Probable Approximate Coordination

Ariel Livshits
Yahoo! Research, Haifa, Israel

Yoram Moses
Technion, Haifa, Israel

Abstract

We study the problem of how to coordinate the actions of independent agents in a distributed system where message arrival times are unbounded, but are determined by an exponential probability distribution. Asynchronous protocols executed in such a model are guaranteed to succeed with probability 1. We demonstrate a case in which the best asynchronous protocol can be improved on significantly. Specifically, we focus on the task of performing actions by different agents in a linear temporal order – a problem known in the literature as *Ordered Response*. In asynchronous systems, ensuring such an ordering requires the construction of a message chain that passes through each acting agent, in order. Solving *Ordered Response* in this way in our model will terminate in time that grows linearly in the number of participating agents n, in expectation. We show that relaxing the specification slightly allows for a significant saving in time. Namely, if *Ordered Response* should be guaranteed with high probability (arbitrarily close to 1), it is possible to significantly shorten the expected execution time of the protocol. We present two protocols that adhere to the relaxed specification. One of our protocols executes exponentially faster than a message chain, when the number of participating agents n is large, while the other is roughly quadratically faster. For small values of n, it is also possible to achieve similar results by using a hybrid protocol.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed coordination, ordered response, exponentially distributed delay

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.19

Related Version *Extended Version*: `https://arxiv.org/pdf/2311.05368.pdf` [16]

Funding This work was supported in part by the Israel Science Foundation under grant 2061/19.

Acknowledgements Yoram Moses is the Israel Pollak academic chair at the Technion. The authors would like to thank Yonathan Shadmi for his most valuable help.

1 Introduction

Numerous applications of distributed systems rely on promptly responding to spontaneously occurring events triggered by the environment. These events can range from the activation of fire alarms or smoke detectors to transactions involving bank account deposits or withdrawals. In order to ensure that the system behaves correctly, it may be necessary to execute one or more relevant actions. The sequential execution of these actions often holds significance, particularly when dealing with financial transactions. Typically, a diverse range of such actions, including condition testing and related updates, must be carried out to successfully complete these transactions.

This paper considers solutions to a distributed coordination problem called *Ordered Response* (*OR*). In *Ordered Response*, agents must perform individual actions in a particular linear order, in response to the spontaneous arrival of an external input to the system. For example, let there be a system with three agents i_1, i_2 and i_3. Let t_1, t_2 and t_3 be the points in time at which the agents perform their respective actions. Then it must hold that $t_1 \leq t_2 \leq t_3 < \infty$ in every run of a protocol that solves *OR*. Notice that if the agents act simultaneously, then the specifications of *OR* are satisfied.

The *Ordered Response* problem has been analyzed previously in the contexts of synchronous and asynchronous message-passing communication models [3, 5]. However, in many modern day multi-agent systems, message delays are subject to probabilistic bounds, i.e., determined according to a probabilistic distribution. Examples can be seen in wireless sensor networks (WSNs). A WSN is a distributed multi-agent system that is often composed of a large quantity of disposable sensors that communicate using wireless broadcast. Messages sent in this network experience random delays due to random changes in the environment. However, it was observed in [1] that a single server M/M/1 queue [12] can represent the cumulative link delay in a WSN, when the random delays are modeled as exponential random variables.

In this work, we investigate the *Ordered Response* problem in a model where message arrival times are unbounded, but are sampled from an exponential probability distribution.[1] We are particularly interested in protocols that achieve correctness with high probability (w.h.p.) and terminate in a short expected time. It is unclear, based on initial observations, whether the previous solutions are the most optimal for this particular task.

One of the conclusions that can be drawn from previous work done by Chandy & Misra [5] is that ensuring a linear ordering of actions at distinct sites of an asynchronous system necessitates the existence of a message chain among the coordinating agents. Namely, in order for agents to ensure that their respective actions are performed in a linear temporal order, a message chain visiting each of the sites in this order must be formed. Concretely, every asynchronous protocol that solves *Ordered Response* must construct a message chain among the agents. Consequently, in an asynchronous system of n agents, *Ordered Response* can only be solved in time that is linear in n, since a chain of at least $n-1$ messages must be sent.

Since message delays are unbounded, our probabilistic model appears to be closer to an asynchronous model, than a synchronous model. However, the assumption of exponentially distributed delays may be leveraged to significantly shorten the expected time of termination. Intuitively, since they naturally induce a distribution on the runs of a given protocol, it is possible to choose appropriate waiting times for the agents such that correct runs are more likely to happen. For example, say an agent broadcasts a message to 99 other agents. Assuming that arrival times are independent and identically distributed exponential random variables with parameter 1 $[sec^{-1}]$, then, with probability 0.999, all messages will arrive within roughly 11.52 seconds. So, assuming that every agent has access to a global clock and that the broadcast happened at time $t = 0$, after 12 seconds all agents may act simultaneously, achieving Ordered Response with high probability. In contrast to a message chain, where the expected response time would be at best 99 seconds, a significant improvement.[2]

The main contributions of this paper are:

- A study of *Ordered Response* protocols in the Exponentially Distributed Delay (EDD) model, which we are the first to formally define.
- Two tunable *Ordered Response* protocols are presented. Both solve the problem with probability as close to 1 as desirable. The first assumes the existence of a global clock and executes **exponentially** faster than a message chain, when the number of participating agents n is large. The second does not use a global clock, and instead employs a

[1] For modelling WSN link delays, the most widely used distributions are Gaussian, exponential, gamma and Weibull [15, 20]. We opted for the exponential distribution for its simplicity in analyzing the model, as considering other distributions would introduce technical challenges beyond the scope of our work.

[2] We assume that agents can send only a constant amount of messages for reasons we will go into later on.

procedure that ensures that, with high probability, the local clocks of the agents are *probably approximately* synchronized. As a result, it executes slower than the first former protocol, but is still quadratically faster than the message chain protocol.

- For each of the above protocols, we also provide a hybrid version that combines the protocol with a message chain to efficiently achieve *Ordered Response* w.h.p. for any n.

2 Related Work

The *Ordered Response* problem was introduced by Ben-Zvi & Moses in [3], and was investigated there in the context of the Bounded Communication Model, in which message delays have (deterministic) upper bounds. Given such bounds, the mere passage of time can provide information about the occurrence of events at remote sites, without the need for explicit confirmation. The authors extend Lamport's *happened-before* relation [13] to define a causal structure called the "centipede" (a strict generalization of the "message chain"), and show that centipedes must exist in every execution in which linear ordering of actions is ensured. The concurrent Ordered Response protocols presented in this work implement a similar communication approach to the centipede by broadcasting information. Our model does not assume deterministic upper bounds on message delays, but rather probabilistic ones. The result is a protocol that solves Ordered Response with high probability.

Assuming an exponential distribution on message arrival times is not new. In fact, such assumptions are frequently made in the context of Wireless Sensor Networks (WSNs). Specifically, for the problem of estimating and/or bounding clock skew of sensors in WSNs [1, 6, 10, 11, 14, 21]. It was observed in [1] that a single server M/M/1 queue can represent the cumulative link delay in a WSN, when the random delays are modeled as exponential random variables. Moreover, the Minimum Link Delay algorithm, proposed in [1], which shows good performance, was derived independently in [10] assuming exponentially distributed network delays. Therefore, the assumption of an exponential distribution seems to be suited for modeling WSN link delays.

