.5550	3.0300	3.4850	3.9400	4.3767	4.8133	5.2500
	1	2.0100	3.0100	4.0200	5.0300	6.0300	7.0300	8.0367	9.0433	10.0500
	2	1.8100	2.6000	3.3550	4.1100	4.8650	5.6200	6.3700	7.1200	7.8700
	3	1.7300	2.4100	3.0950	3.7800	4.4000	5.0200	5.6767	6.3333	6.9900
	4	1.7050	2.3500	2.9825	3.6150	4.2150	4.8150	5.4250	6.0350	6.6450
$\rho = 0.8$	**5**	1.6800	2.2900	2.8700	3.4500	4.0300	4.6100	5.1733	5.7367	6.3000
	6	1.6350	2.2300	2.7825	3.3350	3.8775	4.4200	4.9583	5.4967	6.0350
	7	1.5900	2.1700	2.6950	3.2200	3.7250	4.2300	4.7433	5.2567	5.7700
	8	1.5834	2.1502	2.6620	3.1738	3.6623	4.1508	4.6454	5.1401	5.6347
	9	1.5766	2.1298	2.6280	3.1262	3.5977	4.0692	4.5446	5.0199	5.4953
	10	1.5700	2.1100	2.5950	3.0800	3.5350	3.9900	4.4467	4.9033	5.3600
	1	2.0100	3.0200	4.0350	5.0500	6.0600	7.0700	8.0700	9.0700	10.0700
	2	1.8300	2.6300	3.3950	4.1600	4.9300	5.7000	6.4367	7.1733	7.9100
	3	1.7000	2.4000	3.0500	3.7000	4.3450	4.9900	5.6133	6.2367	6.8600
	4	1.6900	2.3550	2.9775	3.6000	4.2150	4.8300	5.4183	6.0067	6.5950
$\rho = 0.9$	**5**	1.6800	2.3100	2.9050	3.5000	4.0850	4.6700	5.2233	5.7767	6.3300
	6	1.6500	2.2600	2.8275	3.3950	3.9450	4.4950	5.0300	5.5650	6.1000
	7	1.6200	2.2100	2.7500	3.2900	3.8050	4.3200	4.8367	5.3533	5.8700
	8	1.6068	2.1869	2.7121	3.2372	3.7440	4.2507	4.7542	5.2576	5.7611
	9	1.5932	2.1631	2.6730	3.1828	3.6811	4.1793	4.6692	5.1590	5.6489
	10	1.5800	2.1400	2.6350	3.1300	3.6200	4.1100	4.5867	5.0633	5.5400

6. The Serial Split–Join System

This section studies a version of the parallel split–join system called the *serial* system. The parallel processing of tasks gives rise to synchronization constraints that may cause project delays. The trade-off between increasing the number of servers engaged in parallel processing at the expense of synchronization delays plays an important role in choosing the structure of the network, especially when the network must process different types of projects or servers. To mitigate this trade-off, *serial* system is introduced. As in the parallel system, a project arrives and splits into m parallel groups, each of which is composed of n stages. However, after each stage j, all groups must be synchronized before the next stage, $j+1$, begins. The project exits the system when all groups in the n-th stage are completed. A typical serial system is presented in Figure 9. The serial system highlights the effect of synchronization. Here, there are $n \times m$ operational queues, and n synchronization queues where the groups must wait to be joined at each stage.

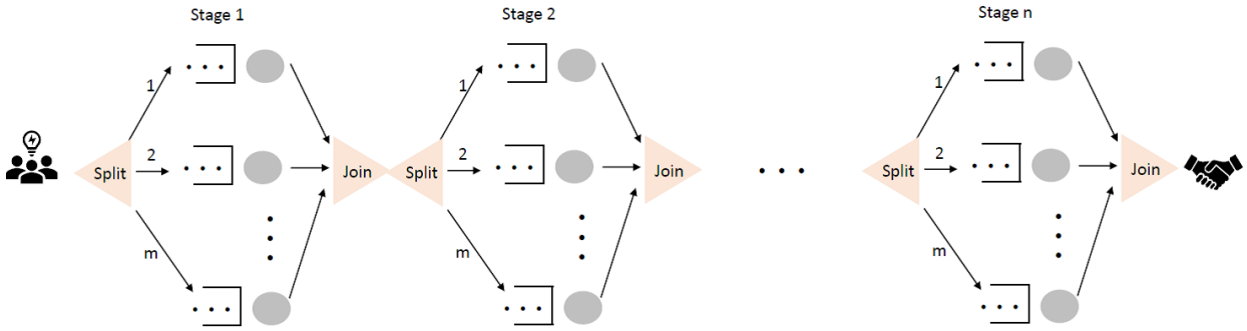


Figure 9. A typical serial system.

In practice, there is a growing interest in serial systems. One example of such a system is a medical process in an emergency room, where some of the initial tests (stage 1) can be administered simultaneously (for example, while a blood sample is being analyzed, a CT scan can be performed). However, the patient cannot be discharged until all these stage-1 tests are completed. After the results are analyzed, the patient continues to the stage-2 analysis, etc. In manufacturing systems, a maintenance procedure for a product requires parallel integrity checks that can only be assessed after receiving the results of previous tests. In supply chains, an order for a product requires several items simultaneously from vendors, where multiple parts are produced in parallel and then assembled into the product [29]. Other examples derive from the increase in multiprocessing technology and parallel programming in computer and telecommunications networks. For example, there are grid systems that divide applications into parallel and synchronized tasks [48], buffer size optimization systems for data transmission, and memory constraints in computer networks [9]. In real systems, it may happen that, although parallelism shortens time, the ensuing synchronization delays may play a significant role and affect managers' decisions.

For background, the serial split–join network has hardly been investigated in the literature; it is only mentioned in the work of Ko [29], who considers a two-stage serial split–join network with $m = 2$ splits and a Poisson arrival process. However, only an approximation of the mean sojourn time of the network is derived.

The serial system at time $t \geq 0$ is characterized by an n-row vector $\mathbb{L}(t) = (L_1(t), L_2(t), \ldots, L_n(t))$, where $L_j(t)$ is an m-column vector $L_j(t) = (L_{1,j}(t), L_{2,j}(t), \ldots, L_{m,j}(t))^T$. The value $L_{i,j}(t)$ indicates the number of waiting tasks at station (i,j) (i.e., in the queue and at the server). Due to the FCFS policy, all tasks of project k are held in the same stage (either in the queue, waiting to be served, or at the join node after being served). For simplicity, denote each subsequent join node at stage $j = 1, .., n$ with the index j, leading to have the next corollary.