In the following, we will introduce a Concurrent Ordered Response protocol designed for an environment lacking a global clock. This protocol employs a procedure to achieve probable approximate synchronization of the agents' local clocks. Previous literature has proposed clock synchronization protocols in environments with probabilistic message delays [2, 8, 18, 19]. These protocols, similar to our solutions, guarantee clock synchronization with probability as close to 1 as desirable. However, they rely on agents being able to remotely read another agent's local clock multiple times to accomplish this. The *novelty* of our approach lies in its minimal communication requirements.[3] Agents broadcast messages only once (or twice in the hybrid variant) in order to achieve probable approximate synchronization. Along with the subsequent second phase, the agents successfully achieve Ordered Response with high probability.

The rest of this work is organized as follows. Section 3 presents our model and existing *Ordered Response* solutions in the synchronous and asynchronous models. Section 4 presents the CORE protocol when a global clock is present in the model. As well as, a hybrid protocol of CORE with a message chain. Section 5 details a procedure to *probably approximately* synchronize local clocks of agents, and how we use it in the CORE protocol when a global clock is not present in the model. Finally, Section 6 concludes this work with our conclusions.

[3] Minimal communication is crucial in WSNs due to battery life concerns.

3 Preliminaries

3.1 Model Formulation

In the Ordered Response problem, we consider a set of $n+1$ agents $\mathbb{P} \triangleq \{i_0, i_1, \dots i_n\}$. The agents are assumed to be connected via a complete communication network, and agents can communicate with one another by sending and receiving messages (including broadcasts) that contain any form of information. We assume that the value of n is available as an input to the protocols, and every agent is aware of its own ID, as well as the IDs of all the other agents in the system. The system is considered to be event-driven, i.e., agents change their state only when some local event occurs. Agents wait between the occurrence of these events, and are assumed inactive prior to receiving the first message or external input from the environment in an execution.

We assume that all agents can measure the passage of time accurately using local clocks. A clock can be used to set a timer that will cause the agent to activate at some point in the future. Hence, any local event experienced by an agent is either the receipt of a message or a timer signalling. All agents are assumed to be inactive up until the first local event takes place. When an agent is activated, it immediately performs some local calculation and/or action as dictated by a deterministic distributed protocol that is shared among all agents in the system. We assume the agents are well-behaved and reliable, i.e., agents do not crash or disobey the protocol.

We designate agent i_0 to be a "supervisor" agent. At time $t = 0$, the agent i_0 receives an external input from the environment that triggers the system into action. Each of the other "worker" agents i_k has a special action α_k unique to itself. When the external input is delivered, the supervisor begins the process by sending messages to a subset of agents. From then on, each of the agents, once active, coordinates with the others with the goal of performing their respective actions in a *linear temporal order*.

Formally, let $\mathcal{P}$ be some shared protocol, and denote the set of runs of $\mathcal{P}$ by $R(\mathcal{P})$. We say that a run $r \in R(\mathcal{P})$ adheres to the specifications of *Ordered Response* if the following two properties hold:

- *Safety:* $\forall k > 1$: agent i_k does not perform the action α_k before agent i_{k-1} performs the action α_{k-1},
- *Liveness:* every agent i_k eventually performs action α_k.

Accordingly, we say that $\mathcal{P}$ solves *Ordered Response* if every run $r \in R(\mathcal{P})$ of the protocol adheres to the *Ordered Response* specifications. Later, we will investigate protocols that do not solve *Ordered Response*, but rather solve it with high probability. We say that a protocol $\mathcal{P}$ solves *Ordered Response* with some probability p, if the probability that a run of the protocol will satisfy both *Safety* and *Liveness* is at least p.

3.2 A Probability Measure on Runs

Unlike in classical message-passing communication models, we assume that message delays are determined by a probability distribution. Specifically, the arrival times of messages that are sent over any communication channel, are assumed to be sampled from an exponential distribution with parameter λ. The delays of distinct messages (including those that are sent via broadcast) are assumed to be statistically independent of one another. We name this communication model the Exponentially Distributed Delay model, denoted by the abbreviation EDD, which is shorthand for EDD(λ) when λ is clear from the context.

In this work, we exclusively consider deterministic protocols, and so the communication infrastructure provides the only source of randomness. The exponential distribution on message delays induces a probability distribution on the set of runs of any given protocol $\mathcal{P}$. We map every run of a protocol $\mathcal{P}$ in the EDD model, to a sampling of a countably infinite set of exponential random variables with parameter λ. Every exponential random variable in this set represents the delay of a message sent in a run of the protocol. This induces a natural probability space $\mathcal{S} = (\Omega, \mathcal{F}, \Pr)$ that we use to prove our claims (see Appendix A for a complete exposition of the probability space).

In a probabilistic environment, a simplistic solution to the Ordered Response problem may involve repeatedly transmitting the same message between agents. The intention is to increase the likelihood of an earlier message arrival beyond what a single message's probability distribution allows. This approach increases the message complexity of the protocol and, perhaps more crucially, imposes considerable energy costs on the individual agents, which is often unreasonable, e.g., in WSNs. Therefore, we assume any agent can only transmit a constant number of messages (i.e., independent of the number of agents in the system). If a protocol does make use of a constant number of retransmissions of the same message to the same destination (say k such messages), we can effectively model this as a single message that is sampled from an exponential distribution with a larger parameter ($\tilde{\lambda} \geq k\lambda$).

3.3 Performance and Correctness Metrics

In this work, we investigate protocols that ensure *Ordered Response* with high probability. Specifically, we allow protocols that ensure a linear temporal order with some probability p. A run in which both the *Safety* and *Liveness* conditions hold we call a *correct run*. Formally, let $t_k : \Omega \to \mathbb{R}$ be a random variable that represents the time that agent i_k performs α_k. The value of t_k is determined by the protocol $\mathcal{P}$ and a realization $\omega \in \Omega$, i.e., for every agent i_k, the protocol $\mathcal{P}$ induces a function τ_k such that $t_k = \tau_k(\omega)$. The probability of a correct run of $\mathcal{P}$ is then given by $p \triangleq \Pr(t_1 \leq t_2 \leq \ldots \leq t_n < \infty)$.

We compare protocols using two performance measurements that are a function of the number of agents n. The first is message complexity, as in the number of messages sent, and the second is *response time*. The response time of an Ordered Response protocol in a given run is defined as the time at which the last agent performs its unique action. Thus, the response time of the protocol is a random variable $RT \triangleq \max\{t_1, \ldots, t_n\}$. Since message delays are unbounded in the EDD model, there exist runs of the protocol that never terminate. Although the probability measure of these runs is zero, measuring the response time of protocols in this model is meaningless. Instead, we measure the *expected* response time, denoted by $\mathbb{E}[RT]$.

3.4 Asynchronous Ordered Response and the Message Chain Protocol

In an asynchronous system, there is no global clock and no bound on the arrival time of messages. Fortunately, there is a very simple Ordered Response protocol in this model as well, called the Message Chain protocol. At time $t = 0$, the supervisor i_0 sends a message to i_1. For all $k \geq 1$, when i_k receives a message from i_{k-1} it performs α_k and sends a message to agent i_{k+1} (if $k < n$). This creates a message chain among agents.

Since communication in the asynchronous setting is reliable, the message chain protocol ensures *Safety* and *Liveness* in every run. The correctness of the protocol stems from the fact that no agent acts before receiving a direct message from its predecessor in the agent ordering. Notice that a direct message is not a necessary condition for an agent to act.

Rather, as the analysis in Chandy & Misra [5] suggests, a message chain between every two consecutive agents in the ordering is required to ensure a linear temporal ordering of actions. Therefore, the message chain protocol presented above is an optimal solution for the asynchronous model in terms of both message complexity and response time.

To measure the response time of a protocol in an asynchronous system, we consider each message arrival time to be of length of one time unit.[4] Under such accounting, the message chain protocol exhibits $\Theta(n)$ response time, due to its sequential behavior. Since the Message Chain protocol is the optimal Ordered Response protocol in an asynchronous system, this is also a lower bound for the asynchronous Ordered Response problem, in general.

3.5 Ordered Response with Probability 1

While message arrival times are not deterministically bounded in the EDD model, they are bounded in the probabilistic sense. In particular, messages are more likely to arrive sooner rather than later. Our focus in this work will be the design and analysis of Ordered Response protocols that achieve good performance in the EDD model.

We consider the Message Chain protocol as our baseline. Notice that the protocol ensures that the probability of a *correct run* is 1. The reason is that the protocol ensures *Liveness* only *w.p.* 1, due to the fact that messages in the EDD model arrive in finite time *w.p.* 1.

▶ **Definition 1.** *We say that a run $r \in R(\mathcal{P})$ contains a message chain, if there exist a set of n messages $m_1, m_2, \ldots, m_n$ that are sent in r, and for all $1 \leq k \leq n-1$ the agent i_k sends m_{k+1} only after it receives m_k from agent i_{k-1}.*

Additionally, in a run that contains a message chain, we say that an agent i_k "receives a message chain of length k at time t" if at time t agent i_k receives message m_k.

▶ **Theorem 2.** *In the EDD model, if every run of a protocol $\mathcal{P}$ contains a message chain, then it holds for $\mathcal{P}$ that $E[RT] \in \Omega(\frac{n}{\lambda})$.*

Proof. If the run implements a message chain, then agent i_{n-1} does not terminate prior to receiving a message chain of length $n-1$, since it has to send the final message m_n. By definition, the response time of a protocol in a given run is greater or equal to the time at which i_{n-1} terminates. Thus, $RT > \xi$ where ξ is a sum of $n-1$ *i.i.d.* exponential random variables with parameter λ. Hence, $\mathbb{E}[RT] > \mathbb{E}[\xi] = \frac{n-1}{\lambda}$. ◀

From Theorem 2 we conclude that the expected response time of the Message Chain protocol in the EDD model is in $\Omega(\frac{n}{\lambda})$. Later on, we investigate protocols that trade off probability of correctness for expected response time that is sub-linear in the number of participating agents n.

4 Ordering with a Global Clock

In order to investigate the design of faster protocols, we relax the specification slightly to allow the protocol to guarantee *Safety* and *Liveness* properties with high probability (*w.h.p.*). More precisely, we now consider protocols that solve Ordered Response with probability $0 < p < 1$, where p can be as close to 1 as desirable. Our goal is to find a protocol that significantly outperforms the Message Chain protocol, in terms of expected response time.

[4] This is generally the accepted approach in the literature for measuring time of execution of a run in an asynchronous system [17].

The main disadvantage of the Message Chain protocol is its sequential behaviour. However, in the EDD model, messages are more likely to arrive earlier rather than later. Our approach leverages this towards speeding-up the response time of the protocol. The main idea behind the protocols that will be described in this section is the concurrent transmission of information to all parties. For example, instead of sending only one message to one agent informing him that the external input has arrived, the supervisor broadcasts this information to all agents in the system. We call this approach – Concurrent Ordered REsponse, abbreviated by CORE.

4.1 The CORE Protocol

In a synchronous system, there usually exists a deterministic upper bound Δ on the arrival time of messages. In the CORE protocol there is a parameter $\mathcal{D}$ that is configurable, rather than being a constant defined by the environment. This parameter is chosen so that *w.h.p.* all the messages that are sent in the supervisor's broadcast are delivered within time $\mathcal{D}$. The psuedocode of the protocol is presented in Algorithm 1.

Algorithm 1 The CORE($\mathcal{D}$) Protocol.

```
1: procedure Protocol for Agent i_0
2:     upon receiving an external input, do
3:         send "trigger" to all

4: procedure Protocol for Agent i_k
5:     upon receipt of a "trigger" message, do
6:         Wait until current time ≥ D
7:         perform α_k
```

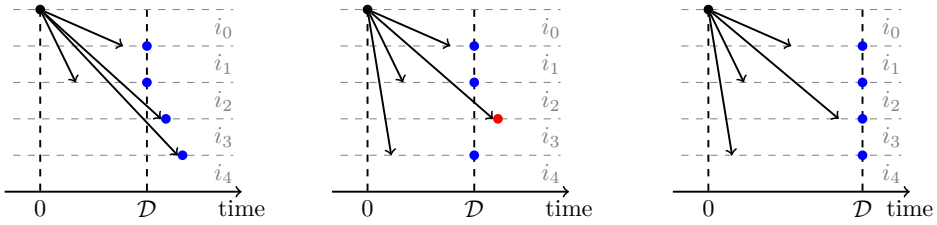
At time $t = 0$, the supervisor agent receives an external input from the environment. The agent then proceeds to broadcast a message to all other agents containing the word "trigger". Agents that receive a "trigger" message from the supervisor, wait until the global clock reads $t = \mathcal{D}$, and then act.[5] If an agent receives the message after time $\mathcal{D}$, it does not wait and acts immediately. Several examples of runs of the CORE protocol are illustrated in Figure 1.

4.2 Probability of a Correct Run

If all messages arrive within time $\mathcal{D}$, then the run will be correct. However, if messages arrive late, this does not mean the run is ruined. For example, suppose the message sent to agent i_k arrives late. Then there is some positive probability that all messages to agents $i_{k+1}, \ldots, i_n$ also arrive late, and that they will arrive in a linear temporal order.

▶ **Theorem 3.** *Fix $\mathcal{D} > 0$, and denote $q \triangleq 1 - e^{-\lambda \mathcal{D}}$, then the probability of a correct run of the CORE($\mathcal{D}$) protocol is $\sum_{k=0}^{n} \frac{1}{k!} q^{n-k} (1-q)^k$.*

[5] Notice that under our protocol several actions can be performed simultaneously, which is consistent with the original definition of Ordered Response in [3]. But the CORE protocol can be easily modified at essentially no cost to support strict ordering of the actions. I.e., if t_i denotes the time at which the i'th action is performed, then $t_{k+1} > t_k$, rather than $t_{k+a} > t_k$, in good runs of the protocol. Namely, for an arbitrarily chosen small ϵ, we can modify line 6 of the protocol so that Agent i_k waits until time $\geq \mathcal{D} + (1 - 2^{-k}) \cdot \epsilon$. In run *(c)* of Figure 1, for example, the actions will be performed in strict linear temporal order, very soon after time $\mathcal{D}$; indeed, they would all complete before time $\mathcal{D} + \epsilon$.



(a) A correct run of CORE. (b) An incorrect run of CORE. (c) Same run, with a larger $\mathcal{D}$.

Figure 1 Three examples of runs of the CORE($\mathcal{D}$) protocol.
At time $t = 0$, agent i_0 broadcasts a message to all other agents. The blue and red dots mark the points in time when the agents perform their actions. (a) – Agents i_3 and i_4 receive their messages late and act immediately and in order. This is a lucky run of the protocol. (b) – Agents i_3 receives its message late and acts immediately, but out of order. This is an unlucky run of the protocol. (c) – Same as the previous run, but $\mathcal{D}$ is chosen to be larger. All agents receive their messages before time $\mathcal{D}$ and act simultaneously.