Corollary 2. *Let $N_{(i,j),S}(t)$ be the number of groups i waiting at join node j, $j = 1, \ldots, n$ at time t. It is easy to verify that $N_{(i,j),S}(t)$ satisfies*

$$N_{(i,j),S}(t) = \underset{1 \le i \le m}{Max} \{L_{i,j}(t)\} - L_{i,j}(t). \tag{17}$$

Example 2. *Let $m = 3, n = 3$, and assume that project 6 arrives to a system with $\mathbb{L}(t) = ((1,0,2)^T, (2,0,1)^T, (1,0,1)^T)$. In this case, groups 1 and 3 of project 6 join the queues, and group 2 enters the server immediately. As a result, we obtain $\mathbb{L}(t^+) = ((2,1,3)^T, (2,0,1)^T, (1,0,1)^T)$ (see Figure 10). We also see that*

$$\underset{i=1,2,3}{Max} \{L_{i,1}(t^+)\} = 3, \quad \underset{i=1,2,3}{Max} \{L_{i,2}(t^+)\} = 2, \quad \underset{i=1,2,3}{Max} \{L_{i,3}(t^+)\} = 1. \tag{18}$$

The number of waiting groups at the join nodes are $N_{(1,1),S}(t^+) = 1$, $N_{(2,1),S}(t^+) = 2$, $N_{(3,1),S}(t^+) = 0$ (at join node 1), $N_{(1,2),S}(t^+) = 0$, $N_{(2,2),S}(t^+) = 2$, $N_{(3,2),S}(t^+) = 1$ (at join node 2), and $N_{(1,2),S}(t^+) = 0, N_{(2,3)S}(t^+) = 1, N_{(3,3),S}(t^+) = 0$ (at join node 3). Examples of possible transitions are, e.g., the state $L_1(t) = (2,1,3)^T$ with rate $\mu_{1,1}(5)$ is changed to $L_1'(t^+) = (1,1,3)^T$, and the states $L_1(t) = (2,1,3)^T$ and $L_2(t) = (2,0,1)^T$ with rate $\mu_{3,1}(4)$ are changed to $L_1'(t^+) = (2,1,2)^T$ and $L_2'(t^+) = (3,1,2)^T$, respectively (here, tasks $(3,1)$ of project 4 is completed, so project 4 finishes stage 1, and continues to stage 2; as a result, both $L_1(t)$ and $L_2(t)$ are changed).

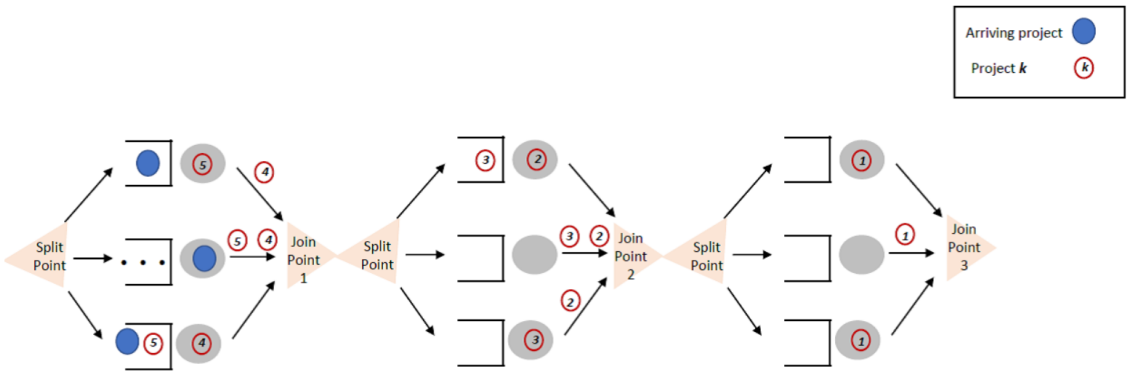


Figure 10. A snapshot of the serial multi-join system upon arrival of project 6.

As discussed above, the mathematical analysis of the serial systems is complicated, even for $m = 3$, and, thus, a numerical analysis is performed. The aim is to investigate the impact of the additional time due to synchronization delays as a function of the system parameters. To do so, the results of the serial system are numerically compared to those of the parallel system with the same parameters.

6.1. The Influence of Synchronization Overhead and Stages

Denote by $E(\widetilde{T}_{m,n})$ the mean sojourn time of the serial system with m groups and n stages. Similar to the ratios defined in (15) and (16), define respectively the m-ratio $\widetilde{I}_{synch}(m)$ and the n-ratio $\widetilde{I}_{\mathtt{seq}}(n)$ as follows:

$$\widetilde{I}_{synch}(m) = \frac{E\left(\widetilde{T}_{m,n}\right)}{E\left(\widetilde{T}_{1,n}\right)}, \widetilde{I}_{\mathtt{seq}}(n) = \frac{E\left(\widetilde{T}_{m,n}\right)}{E\left(\widetilde{T}_{m,1}\right)}. \tag{19}$$

Clearly, $\widetilde{I}_{synch}(m) \ge 1$ and $\widetilde{I}_{\mathtt{seq}}(n) \ge 1$ (equality holds when $m = 1$ and $n = 1$, respectively). The ratio $\widetilde{I}_{synch}(m)$ denotes the effect of synchronization overhead that can be attributed to parallelism. The ratio $\widetilde{I}_{\mathtt{seq}}(n)$ denotes the effect of seriality that contributes to resource (server) delays and synchronization overhead simultaneously.

Tables 4 and 5 tabulate $\widetilde{I}_{synch}(m)$ and $\widetilde{I}_{\mathtt{seq}}(n)$ where m, n vary in $\{1, \ldots, 10\}$, respectively, and $\rho = \{0.2, 0.4, 0.5, 0.8, 0.9\}$. For $\rho = 0.5$, Figures 11 and 12 illustrate $\widetilde{I}_{synch}(m)$ and $\widetilde{I}_{\mathtt{seq}}(n)$ for $n \in \{1, \ldots, 10\}$ and $m \in \{1, \ldots, 10\}$, respectively.

Table 4. The ratio $\widetilde{I}_{synch}$ for $n, m = \{1, \ldots, 10\}$, $\rho \in \{0.2, 0.4, 0.5, 0.8, 0.9\}$.

$\widetilde{I}_{synch}(m)$		m				
ρ	n	**2**	**3**	**5**	**7**	**10**
0.2	**1**	1.48	1.80	2.19	2.48	2.78
	2	1.47	1.78	2.19	2.46	2.76
	3	1.47	1.77	2.18	2.45	2.75
	5	1.47	1.77	2.17	2.44	2.73
	7	1.46	1.76	2.16	2.43	2.72
	10	1.46	1.76	2.16	2.42	2.71
0.4	**1**	1.45	1.74	2.12	2.37	2.66
	2	1.44	1.72	2.09	2.34	2.61
	3	1.44	1.71	2.08	2.32	2.58
	5	1.43	1.70	2.06	2.30	2.55
	7	1.43	1.70	2.05	2.29	2.54
	10	1.42	1.69	2.04	2.28	2.53
0.5	**1**	1.44	1.73	2.07	2.33	2.59
	2	1.43	1.70	2.05	2.28	2.54
	3	1.42	1.68	2.03	2.26	2.51
	5	1.41	1.67	2.01	2.24	2.48
	7	1.41	1.67	2.00	2.23	2.46
	10	1.40	1.66	1.99	2.21	2.45
0.8	**1**	1.41	1.68	1.96	2.22	2.44
	2	1.38	1.62	1.92	2.13	2.35
	3	1.38	1.62	1.92	2.11	2.33
	5	1.37	1.59	1.88	2.08	2.29
	7	1.37	1.59	1.88	2.07	2.27
	10	1.36	1.58	1.87	2.06	2.26
0.9	**1**	1.40	1.68	1.93	2.16	2.36
	2	1.36	1.59	1.90	2.10	2.32
	3	1.37	1.58	1.88	2.07	2.39
	5	1.34	1.57	1.86	2.04	2.24
	7	1.34	1.56	1.85	2.02	2.22
	10	1.34	1.56	1.83	2.01	2.22