Proof. In the CORE($\mathcal{D}$) protocol, every agent i_k acts at a time $t_k \geq \mathcal{D}$. Let $e_1, e_2, \ldots, e_n$ be the exponential random variables associated with the messages of the supervisor's broadcast to agents $i_1, i_2, \ldots, i_n$ respectively. for every agent i_k, if $e_k \leq \mathcal{D}$, then $t_k = \mathcal{D}$, and otherwise $t_k = e_k$. Hence, for all $k \geq 1$:

$$t_k = \max\{\mathcal{D}, e_k\}. \tag{1}$$

Notice that the set $\{t_1, t_2, \ldots, t_n\}$ are *i.i.d.* random variables. To calculate the probability $\Pr(t_1 \leq t_2 \leq \cdots \leq t_n)$, we use the Law of Total Probability by conditioning on the values of the set $\{e_k\}_{k=1}^n$ relative to $\mathcal{D}$, i.e:

$$E_0 = e_1 \leq \mathcal{D},\ e_2 \leq \mathcal{D}, \ldots,\ e_{n-2} \leq \mathcal{D},\ e_{n-1} \leq \mathcal{D},\ e_n \leq \mathcal{D} \tag{2}$$
$$E_1 = e_1 \leq \mathcal{D},\ e_2 \leq \mathcal{D}, \ldots,\ e_{n-2} \leq \mathcal{D},\ e_{n-1} \leq \mathcal{D},\ e_n > \mathcal{D} \tag{3}$$
$$E_2 = e_1 \leq \mathcal{D},\ e_2 \leq \mathcal{D}, \ldots,\ e_{n-2} \leq \mathcal{D},\ e_{n-1} > \mathcal{D},\ e_n > \mathcal{D} \tag{4}$$
$$\vdots \tag{5}$$
$$E_n = e_1 > \mathcal{D},\ e_2 > \mathcal{D}, \ldots,\ e_{n-2} > \mathcal{D},\ e_{n-1} > \mathcal{D},\ e_n > \mathcal{D}\,. \tag{6}$$

We use only these $n + 1$ combinations out of the possible 2^n, since for any other event the agents will necessarily act out of order, and the probability of Ordered Response conditioned on such an event would be zero. By the Law of Total Probability:

$$\Pr(t_1 \leq \ldots \leq t_n) = \sum_{k=0}^{n} \Pr(t_1 \leq \ldots \leq t_n \mid E_k) \Pr(E_k). \tag{7}$$

The probability of event E_k is the product of the probability that the first $n - k$ messages are delayed at most $\mathcal{D}$ time, and the probability that the remaining k messages are delayed longer:

$$\Pr(E_k) = q^{n-k}(1 - q)^k. \tag{8}$$

Given an event E_k, it holds that the first $n - k$ agents will act simultaneously. However the remaining k messages arrive late, and the agents act immediately upon their delivery. Due to symmetry, for the last k agents, any temporal ordering of their actions is equally likely. Thus:

$$\Pr(t_1 \leq \ldots \leq t_n \mid E_k) = \Pr(t_{n-k+1} \leq \ldots \leq t_n \mid E_k) = \frac{1}{k!}. \tag{9}$$

Plugging 8 and 9 into Equation 7 yields:

$$\Pr(t_1 \leq \ldots \leq t_n) = \sum_{k=0}^{n} \frac{1}{k!} p^{n-k}(1-p)^k. \tag{10}$$

◀

As can be seen from Theorem 3, the probability of a correct run depends both on $\mathcal{D}$ and n. Also, it includes runs that are "lucky", similarly to the one presented in Figure 1 (a). The probability that a run is correct "by design", i.e., every message arrives by time $\mathcal{D}$, is equal to $q^n = (1 - e^{-\lambda \mathcal{D}})^n$ and, in particular, the probability of a correct run tends to 1 as $\mathcal{D}$ tends to ∞, as can be seen from the following Corollary:

▶ **Corollary 4.** *The probability of a correct run of the CORE($\mathcal{D}$) protocol is at least* $(1 - e^{-\lambda \mathcal{D}})^n$.

Proof. From Theorem 3, the probability of a correct run of CORE($\mathcal{D}$) is given by:

$$\sum_{k=0}^{n} \frac{1}{k!} q^{n-k}(1-q)^k = q^n + \sum_{k=1}^{n} \frac{1}{k!} q^{n-k}(1-q)^k \geq q^n = (1 - e^{-\lambda \mathcal{D}})^n. \tag{11}$$

◀

Corollary 4 characterizes the relationship between the probabilistic communication bound $\mathcal{D}$ and the probability of a correct run of the CORE protocol. We now derive a closed-form expression for the probabilistic communication bound $\mathcal{D}$.

▶ **Corollary 5.** *Fix* $0 < p < 1$, *and let* $\mathcal{D} \geq \frac{-\ln(1-\sqrt[n]{p})}{\lambda}$. *Then the probability of a correct run of the CORE($\mathcal{D}$) protocol in a setting with a global clock, is at least* p.

Proof. Follows directly from Corollary 4: $\mathcal{D} \geq \frac{-\ln(1-\sqrt[n]{p})}{\lambda} \Rightarrow p \leq (1 - e^{-\lambda \mathcal{D}})^n$. ◀

4.3 Expected Response Time

We are interested in the expected response time of the CORE protocol. Specifically, this is the expected time of the last agent to perform its action. In a given run of the CORE protocol, the response time is determined by the arrival times of the messages from the supervisor's broadcast. They either all arrive by time $\mathcal{D}$, or at least one of them is delayed longer.

Let $e_1, e_2, \ldots, e_n$ be the set of exponential random variables that are associated with messages sent in the supervisor's broadcast. Let $\mathcal{M} = \max\{e_1, e_2, \ldots, e_n\}$ be a random variable that denotes the time when the last message in a run is delivered. Then the response time of a run of the protocol is given by $RT = \max\{\mathcal{D}, \mathcal{M}\}$.

We are interested in the expected value of the response time $\mathbb{E}[RT]$. However, deriving the expectation of the above non-linear function of a random variable and a constant is a non-trivial exercise. Fortunately, there is a simple bound on $\mathbb{E}[RT]$ that is good enough for our purposes:

$$\mathbb{E}[RT] = \mathbb{E}[\max\{\mathcal{D}, \mathcal{M}\}] \leq \mathcal{D} + \mathbb{E}[\mathcal{M}]. \tag{12}$$

The inequality follows from the fact the maximum is smaller than the sum, and the linearity of probabilistic expectation. Before we proceed in characterizing the expected response time, we prove a useful lemma:

▶ **Lemma 6.** *$E[\mathcal{M}] = \frac{H_n}{\lambda}$, where H_n is the n^{th} harmonic number; i.e. $H_n = \sum_{m=1}^{n} \frac{1}{m}$.*

Proof. Recall that $\mathcal{M} = \max\{e_1, e_2, \ldots, e_n\}$. The random variables $\{e_k\}_{k=1}^{n}$ are assumed to be *i.i.d.* exponential random variables with parameter λ. Denote by T_m to be the m^{th} smallest of the set $\{e_k\}_{k=1}^{n}$, i.e., $T_1 = \min_k\{e_k\}$ and $T_m = \min\left\{\{e_k\}_{k=1}^{n} \setminus \{T_1, \ldots, T_{m-1}\}\right\}$.

It is well known that the minimum of n *i.i.d.* exponential random variables with parameter λ is also an exponential random variable with parameter $n\lambda$. Thus, $\mathbb{E}[T_1] = \frac{1}{n\lambda}$.

Since exponential random variables are continuous, the probability that any two of the n random variables have the same value is 0. Thus, the $n-1$ other random variables all have values strictly larger than T_1. However, the memorylessness property of the exponential distribution implies that knowledge of T_1 essentially "resets" the values of the other random variables, so that the time between T_1 and T_2 is the same (distributionally) as the time until the first of $n-1$ *i.i.d.* exponential random variables takes on a value. Hence, $\mathbb{E}[T_2 - T_1] = \frac{1}{(n-1)\lambda}$.

By inductive reasoning, we get that $\forall 1 \leq m \leq n-1 : \mathbb{E}[T_{m+1} - T_m] = \frac{1}{(n-m)\lambda}$. As a result, the expected value of the maximum of the n exponential random variables is:

$$\mathbb{E}[\mathcal{M}] = \mathbb{E}[T_n] = \mathbb{E}\left[T_1 + \sum_{m=1}^{n-1}(T_{m+1} - T_m)\right] = \sum_{m=0}^{n-1}\frac{1}{(n-m)\lambda} = \sum_{m=1}^{n}\frac{1}{m\lambda} = \frac{H_n}{\lambda}. \tag{13}$$

◀

The harmonic numbers roughly approximate the natural logarithm function [7], i.e., $H_n \in \Theta(log(n))$. As a result, we gain the following theorem:

▶ **Theorem 7.** *Fix $0 < p < 1$, and let $\mathcal{D} \geq \frac{-\ln(1-\sqrt[n]{p})}{\lambda}$. The expected response time of the CORE($\mathcal{D}$) protocol is logarithmic in the number of participating agents i.e., $\mathbb{E}[RT] \in \mathcal{O}\left(\frac{\log(n)}{\lambda}\right)$.*

Proof. From Inequality 12:

$$\mathbb{E}[RT] \leq \mathcal{D} + \mathbb{E}[\mathcal{M}]. \tag{14}$$

We plug in the expressions for $\mathcal{D}$ and $\mathbb{E}[\mathcal{M}]$ from Corollary 5 and Lemma 6:

$$\mathbb{E}[RT] \leq \mathcal{D} + \mathbb{E}[\mathcal{M}] \ = \frac{1}{\lambda} \cdot \left(-\ln(1 - \sqrt[n]{p}) + H_n\right). \tag{15}$$

Notice that $-\ln(1 - \sqrt[n]{p}) \in \Theta(\log(n))$ for any $0 < p < 1$:

$$\lim_{n\to\infty} \frac{-ln(1-\sqrt[n]{p})}{\ln(n)} \underset{\text{Heine}}{=} \lim_{x\to\infty} \frac{-ln(1-\sqrt[x]{p})}{\ln(x)} \underset{\text{L'Hôpital}}{=} \lim_{x\to\infty} \frac{-\frac{1}{1-\sqrt[x]{p}} \cdot \frac{\sqrt[x]{p}\ln(p)}{x^2}}{\frac{1}{x}} \tag{16}$$

$$= \lim_{x\to\infty} \frac{-\frac{\ln(p)}{x}}{\frac{1-\sqrt[x]{p}}{\sqrt[x]{p}}} = \lim_{x\to\infty} \frac{-\frac{\ln(p)}{x}}{-\sqrt[x]{p}-1} \underset{\text{L'Hôpital}}{=} \lim_{x\to\infty} \frac{\frac{\ln(p)}{x^2}}{\frac{-\sqrt[x]{p}\ln(p)}{x^2}} \tag{17}$$

$$= \lim_{x\to\infty} \sqrt[x]{p} = 1. \tag{18}$$

Additionally, the harmonic numbers roughly approximate the natural logarithm function [7], i.e., $H_n \in \Theta(log(n))$. Thus, $\mathbb{E}[RT] \in \mathcal{O}\left(\frac{\log(n)}{\lambda}\right)$. ◀

4.4 The CORE-Message-Chain Hybrid Protocol

From Theorem 7, we see that the CORE protocol's expected response time grows logarithmically in n. In comparison, the Message Chain protocol's expected response time grows linearly in n. Hence, the CORE protocol is exponentially faster than the Message Chain protocol. However, the Message Chain protocol, while slower for large n, guarantees Ordered Response *w.p.* 1. Nonetheless, a case can be made in favour of the CORE protocol if it can maintain its edge for any n, and for any probability of a correct run p that is as close to 1 as desirable.

Currently, this is not the case. Recall that $\mathcal{D}$ tends to ∞ as p tends to 1. As a result, for any given n, there exists some probability $\bar{p}$ such that for all $\bar{p} < p < 1$ the Message Chain protocol's expected response time $\frac{n}{\lambda}$ is smaller than $\mathcal{D} = \frac{-\ln(1-\sqrt[n]{p})}{\lambda}$. The key insight here is that for small n the Message Chain protocol is very efficient and quick, but for large n it suffers from a significant slowdown in performance. Our solution is to combine the two protocols. Essentially, we now design a protocol that runs both protocols concurrently, in order to benefit from the best of both worlds.

In this hybrid protocol, the supervisor broadcasts the "trigger" message to all agents, and also sends a special message containing the word "act" to i_1. When i_1 receives this message, it starts a message chain that will pass through all agents. Agents that receive a message chain perform their action immediately, pass it on to the next agent in the ordering, and then terminate. Until then, they behave according to the CORE protocol described in Algorithm 1.

4.5 Performance and Probability of Correctness

Deriving an exact expression for the probability of a correct run in the hybrid protocol is difficult. This is mainly due to the fact that in the hybrid protocol an agent may perform its action for two reasons. The agent may act either because it received a message chain, or because it previously received a message from the supervisor and the the global time is $\mathcal{D}$. It is enough for our purposes to lower bound the probability of a correct run. Similarly to the pure CORE protocol, if all messages from the trigger broadcast are delivered by time $\mathcal{D}$, then Ordered Response is guaranteed. From Corollary 4 we have that:

▶ **Theorem 8.** *Fix a parameter $\mathcal{D} > 0$. The probability of a correct run in the CORE($\mathcal{D}$)-Message-Chain protocol is at least* $(1 - e^{\lambda \mathcal{D}})^n$.

We focus the reader's attention on two important observations. The first is that, in terms of expected response time, the hybrid protocol is no worse than the Message Chain protocol in every run. Once an agent receives a message chain, the agent immediately performs its action and terminates. The second observation is that, from some value of n, the expected response time stops growing linearly in n, and starts growing logarithmically in n. Thus, it holds for the hybrid protocol that $\mathbb{E}[RT] \in \mathcal{O}\left(\frac{\log(n)}{\lambda}\right)$.

With high probability, the message chain is automatically truncated once time $\mathcal{D}$ is reached, since no agent propagates the message chain after termination. For small n, and large $\mathcal{D}$, the message chain will more likely terminate the agents before time $\mathcal{D}$, which yields performance linear in n. However, when n is large, the message chain will be much slower. In this case, the CORE part of the hybrid protocol dominates, and the expected response time will be logarithmic in n.

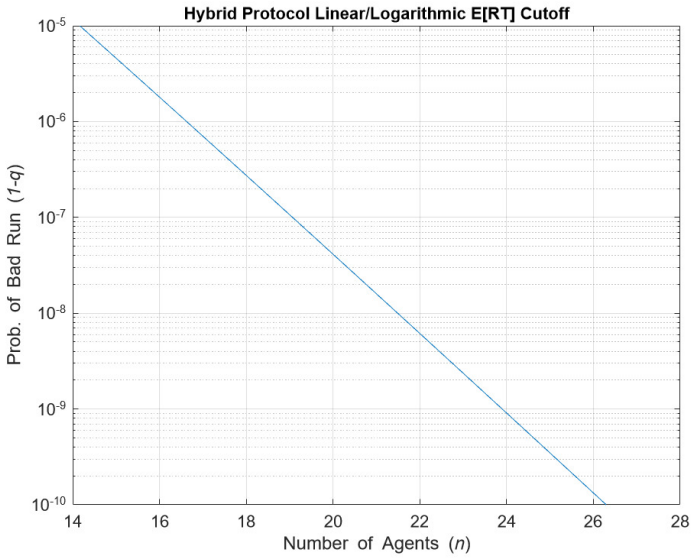


Figure 2 Plot of the Linear/Logarithmic expected response time of the CORE-Message-Chain protocol.