Conclusion 7. (i) *Figure 11 shows a statistically significant logarithmic growth rate of* $\widetilde{I}_{synch}(m)$ *in m (the test shows that* $R^2 > 95\%$*). Similarly, Figure 12 shows a statistically significant linear growth rate of* $\widetilde{I}_{\mathtt{seq}}(n)$ *in n (the test shows that* $R^2 > 99\%$*);*

(ii) *We see that* $\widetilde{I}_{synch}(m)$ *is hardly affected by n when m is fixed, and, similarly,* $\widetilde{I}_{\mathtt{seq}}(n)$ *is hardly affected by m when n is fixed. This can be explained by the synchronization constraints needed at the end of each stage (which is the beginning of the next stage). Specifically, the variability obtained between the different groups does not accumulate, and each stage is almost identical to the first stage. This absence of dispersion aggregation causes* $\widetilde{I}_{synch}(m)$ *to be almost independent in n, and* $\widetilde{I}_{\mathtt{seq}}(n) \approx n$*;*

(iii) *Tables 4 and 5 show that, as in the parallel system,* $\widetilde{I}_{synch}(m)$ *is slightly decreasing in* ρ*, and* $\widetilde{I}_{\mathtt{seq}}(n)$ *is hardly affected by* ρ*; see Conclusions* $4(iii)$ *and* $5(iii)$*.*

To summarize, we see that the effects of m, n, and ρ are similar in both the serial system and the parallel system. However, and contrary to expectation, the rate of change in the serial system is slower and may even be negligible. Thus, it seems that the serial system is significantly less sensitive to marginal changes and sometimes even indifferent.

Table 5. The ratio $\widetilde{I}_{\mathtt{seq}}$ for $n, m = \{1, \ldots, 10\}$, $\rho \in \{0.2, 0.4, 0.5, 0.8, 0.9\}$.

$\tilde{I}_{seq}(n)$		n				
ρ	**m**	**2**	**3**	**5**	**7**	**10**
0.2	**1**	2.00	3.00	5.00	7.00	10.00
	2	1.99	2.98	4.97	6.94	9.90
	3	1.98	2.96	4.92	6.87	9.79
	5	1.99	2.98	4.94	6.90	9.83
	7	1.99	2.97	4.92	6.87	9.79
	10	1.99	2.96	4.91	6.85	9.76
0.4	**1**	2.00	3.01	5.02	7.02	10.02
	2	1.99	2.97	4.93	6.89	9.82
	3	1.98	2.96	4.90	6.84	9.74
	5	1.97	2.94	4.86	6.77	9.63
	7	1.97	2.94	4.85	6.76	9.76
	10	1.97	2.92	4.83	6.72	9.55
0.5	**1**	2.00	3.00	5.01	7.01	10.02
	2	1.99	2.96	4.91	6.86	9.78
	3	1.96	2.92	4.83	6.75	9.60
	5	1.98	2.94	4.87	6.77	9.62
	7	1.96	2.92	4.81	6.70	9.51
	10	1.96	2.91	4.79	6.67	9.47
0.8	**1**	2.01	3.01	5.03	7.03	10.05
	2	1.97	2.94	4.90	6.82	9.72
	3	1.95	2.90	4.78	6.67	9.48
	5	1.98	2.94	4.85	6.74	9.59
	7	1.93	2.86	4.72	6.57	9.32
	10	1.94	2.87	4.73	6.55	9.31
0.9	**1**	2.01	3.02	5.05	7.07	10.07
	2	1.96	2.95	4.84	6.79	9.67
	3	1.91	2.85	4.75	6.59	9.40
	5	1.97	2.94	4.85	6.76	9.55
	7	1.96	2.90	4.78	6.63	9.39
	10	1.98	3.06	4.81	6.66	9.50

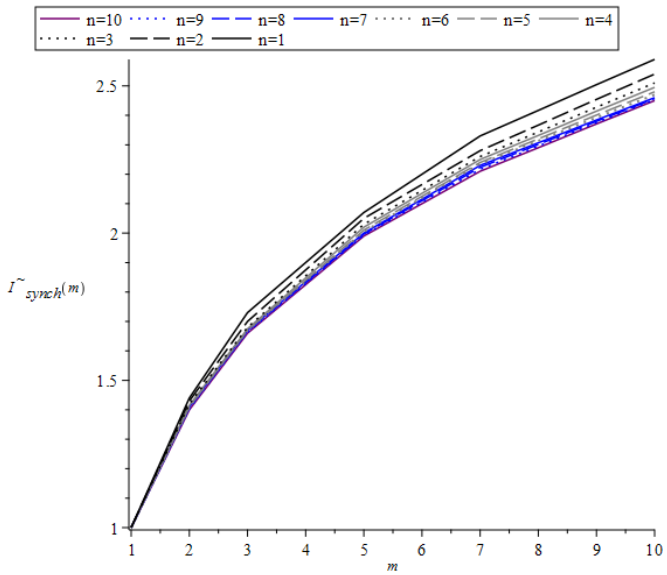


Figure 11. $\widetilde{I}_{synch}(m)$ for $n = \{1, \ldots, 10\}$, $\rho = 0.5$.

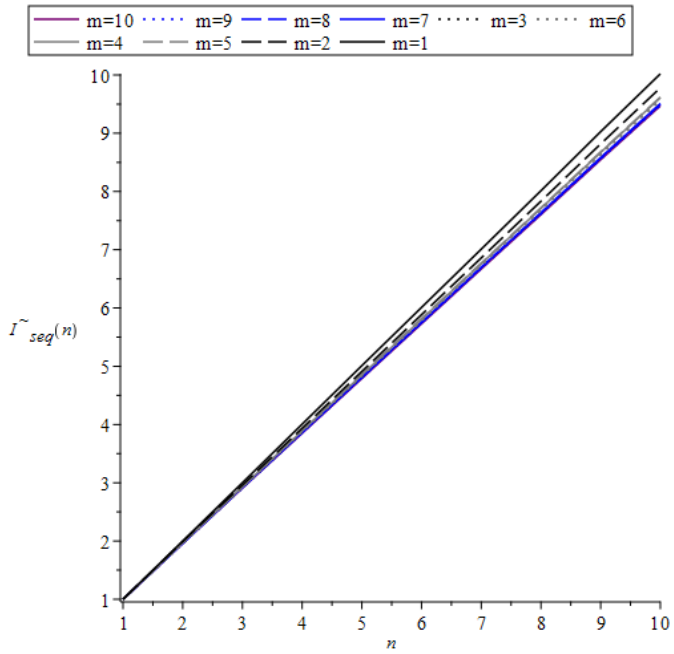


Figure 12. $\widetilde{I}_{\mathsf{seq}}(n)$ for $m = \{1, \ldots, 10\}$, $\rho = 0.5$.

6.2. Comparison of the Systems

In the previous section, we studied the performance of the serial system as a function of the different parameters. Here, to complete our investigation, we explore in greater depth the influence of multiple synchronization constraints on the project duration. To do so, we compare the mean sojourn time of the two systems with the same parameters. Obviously, due to the additional synchronizations, the time in the serial system $E(\widetilde{T}_{m,n})$ is greater than that of corresponding parallel system $E(T_{m,n})$ (the equality holds when $m = 1$ or $n = 1$). Accordingly, it is interesting to study how significant the differences are in relation to the parameters. The insights gained from this study can be used in practice when more synchronization constraints are required for a project or, vice versa, when synchronization constraints are no longer necessary and can be eliminated. Let $I_{S/P}$ be the ratio of the mean sojourn times:

$$I_{S/P} = \frac{E(\widetilde{T}_{m,n})}{E(T_{m,n})}. \tag{20}$$

The values $I_{S/P}$ are summarized in Table 6 for $n, m \in \{2, \ldots, 10\}$ and $\rho \in \{0.2, 0.4, 0.5, 0.8, 0.9\}$; the cases $m = 1$ or $n = 1$, where $I_{S/P} = 1$, are omitted.

Table 6 implies that the ratio $I_{S/P}$ is increasing in n and m, and decreasing in ρ. This is more formally stated in Conclusion 7. In addition, Figures 13 and 14 plot $I_{S/P}$ as a function of m and n for $\rho = 0.5$, respectively. We see that $I_{S/P}$ grows logarithmically in m, and, surprisingly, also logarithmically in n. Checking this growth rate for other utilizations leads to the same conclusion. For example, the blue, black, and purple surfaces of Figure 15 show $I_{S/P}$ as a function of n and m for $\rho = 0.2$, 0.5 and 0.9.

Table 6. The ratio $I_{S/P}$ for $m, n \in \{2, \ldots, 10\}$, $\rho \in \{0.2, 0.4, 0.5, 0.8, 0.9\}$.

ρ	$I_{S/P}$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
0.2		1.0960	1.1466	1.1751	1.2036	1.2227	1.2418	1.2529	1.2642	1.2752
0.4		1.0910	1.1461	1.1786	1.2110	1.2218	1.2325	1.2439	1.2554	1.2667
0.5	$m = 2$	1.0961	1.1390	1.1649	1.1908	1.2091	1.2274	1.2368	1.2464	1.2557
0.8		1.0856	1.1340	1.1624	1.1907	1.2023	1.2139	1.2208	1.2278	1.2347
0.9		1.0714	1.1214	1.1416	1.1617	1.1761	1.1905	1.2011	1.2119	1.2224
0.2		1.1447	1.2244	1.2720	1.3195	1.3491	1.3787	1.3969	1.4152	1.4332
0.4		1.1431	1.2175	1.2637	1.3099	1.3366	1.3633	1.3819	1.4006	1.419
0.5	$m = 3$	1.1468	1.2148	1.2581	1.3013	1.3297	1.3581	1.3747	1.3914	1.4078
0.8		1.1266	1.2020	1.2340	1.2659	1.2977	1.3294	1.3384	1.3476	1.3565
0.9		1.1237	1.1898	1.2363	1.2827	1.3018	1.3208	1.3372	1.3537	1.3699
0.2		1.1721	1.2695	1.3290	1.3885	1.4251	1.4618	1.4847	1.5079	1.53065
0.4		1.1685	1.2609	1.3192	1.3775	1.4102	1.4430	1.4645	1.4863	1.5076
0.5	$m = 4$	1.1680	1.2581	1.3108	1.3636	1.3984	1.4332	1.4531	1.4733	1.493
0.8		1.1516	1.2448	1.2905	1.3362	1.3665	1.3969	1.4110	1.4253	1.43935
0.9		1.1490	1.2299	1.2819	1.3340	1.3588	1.3836	1.4021	1.4208	1.4391
0.2		1.1995	1.3145	1.3860	1.4575	1.5012	1.5448	1.5726	1.6006	1.6281
0.4		1.1938	1.3042	1.3746	1.4450	1.4839	1.5227	1.5472	1.5719	1.5962
0.5	$m = 5$	1.1892	1.3014	1.3636	1.4258	1.4671	1.5083	1.5316	1.5551	1.5782
0.8		1.1766	1.2875	1.3470	1.4064	1.4354	1.4643	1.4836	1.5031	1.5222
0.9		1.1743	1.2700	1.3276	1.3852	1.4158	1.4463	1.4670	1.4878	1.5083
0.2		1.2157	1.3430	1.4208	1.4986	1.5468	1.5950	1.6266	1.6584	1.68965
0.4		1.2092	1.3335	1.4078	1.4820	1.5261	1.5702	1.5982	1.6266	1.6544
0.5	$m = 6$	1.2099	1.3268	1.3971	1.4674	1.5113	1.5552	1.5810	1.6071	1.6326
0.8		1.1947	1.3023	1.3697	1.4372	1.4731	1.5091	1.5291	1.5492	1.56895
0.9		1.1918	1.2901	1.3551	1.4200	1.4554	1.4908	1.5115	1.5325	1.5531
0.2		1.2319	1.3715	1.4556	1.5397	1.5925	1.6452	1.6805	1.7162	1.7512
0.4		1.2245	1.3628	1.4409	1.5190	1.5683	1.6176	1.6493	1.6813	1.7126
0.5	$m = 7$	1.2305	1.3521	1.4305	1.5089	1.5555	1.6021	1.6304	1.6590	1.687
0.8		1.2128	1.3170	1.3925	1.4679	1.5109	1.5539	1.5745	1.5953	1.6157
0.9		1.2092	1.3102	1.3825	1.4548	1.4950	1.5352	1.5561	1.5772	1.5979
0.2		1.2423	1.3898	1.4787	1.5676	1.6240	1.6805	1.7180	1.7558	1.7929
0.4		1.2360	1.3783	1.4620	1.5456	1.5981	1.6505	1.6843	1.7183	1.7517
0.5	$m = 8$	1.2370	1.3680	1.4506	1.5332	1.5831	1.6329	1.6639	1.6952	1.7259
0.8		1.2202	1.3313	1.4110	1.4907	1.5364	1.5822	1.6069	1.6318	1.6563
0.9		1.2244	1.3498	1.4156	1.4814	1.5222	1.5630	1.5876	1.6124	1.6368
0.2		1.2529	1.4084	1.5021	1.5958	1.6559	1.7161	1.7558	1.7958	1.835
0.4		1.2477	1.3939	1.4832	1.5725	1.6282	1.6838	1.7196	1.7558	1.7911
0.5	$m = 9$	1.2436	1.3840	1.4709	1.5577	1.6109	1.6640	1.6978	1.7318	1.7652
0.8		1.2277	1.3457	1.4297	1.5137	1.5622	1.6107	1.6396	1.6687	1.6973
0.9		1.2398	1.3897	1.4490	1.5083	1.5497	1.5911	1.6194	1.6480	1.676
0.2		1.2632	1.4265	1.5250	1.6234	1.6872	1.7510	1.7928	1.8350	1.8764
0.4		1.2591	1.4092	1.5041	1.5989	1.6577	1.7164	1.7542	1.7924	1.8299
0.5	$m = 10$	1.2500	1.3997	1.4908	1.5818	1.6382	1.6945	1.7309	1.7677	1.8038
0.8		1.2351	1.3598	1.4481	1.5363	1.5875	1.6387	1.6716	1.7049	1.7375
0.9		1.2548	1.4289	1.4818	1.5346	1.5766	1.6186	1.6506	1.6829	1.7146