In Figure 2, given some probability of an incorrect run $1 - p$, we plot for which n it holds that $\frac{n}{\lambda} = \mathcal{D}$. The plot illustrates the linear/logarithmic cutoff point of the hybrid protocol, i.e., for a given probability of a correct run $p < 1$, what is the minimum number of participating agents N required such that for all $n > N$ the expected response time stops growing linearly in n, and starts growing logarithmically in n.

Figure 2 shows that the hybrid protocol exhibits linear performance only when n is small. For example, if we allow the rate of incorrect runs to be 1 in 10^6 (i.e., $p = 1 - 10^{-6}$), then a lower bound on the number of participating agents needed for the hybrid protocol to exhibit logarithmic performance is roughly $n = 18$. For $p = 1 - 10^{-9}$, the lower bound is $n = 24$. As can be seen from the slope of the graph, the hybrid protocol requires an additional 3 participating agents for a tenfold improvement in the probability of correctness. Hence, correctness probability of $p = 1 - 10^{-100}$, would require roughly $n = 300$ agents.

5 CORE without a Global Clock

In Section 4, we presented a concurrent Ordered Response protocol for the EDD model when we can assume the existence of an accurate global clock. However, agents are only assumed to be able to measure time accurately using their own local clocks, though they may be initially offset relative to each other by some unknown quantity.

We now consider the case where agents do not share a global clock, and so they initially have no idea how early or late they are relative to other agents. Previously, we assumed that the environment's external input would arrive at a point in time that we could denote as "$t = 0$", and that when an agent becomes active it would have access to the current global time. However, in the current model, the arrival of the initial external input is only observed by the supervisor, and the knowledge of when it arrived cannot be disseminated accurately to all the other agents.

This is a major challenge to the design of a concurrent Ordered Response protocol. To overcome this, we again leverage the probability distribution on message delays, to *probably approximately* synchronize the local clocks of the agents. Specifically, the worker agents will attempt to approximate when the external input has arrived based on the arrival times of messages broadcast by the supervisor and synchronize their local clocks to fit this hypothesis.

5.1 Probably Approximately Synchronized Local Clocks

For ease of exposition, from this point onward assume that in addition to the supervisor agent i_0, there are $n+1$ worker agents named $i_1, i_2, \ldots, i_{n+1}$. As before, the supervisor agent i_0 broadcasts a "trigger" message to all the other agents. When a worker agent receives a trigger message it rebroadcasts it to all of its coworkers, with the message "redirect". Concurrently, it waits for n such "redirect" messages, and logs a timestamp of the arrival time of each such message according to its local clock. When the final message arrives, the agent proceeds to calculate its hypothesis of when the external input arrived based on the observed timestamps.

Agent i_k's hypothesis T_k is the mean of the measured timestamps $\{\tau_1, \tau_2, \ldots, \tau_n\}$ minus the expected delay of a message chain of length 2, i.e. $T_k = \frac{1}{n}\sum_{m=1}^{n}\tau_m - \frac{2}{\lambda}$. Notice that the sequence of timestamps are *i.i.d.* Erlang[6] random variables with parameters 2 and λ. By the Law of Large Numbers, as $n \to \infty$ their mean converges to their expected value: $\frac{2}{\lambda}$. This means that $T_k \xrightarrow{n\to\infty} 0$, or, in this case, it converges to the true arrival time of the external input to the system.

The value T_k is used by agent i_k to offset its local clock. Formally, we denote by $C_{i_k}(t)$ the time that the local clock of agent i_k shows at time t. We assume that $C_{i_0}(t) = t$, and that the external input arrives at $t = 0$. When an agent i_k hypothesizes that the external input arrived at time T_k, it sets an adjusted local clock $C^{Adj}_{i_k}(t)$ to be the value of $C_{i_k}(t)$ offset by T_k. I.e., for all $t > 0$, agent i_k's adjusted local clock shows that $C^{Adj}_{i_k}(t) = C_{i_k}(t) - T_k$.

Since the number of agents in the system is finite, the agents' local clocks are still offset relative to the supervisor's local clock. However, when the aforementioned procedure concludes, the relative offset between agents' local clocks can be bounded with some probability.

▶ **Theorem 9.** *Let $\tau_m \sim Erlang(2,\lambda)$ for $m = 1,\ldots,n$, and let $T_k = \frac{1}{n}\sum_{m=1}^{n}\tau_m - \frac{2}{\lambda}$. Then:*

$$\Pr\left(\max_{k\geq 1}|T_k| \leq \frac{\ln(n)}{\lambda\sqrt{n}}\right) \geq \Psi(n) \tag{19}$$

where:

$$\Psi(n) \triangleq \max\left\{0,\ 1 - \left(1 - \frac{\ln(n)}{2\sqrt{n}}\right)^{2n} n^{1+\sqrt{n}} - \left(1 + \frac{\ln(n)}{2\sqrt{n}}\right)^{2n} n^{1-\sqrt{n}}\right\}. \tag{20}$$

In particular, $\Psi(n) \xrightarrow{n\to\infty} 1$.

The proof of Theorem 9 is quite lengthy and appears in the extended version of this paper [16] in Appendix B.4. Denote $\delta \triangleq \frac{\ln(n)}{\lambda\sqrt{n}}$. Then by Theorem 9, with probability at least $\Psi(n)$, the local clocks of all agents are δ-*synchronized*:

▶ **Definition 10.** *If for all $k, m \geq 1$ it holds that $\left|C^{Adj}_{i_k}(t) - C^{Adj}_{i_m}(t)\right| \leq \delta$, we say that the agents' local clocks are* δ-synchronized.

[6] An Erlang random variable is the probabilistic distribution of a sum of *i.i.d.* exponential random variables (see [9]).

As can be seen from Theorem 9, the probability that the agents' local clocks will be δ-synchronized converges to 1 as $n \to \infty$. However, for $n < 115$, $\Psi(n) = 0$. This is due to the fact that in the proof of Theorem 9, we use the Union Bound Theorem, which results in an overly course lower bound. As a result, for these values of n, the probabilistic bound given in Theorem 9 is not very informative. To see the actual probability of δ-synchronization for small n, we have performed a small experiment. We sampled the vector $(T_1, T_2, \ldots, T_n)$ a thousand times and calculated the empirical mean over the binary results of whether the vector elements are at most 2δ far from each other. As can be seen in Figure 3, the real probability of δ-synchronization is much higher, and for $n > 400$ is nearly 1.

5.2 The PA-CORE Protocol

In a setting without a global clock, we design a two phase protocol that we call PA-CORE for *Probable Approximate* Concurrent Ordered Response. The first phase is the δ-synchronization of the agents' local clocks using the procedure described in Section 5.1. In the second phase, the agents use this to perform their actions in a linear temporal order. The pusdeocode of the protocol is presented in Algorithm 2.