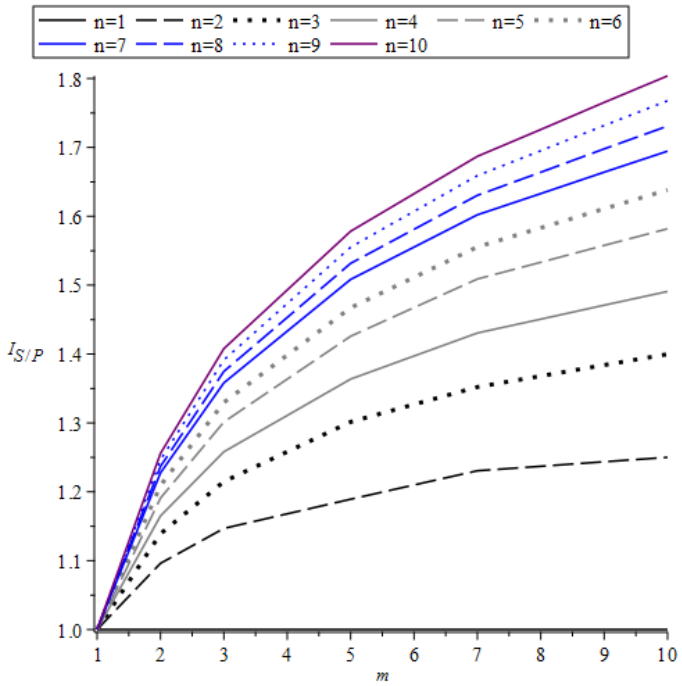


Figure 13. $I_{S/P}(m)$ for $n = \{1, \ldots, 10\}$, $\rho = 0.5$.

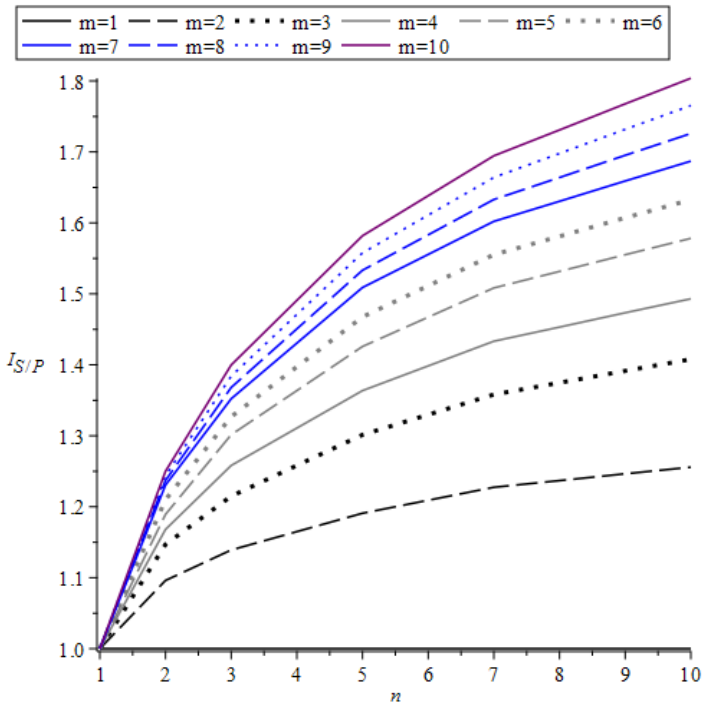


Figure 14. $I_{S/P}(n)$ for $m = \{1, \ldots, 10\}$, $\rho = 0.5$.

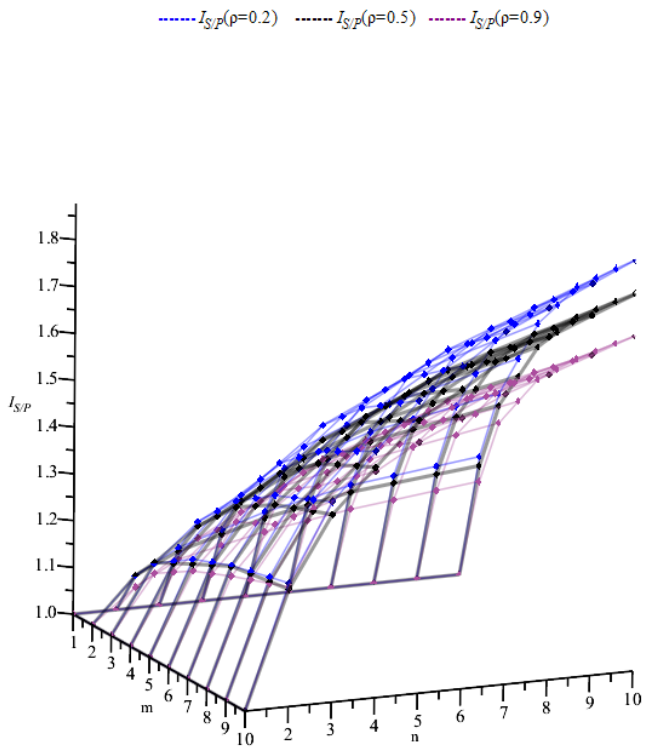


Figure 15. The ratio $I_{S/P}(m,n)$ for $\rho = \{0.2, 0.5, 0.9\}$.