Algorithm 2 The PA-CORE($\mathcal{D}$) Protocol.

procedure PROTOCOL FOR AGENT i_0
  upon receiving an external input, do
    send "**trigger**" to all

procedure PROTOCOL FOR AGENT i_k
  Setup:
    $m \leftarrow 0$

  upon receipt of a "trigger" message, do
    send "**redirect**" to all

  upon receipt of a "redirect" message, do
    $m \leftarrow m + 1$
    $\tau_m \leftarrow C_{i_k}(t)$ ▷ Measure and record current time.
    if $m = n$ **then**
      $T \leftarrow \frac{1}{n}\sum_{m=1}^{n} \tau_m - \frac{2}{\lambda}$
      $C_{i_k}^{Adj}(t) \leftarrow C_{i_k}(t) - T$ ▷ Set adjusted local clock.
      Wait until $C_{i_k}^{Adj}(t) \geq \mathcal{D} + 2\delta \cdot (k-1)$
      perform α_k

At time $t = 0$, agent i_0 broadcasts a "trigger" message to all other agents. When a worker agent i_k receives a "trigger" message, it broadcasts a message with the word "redirect" to all worker agents. Upon receiving a "redirect" message, the agent records a timestamp according to its local clock. After n such "redirect" messages have been received, the agent approximates the arrival time of the external input and offsets its local clock to match this hypothesis.

Now, the agent i_k moves into phase 2. Being the k-th agent in the order, the agent waits until time $\mathcal{D} + 2\delta \cdot (k-1)$, and then performs its action. If the local agent's time is already past this point when it receives the last "redirect" message, it performs its action immediately. The additional delay of $2\delta \cdot (k-1)$ for agent i_k ensures that agents act in non-overlapping time windows.

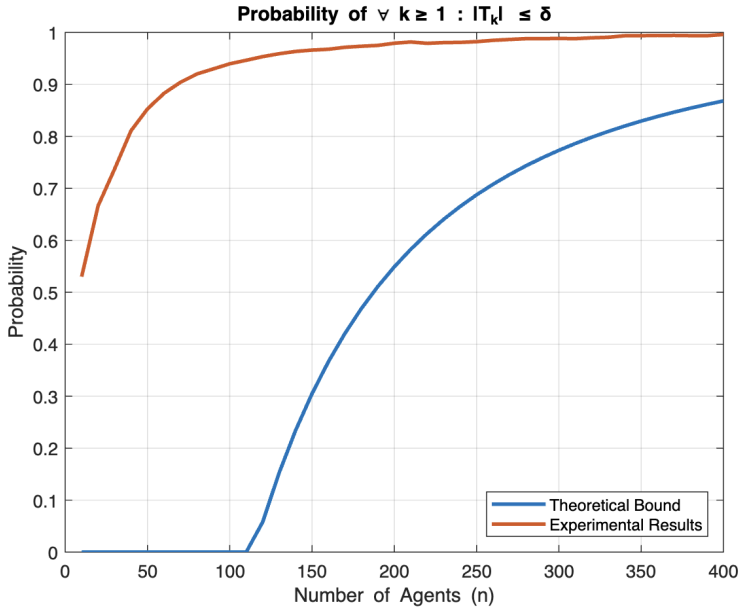


Figure 3 Probability that clocks are δ-synchronized.

Notice that if n was infinite, then $\delta = 0$, and the PA-CORE protocol would closely mirror the CORE protocol in Section 4. Intuitively, the first phase of the protocol attempts to δ-synchronize the agents' local clocks with high probability. However, δ-synchronization is not a global clock, and so agents cannot act simultaneously. Instead, in phase 2 of the protocol, the agents wait an index-dependent time delay that guarantees that they act in non-overlapping windows.

In contrast to the Message Chain protocol, the expected delay between the acting times of two consecutive agents is not a constant. In PA-CORE, this delay is at most 2δ *w.h.p.* (assuming that the agents' local clocks are δ-synchronized). Recall that $\delta \triangleq \frac{\ln(n)}{\lambda\sqrt{n}}$. This value was chosen because the product $n\delta$ is in $\mathcal{O}\left(\frac{\sqrt{n}\log(n)}{\lambda}\right)$ which we will show results in the expected response time of the protocol to grow sub-linearly in n.

5.3 Probability of a Correct Run

▶ **Lemma 11.** *In a run of the PA-CORE($\mathcal{D}$) protocol, if all of the messages arrive by time $\mathcal{D}$ and the agents' local clocks are δ-synchronized, then the run is guaranteed to be correct.*

Proof. We prove the claim by induction on the number of worker agents $n+1$.

Base Case: ($n = 1$) Since all messages arrive by time $\mathcal{D}$, and the agents' local clocks are δ-synchronized, we have that $C_{i_1}^{Adj}(\mathcal{D}) \leq \mathcal{D}+\delta$. Therefore, agent i_1 acts no later than time $\mathcal{D}+\delta$. Also, for agent i_2 we have that $C_{i_2}^{Adj}(\mathcal{D}) \geq \mathcal{D}-\delta$. Since agent i_2 waits an additional 2δ before acting, agent i_2 acts no earlier than time $\mathcal{D}+\delta$. Thus, agents i_1 and i_2 act in the proper linear temporal order. Step: Suppose that the claim holds for all $k \leq n$. By the same logic as in the base case, agent i_{n+1} acts only after agent i_n, and by the induction hypothesis, agent i_n also acts only after all the previous agents in the ordering have acted. ◀

▶ **Theorem 12.** *Fix $\mathcal{D} > 0$. The probability of a correct run of the PA-CORE($\mathcal{D}$) protocol is at least $\Psi(n) \cdot (1 - e^{-0.5\lambda\mathcal{D}})^{(n+1)^2}$.*

Proof. By Lemma 11, if all messages arrive by time $\mathcal{D}$ and all the agents' local clocks are δ-synchronized, then Ordered Response is assured. Therefore, the probability of a correct run of the protocol is greater or equal to the probability that both of these events occur.

We now derive the probability that all the messages arrive by time $\mathcal{D}$. For agent i_k, let e_k be the exponential random variable associated with the delivery time of the message sent from i_0 to i_k in agent i_0's initial broadcast. Let M_k be the maximum of the n random variables that are associated with agent i_k's redirect messages. Then, the probability that all messages arrive by time $\mathcal{D}$ is:

$$\Pr\left(\max_{1\le k\le n+1}\{e_k + M_k\} \le \mathcal{D}\right) \ge \Pr\left(\max_{1\le k\le n+1}\{2\cdot\max\{e_k, M_k\}\} \le \mathcal{D}\right) \tag{21}$$

$$= \Pr\left(\max_{1\le k\le n+1}\{\max\{e_k, M_k\}\} \le 0.5\,\mathcal{D}\right) \tag{22}$$

where the inequality above follows from the fact that $e_k + M_k \le 2\cdot\max\{e_k, M_k\}$. Notice that for all $k \ge 1$ the random variables in the set $\{\max\{e_k, M_k\}\}_{k\ge1}$ are independent of one another. Hence:

$$\Pr\left(\max_{1\le k\le n+1}\{\max\{e_k, M_k\}\} \le 0.5\,\mathcal{D}\right) = \Pr\left(\{\max\{e_k, M_k\}\} \le 0.5\,\mathcal{D}\right)^{n+1}. \tag{23}$$

Notice that $\max\{e_k, M_k\}$ is the maximum over $n+1$ *i.i.d.* exponential random variables, and therefore the CDF of the maximum is the product of their individual CDFs:

$$\Pr\left(\{\max\{e_k, M_k\}\} \le 0.5\,\mathcal{D}\right) = (1 - e^{-0.5\lambda\,\mathcal{D}})^{n+1}. \tag{24}$$

Hence, the probability that all of the messages arrive by time $\mathcal{D}$ is:

$$\tilde{p} = (1 - e^{-0.5\lambda\,\mathcal{D}})^{(n+1)^2}. \tag{25}$$

Let $E_{\mathcal{D}}$ denote the event that all messages arrive by time $\mathcal{D}$, and let E_δ denote the event that all the agents' local clocks are δ-synchronized. By Lemma 11 and Theorem 9, we obtain:

$$p \ge \Pr(E_\delta \wedge E_{\mathcal{D}}) = \Pr(E_\delta | E_{\mathcal{D}})\Pr(E_{\mathcal{D}}) \ge \Psi(n)\cdot(1 - e^{-0.5\lambda\,\mathcal{D}})^{(n+1)^2}. \tag{26}$$

◀

Denote by $\tilde{p}$ the probability that all messages arrive by time $\mathcal{D}$. By Theorem 12, $\tilde{p} = (1 - e^{-0.5\lambda\,\mathcal{D}})^{(n+1)^2}$, and the probability of a correct run is lower bounded by $\Psi(n)\cdot\tilde{p}$, where $\Psi(n)$ is as defined in Theorem 9. The probability $\tilde{p}$ depends on the value of $\mathcal{D}$, and as $\mathcal{D}$ is increased, $\tilde{p}$ draws closer to 1.