Conclusion 8. (i) *The impact of* n *and* m. *Figures 13–15 show that* $I_{S/P}$ *increases in both* n *and* m *at a logarithmic rate. Its logarithmic growth rate in* m *is consistent with the previous conclusions. However, its logarithmic growth rate in* n *is quite surprising, and contrary to the expectation of a linear growth rate. This can be explained as follows. The addition of stages adds operational delays in a relatively similar way to both systems, but it adds synchronization delays only to the serial system (since the parallel system has only one final join node, independently of* n*). Thus, these delays intensify the logarithmic component of the growth rate and offset the other components.*
Furthermore, this observation leads to an unexpected result. Since m *and* m *have the same logarithmic effect on the growth rate, the ratio* $I_{S/P}$ *shows a kind of duality for a fixed* ρ*:*

$$I_{S/P}(m,n) \approx I_{S/P}(n,m). \tag{21}$$

Equation (21) implies that the added time in the serial system beyond that of the parallel system is relatively similar due to parallelism (i.e., increasing m*) or due to seriality (i.e., increasing* n*). For example, Figure 16 shows* $I_{S/P}(m,n)$ *and its dual-value* $I_{S/P}(n,m)$ *(solid and dashed curves, respectively) for the pairs* $(m,n) = \{(7,3),(10,3),(8,5)\}$ *as a function of* ρ*. We see that* $I_{S/P}(7,3) \approx I_{S/P}(3,7)$ *(blue curves),* $I_{S/P}(10,3) \approx I_{S/P}(3,10)$ *(black curves), and* $I_{S/P}(8,5) \approx I_{S/P}(5,8)$ *(gray curves);*

(ii) *The impact of* ρ. *In most cases, increasing* ρ *decreases* $I_{S/P}$ *for fixed* n *and* m. *The explanation is simple: for a low utilization, the operational queues are almost empty and, thus, synchronization times (which exist mainly in the serial system) constitute the key share of the total time. Here, the serial system is significantly slower than the parallel system and, thus,* $I_{S/P}$ *increases. However, for an overloaded system (with a high utilization), the operational delays increase in both systems, thereby offsetting the synchronization delays that slightly decrease* $I_{S/P}$*. In summary, the effect of efficiency is significant mainly for high* n *and* m *and for a low utilization. In this case, the large number of synchronizations becomes a major component in the time profile.*

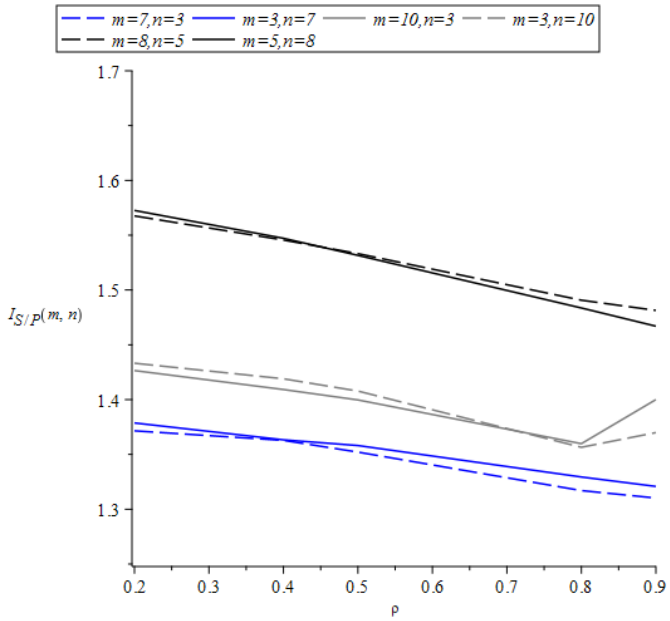


Figure 16. The duality property $I_{S/P}(m,n) \approx I_{S/P}(n,m)$ for $(m,n) = \{(7,3), (10,3), (8,5)\}$.

7. Concluding Remarks and Future Research

The present paper studies the influence of multiple stages and multiple synchronizations in expanded split–join/fork–join networks. Two different architectures are introduced, the parallel structure and the serial one. Since the mathematical analysis of such systems is difficult and mostly impossible, a sensitivity analysis is obtained to evaluate the efficiency of various bounds for the mean sojourn time. For each system, an extensive numerical study is performed to investigate the impact of (i) the number of stages n, (ii) the number of groups m, and (iii) the utilization ρ on the mean sojourn time. Numerical results show that the mean sojourn time of both systems is hardly effected by ρ, and increases linearly in n and logarithmically in m, although the serial system is significantly less sensitive to changes in m and n. If we attribute the number of stages to operational delays due to slow servers or limited resources, and the number of groups to synchronization delays, then the limited resources have a linear influence, while the synchronization gap has a logarithmic influence. Accordingly, it is advisable to invest extensively in limited resource in order to improve the performance of the system.

Furthermore, comparing the two systems with the same parameters surprisingly shows a logarithmic rate effect of both parallelism (the synchronization delays) and seriality (operational delays). Thus, we obtain a kind of *duality* property for the ratio between the two systems. This *duality* can be used for estimation purposes in designing and optimizing the systems when more synchronization constraints are required or, alternatively, when fewer synchronization constraints are required.

For future research, it would be interesting to investigate mixed split–join systems that include both parallel and serial structures. In real life, the need to synchronize may be a probabilistic decision. Thus, assigning a probability to each join node (which is actually a combination of the serial and parallel systems) would be a promising practical stream of research. Moreover, this paper assumed an exponential service time and a Poisson arrival process. It would be interesting to generalize this work by including other distributions for service times or studying more general arrival processes. In this case, however, the author believes that it would be highly difficult to analyze these systems using only mathematical tools and, thus, the use of numerical analysis would probably be essential.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. The Proof of Claim 1

Proof. Claim 1 is obtained by a double induction on the number of projects k and the number of stages j in group i.

Step-1. Substituting $j = 1$ in (1) yields

$$W_{k+1}(i,1) = [T_k(i,l) - \nu_{k+1}]^+ = [W_k(i,1) + S_k(i,1) - \nu_{k+1}]^+, \tag{A1}$$

which is equal to the original Lindley equation with one task (i.e., $m = 1$), and, thus, holds for all k.

Induction-Step. Assume that Claim 1 holds for j; it is now proven for $j+1$.

Step-2. Use induction on k.

Step-1. Let $k = 1$.

$$W_1(i,j+1) = \left[\sum_{l=1}^{j+1} T_0(i,l) - \sum_{l=1}^{j} T_1(i,l) - \nu_1\right]^+ = 0$$

Clearly for the first project, the system is empty.

Induction-Step. Assume that Claim 1 holds for k; prove it for $k+1$ (recall that we assume $j+1$).

Step-2. By the induction, $W_k(i,1), W_k(i,2), \ldots, W_k(i,j+1)$ are the waiting times of project k in group i. Since $S_k(i,1), S_k(i,2), \ldots, S_k(i,j+1)$ are the service times, we have

$$\sum_{l=1}^{j+1} T_k(i,l) = W_k(i,1) + S_k(i,1) + \ldots + W_k(i,j+1) + S_k(i,j+1)$$

to determine the time project k finishes task $j+1$. Similarly, $\sum_{l=1}^{j} T_{k+1}(i,l)$ is the time that project $k+1$ finishes task j. Project $k+1$ starts ν_{k+1} units of time after project k. Thus, if $\sum_{l=1}^{j+1} T_k(i,l) < \sum_{l=1}^{j} T_{k+1}(i,l) + \nu_{k+1}$, the station $j+1$ is empty when project $k+1$ arrives; otherwise, project $k+1$ will wait, i.e.,

$$W_{k+1}(i,j) = \left[\sum_{l=1}^{j} T_k(i,l) - \sum_{l=1}^{j-i} T_{k+1}(i,l) - \nu_{k+1}\right]^+.$$

□

Appendix B. The Proof of Claim 2(1)

Proof. The proof is obtained by using a double induction on the number of projects k and the number of stages j in group i.

Step-1. Assume $j = 1$. Use induction on k.

Step-1. $k = 1$. Clearly, $W_1(i,1) = 0, \forall i$. The service times $\{S_1(i,1)\}, i = 1, \ldots, m$ are independent r.v.s; thus, by property **P1**, they are associated r.v.s. Since ν_1 is an independent r.v., by **P2**, the r.v.s in the set

$$\{W_1(i,1), S_1(i,1), -t_1, i = 1, \ldots, m\}$$

are associated.

Induction-Step. Assume that

$$\{W_l(i,1), S_l(i,1), -t_l, i = 1,\ldots,m, l = 1,\ldots,k\}$$

forms a set of associated r.v.s. Now, prove for $k+1$.

Step-2. The proof is obtained immediately, by applying Appendix A of Nelson and Tantawi [30], and Chapter 4.2.III of Baccelli and Makowski [23].

Induction-Step. Assume $\{W_l(i,r), S_l(i,r), -t_l, r = 1,\ldots,j, l = 1,\ldots,k\}$ forms a set of associated r.v.s. Prove for $j+1$.

Step-2. Assume $j+1$. Use induction on k. □

Step-1. $k = 1$. For the first project, $W_1(i,j+1) = 0$; thus, applying P1 and P2, the r.v.s in set

$$\{W_1(i,r), S_1(i,r), -t_1, W_1(i,j+1), S_1(i,j+1), r = 1,\ldots,j\}$$

are associated.

Induction-Step. Assume that the r.v.s in set

$$\{W_l(i,r), S_l(i,r), -t_l, W_l(i,j+1), S_l(i,j+1), r = 1,\ldots,j, l = 1,\ldots,k\}$$

are associated. Prove for $k+1$.

Step-2. By Claim 1 we have

$$W_{k+1}(i,j+1) = \left[\sum_{l=1}^{j+1} T_k(i,l) - \sum_{l=1}^{j} T_{k+1}(i,l) - \nu_{k+1}\right]^+$$
$$= \left[\sum_{l=1}^{j+1} W_k(i,l) + S_k(i,l) - \sum_{l=1}^{j}(W_{k+1}(i,l) + S_{k+1}(i,l)) - \nu_{k+1}\right]^+.$$

The function $[x]^+$ is a non-decreasing monotonic function. Thus, by the induction assumption and applying **P4**, the elements of set $\{W_{k+1}(i,l), l = 1,\ldots,j+1\}$ are also associated. Furthermore, the set $\{S_{k+1}(i,l), l = 1,\ldots,j+1\}$ contains independent r.v.s, and, thus, by **P1**, are associated. Finally, applying **P2**, the elements in set

$$\{W_l(i,r), S_l(i,r), -t_l, W_l(i,j+1), S_l(i,j+1), r = 1,\ldots,j+1, l = 1,\ldots,k+1\}$$

are associated.

References

1. Alesawi, S.; Ghanem, S. Overcome heterogeneity impact in modeled fork-join queuing networks for tail prediction. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), IEEE, Honolulu, HI, USA, 18–21 February 2019; pp. 270–275.
2. Gorbunova, A.; Vishnevsky, V. The analysis of big data centers performance. *Adv. Syst. Sci. Appl.* **2022**, *22*, 70–83.
3. Nguyen, M.; Alesawi, S.; Li, N.; Che, H.; Jiang, H. A black-box fork-join latency prediction model for data-intensive applications. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1983–2000. [CrossRef]
4. Ardagna, D.; Bernardi, S.; Gianniti, E.; Karimian Aliabadi, S.; Perez-Palacin, D.; Requeno, J.I. Modeling performance of hadoop applications: A journey from queueing networks to stochastic well formed nets. In Proceedings of the Algorithms and Architectures for Parallel Processing: 16th International Conference, ICA3PP 2016, Granada, Spain, 14–16 December 2016; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; Proceedings 15, pp. 599–613.
5. Delias, P.; Lagopoulos, A.; Tsoumakas, G.; Grigori, D. Using multi-target feature evaluation to discover factors that affect business process behavior. *Comput. Ind.* **2018**, *99*, 253–261. [CrossRef]
6. Sethuraman, S. *Analysis of Fork-Join Systems: Network of Queues with Precedence Constraints*; CRC Press: Boca Raton, FL, USA, 2022.
7. Enganti, P.; Rosenkrantz, T.; Sun, L.; Wang, Z.; Che, H.; Jiang, H. ForkMV: Mean-and-variance estimation of fork-join queuing networks for datacenter applications. In Proceedings of the 2022 IEEE International Conference on Networking, Architecture and Storage (NAS), IEEE, Philadelphia, PA, USA, 3–4 October 2022; pp. 1–8.
8. Wang, W.; Harchol-Balter, M.; Jiang, H.; Scheller-Wolf, A.; Srikant, R. Delay asymptotics and bounds for multi-task parallel jobs. *ACM Sigmetrics Perform. Eval. Rev.* **2019**, *46*, 2–7. [CrossRef]

9. Ding, S. Multi-Class Fork-Join Queues & The Stochastic Knapsack Problem. Ph.D. Thesis, Universiteit Leiden, Amsterdam, The Netherlands, 2011.
10. Krishnamurthy, A.; Suri, R. Performance analysis of single stage kanban controlled production systems using parametric decomposition. *Queueing Syst.* **2006**, *54*, 141–162. [CrossRef]
11. Shaaban, S.; Romero-Silva, R. Performance of merging lines with uneven buffer capacity allocation: The effects of unreliability under different inventory-related costs. *Cent. Eur. J. Oper. Res.* **2021**, *29*, 1253–1288. [CrossRef]
12. Matta, A.; Dallery, Y.; Di Mascolo, M. Analysis of assembly systems controlled with kanbans. *Eur. J. Oper. Res.* **2005**, *166*, 310–336. [CrossRef]
13. Raghavan, N.S.; Viswanadham, N. Generalized queueing network analysis of integrated supply chains. *Int. J. Prod. Res.* **2001**, *39*, 205–224.
14. Atar, R.; Mandelbaum, A.; Zviran, A. Control of fork-join networks in heavy traffic. In Proceedings of the 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, Monticello, IL, USA, 1–5 October 2012; pp. 823–830.
15. Özkan, E. Control of fork-join processing networks with multiple job types and parallel shared resources. *Math. Oper. Res.* **2022**, *47*, 1310–1334. [CrossRef]
16. Prabhakar, B.; Bambos, N.; Mountford, T.S. The synchronization of Poisson processes and queueing networks with service and synchronization nodes. *Adv. Appl. Probab.* **2000**, *32*, 824–843. [CrossRef]
17. Ramakrishnan, R.; Krishnamurthy, A. Analysis of kitting Operations in manufacturing systems. *Asia Pac. J. Oper. Res.* **2008**, *25*, 187–216. [CrossRef]
18. Ramakrishnan, R.; Krishnamurthy, A. Performance evaluation of a synchronization station with multiple inputs and population constraints. *Comput. Oper. Res.* **2012**, *39*, 560–570. [CrossRef]
19. Schol, D.; Vlasiou, M.; Zwart, B. Large fork-join networks with nearly deterministic service times. *Mathematics* **2019**, arXiv:1912.11661. [CrossRef]
20. Roy, D.; van Ommeren, J.K.; de Koster, R.; Gharehgozli, A. Modeling landside container terminal queues: Exact analysis and approximations. *Transp. Res. Part B Methodol.* **2022**, *162*, 73–102. [CrossRef]
21. Towsley, D.; Rommel, C.G.; Stankovic, J.A. Analysis of fork-join program response times on multiprocessors. *IEEE Trans. Parallel Distrib. Syst.* **1990**, *1*, 286–303. [CrossRef] [PubMed]
22. Baccelli, F.; Makowski, A.M. Simple Computable Bounds for the Fork-Join Queue. Ph.D. Thesis, INRIA, Le Chesnay-Rocquencourt, France 1985.
23. Baccelli, F.; Makowski, A.M. Queueing models for systems with synchronization constraints. *Proc. IEEE* **1989**, *77*, 138–161. [CrossRef]
24. Baccelli, F.; Makowski, A.M.; Shwartz, A. The fork-join queue and related systems with synchronization constrains: Stochastic Ordering and Computable Bounds. *Adv. Appl. Probab.* **1989**, *21*, 629–660. [CrossRef]
25. Baccelli, F.; Massey, W.A.; Towsley, D. Acyclic fork join queuing networks. *J. Assoc. Comput. Machanics* **1989**, *36*, 615–642. [CrossRef]
26. Ko, S.S.; Serfozo, R.F. Response times in M/M/s fork-join networks. *Adv. Appl. Probab.* **2004**, *36*, 854–871. [CrossRef]
27. Ko, S.S.; Serfozo, R.F. Sojourn Times in G/M/1 fork-join networks. *Nav. Res. Logist.* **2008**, *55*, 432–443. [CrossRef]
28. Varki, E. Mean value technique for closed fork-join networks. *Perform. Eval. Rev.* **1999**, *27*, 103–112. [CrossRef]
29. Ko, S.S. Cycle times in a serial fork-join network. In Proceedings of the Computational Science and Its Applications–ICCSA 2007: International Conference, Kuala Lumpur, Malaysia, 26–29 August 2007; Springer: Berlin/Heidelberg, Germany, 2007; Proceedings, Part I 7, pp. 758–766.
30. Nelson, R.; Tantawi, A.N. Approximation analysis of Fork/Join synchronization in parallel queues. *IEEE Trans. Comput.* **1988**, *37*, 739–743. [CrossRef]
31. Nelson, R.; Towsley, D.; Tantawi, A.N. Performance analysis of parallel processing systems. *IEEE Trans. Softw. Eng.* **1988**, *14*, 532–540. [CrossRef]
32. Fiorini, P.M. Analytic approximations of fork-join queues. In Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE, Warsaw, Poland, 24–26 September 2015; Volume 2, pp. 966–971.
33. Lebrecht, A.S.; Knottenbelt, W.J. Response time approximations in fork-join queues. In Proceedings of the 23rd Annual UK Performance Engineering Workshop (UKPEW 2007), ORMS, Kirk, UK, 31 July 2007.
34. Kemper, B.; Mandjes, M. Mean sojourn times in two-queue fork-join systems: Bounds and approximations. *OR Spectr.* **2012**, *34*, 723–742. [CrossRef]
35. Takahashi, M.; Osawa, H.; Fujisawa, T. On a synchronization queue with two finite buffers. *Queueing Syst.* **2000**, *36*, 107–123. [CrossRef]
36. Qiu, Z.; Pérez, J.F.; Harrison, P.G. Beyond the mean in fork-join queues: Efficient approximation for response-time tails. *Perform. Eval.* **2015**, *91*, 99–116. [CrossRef]
37. Varma, S.; Makowski, A.M. Interpolation approximations for symmetric fork-join queues. *J. Perform. Eval.* **1994**, *20*, 245–265. [CrossRef]

38. Tan, X.; Knessl, C. A fork-join queueuing model:diffusion approximation, integral representations and asymptotics. *Queueing Syst.* **1996**, *22*, 287–332. [CrossRef]
39. Knessl, C. A diffusion model for two parallel queues with processor sharing: Transient behavior and asymptotics. *J. Appl. Math. Stoch. Anal.* **1999**, *12*, 311–338. [CrossRef]
40. Kushner, H.J. *Heavy Traffic Analysis of Controlled Queueing and Communication Networks*; Springer: New York, NY, USA, 2001.
41. Zeng, Y.; Tan, J.; Xia, C.H. Fork and join queueing networks with heavy tails: Scaling dimension and throughput limit. *J. ACM (JACM)* **2021**, *68*, 1–30. [CrossRef]
42. Meijer, M.S.; Schol, D.; van Jaarsveld, W.; Vlasiou, M.; Zwart, B. Extreme-value theory for large fork-join queues, with an application to high-tech supply chains. *arXiv* **2021**. arXiv:2105.09189.
43. Burke, P.J. The output process of a stationary M/M/s queueing system. *Ann. Math. Stat.* **1968**, *39*, 1144–1152. [CrossRef]
44. Walrand, J. *An Introduction to Queueing Networks*; Caopter 4; Prentice Hall: Hoboken, NJ, USA, 1988.
45. Latouche, G.; Ramaswami, V. *Introduction to Matrix Analytic Methods in Stochastic Modeling*; SIAM: Philadelphia, PA, USA, 1999.
46. Lindley, D.V. The theory of queues with a single server. In *Mathematical Proceedings of the Cambridge Philosophical Society*; Cambridge University Press: Cambridge, UK, 1952; Volume 48, pp. 277–289.
47. Kang, S.; Serfozo, R.F. Extreme values of phase-type and mixed random variables with parallel-processing examples. *J. Appl. Probab.* **1999**, *36*, 194–210. [CrossRef]
48. Cremonesi, P.; Turrin, R.; Alexandrov, V.N. Modeling the effects of node heterogeneity on the performance of grid applications. *J. Netw.* **2009**, *4*, 837–854. [CrossRef]