The lower bound we have derived is a product of two probabilities, $\Psi(n)$ and $\tilde{p}$. While the latter is controllable by $\mathcal{D}$, the former is static, since it is solely dependent on n. From this point onward, we focus our analysis for large n, and we assume that $\Psi(n) \approx 1$, as the experimental results that are illustrated in Figure 3, suggest. We leave the proof of a tighter lower bound for future work.

▶ **Corollary 13.** *Fix $0 < p < 1$, and let $\mathcal{D} \ge \frac{-2\ln(1-\sqrt[(n+1)^2]{p})}{\lambda}$. Then, the probability of a correct run of the PA-CORE($\mathcal{D}$) protocol is at least p.*

Proof. From Theorem 12, the probability of a correct run is lower bounded by $(1 - e^{-0.5\lambda\,\mathcal{D}})^{(n+1)^2}$. Thus: $\mathcal{D} \ge -\frac{2\ln\left(1-\sqrt[(n+1)^2]{p}\right)}{\lambda} \Rightarrow p \le (1 - e^{-0.5\lambda\,\mathcal{D}})^{(n+1)^2}$. ◀

5.4 Expected Response Times

Let RT_k be the response time of agent i_k in a run of the protocol. Before an agent performs its action, it waits for n message chains of length 2, which it uses to approximate the external input arrival time. Independently of that, the agent also relays a message from the supervisor's original broadcast. Hence, the agent's response time is the maximum over two random variables.

Formally, let e_k be the exponential random variables associated with the delivery time of the "trigger" message sent to agent i_k, and let $\tau_1^k, \tau_2^k, \ldots, \tau_n^k$ be the timestamps measured by the agent when its receives the "redirect" messages. Notice that $\{\tau_m^k\}_{m=1}^n$ is a set of *i.i.d.* Erlang random variables with parameters 2 and λ. Let $\mathcal{T}^k \triangleq \max\{\tau_1^k, \tau_2^k, \ldots, \tau_n^k\}$, then:

$$RT_k = \max\left\{e_k, \max\left\{\mathcal{T}^k, \mathcal{D} + 2\delta \cdot (k-1)\right\}\right\} \tag{27}$$

$$= \max\left\{e_k, \mathcal{T}^k, \mathcal{D} + 2\delta \cdot (k-1)\right\} \leq \max\left\{e_k, \mathcal{T}^k, \mathcal{D} + 2\delta n\right\}. \tag{28}$$

Then, the expected response time can be upper bounded:

$$RT = \max_k \{RT_k\} = \max_k \left\{\max\left\{e_k, \mathcal{T}^k, \mathcal{D} + 2\delta n\right\}\right\} = \max\left\{\max_k \left\{e_k, \mathcal{T}^k\right\}, \mathcal{D} + 2\delta n\right\} \tag{29}$$

$$\leq \max_k \left\{e_k, \mathcal{T}^k\right\} + \mathcal{D} + 2\delta n \quad \Rightarrow \quad \mathbb{E}[RT] \leq \mathbb{E}\left[\max_k \left\{e_k, \mathcal{T}^k\right\}\right] + \mathcal{D} + 2\delta n. \tag{30}$$

In order to prove that the expected response time of the PA-CORE($\mathcal{D}$) protocol is sublinear in the number of agents, we first prove the following Lemma:

▶ **Lemma 14.** *The following holds:* $\mathbb{E}\left[\max_k \left\{e_k, \mathcal{T}^k\right\}\right] \leq \frac{2H_{n^2+n}}{\lambda}$.

Proof. Recall that $\mathcal{T}^k = \max\left\{\tau_1^k, \tau_2^k, \ldots, \tau_n^k\right\}$. Each timestamp τ_m^k is measured when the agent i_k receives a "redirect" message, which completes a message chain composed of two messages. Hence, τ_m^k is a sum of two exponential random variables, i.e., $\tau_m^k = e_{m,1}^k + e_{m,2}^k$. Notice that $e_{m,1}^k + e_{m,2}^k \leq 2\max\left\{e_{m,1}^k, e_{m,2}^k\right\}$. We have that:

$$\mathcal{T}^k = \max\left\{\tau_1^k, \tau_2^k, \ldots, \tau_n^k\right\} \leq \max_{1\leq m\leq n}\left\{2\max\left\{e_{m,1}^k, e_{m,2}^k\right\}\right\} = 2\max_{\substack{1\leq m\leq n\\ \ell\in\{1,2\}}}\left\{e_{m,\ell}^k\right\}$$

$$\Rightarrow \quad \mathbb{E}\left[\max_k \left\{e_k, \mathcal{T}^k\right\}\right] \leq \mathbb{E}\left[\max_k\left\{e_k, 2\max_{\substack{1\leq m\leq n\\ \ell\in\{1,2\}}}\left\{e_{m,\ell}^k\right\}\right\}\right]$$

$$\leq \mathbb{E}\left[\max_k\left\{2e_k, 2\max_{\substack{1\leq m\leq n\\ \ell\in\{1,2\}}}\left\{e_{m,\ell}^k\right\}\right\}\right]$$

$$= 2\,\mathbb{E}\left[\max_k\left\{e_k, \max_{\substack{1\leq m\leq n\\ \ell\in\{1,2\}}}\left\{e_{m,\ell}^k\right\}\right\}\right].$$

Notice that the argument inside the expectation operator is just a maximum over $n^2 + n$ *i.i.d.* exponential random variables. By Lemma 6, we have that:

$$\mathbb{E}\left[\max_k \left\{e_k, \mathcal{T}^k\right\}\right] \leq 2\,\mathbb{E}\left[\max_k\left\{e_k, \max_{\substack{1\leq m\leq n\\ \ell\in\{1,2\}}}\left\{e_{m,\ell}^k\right\}\right\}\right] \leq \frac{2H_{n^2+n}}{\lambda}. \tag{31}$$

◀

▶ **Theorem 15.** *The expected response time of the PA-CORE($\mathcal{D}$) protocol is sub-linear in the number of participating agents. In particular,* $\mathbb{E}[RT] \in \mathcal{O}\left(\frac{\sqrt{n}\log(n)}{\lambda}\right)$.

Proof. By Lemma 14:

$$\mathbb{E}[RT] \quad \leq \quad \frac{2H_{n^2+n}}{\lambda} + \mathcal{D} + 2\delta n. \tag{32}$$

We now that prove each expression in the sum is sub-linear in n:

- For $\frac{2H_{n^2+n}}{\lambda}$:

$$H_{n^2+n} \approx \ln(n^2+n) \in \mathcal{O}(\log(n)) \quad \Rightarrow \quad \frac{2H_{n^2+n}}{\lambda} \in \mathcal{O}\left(\frac{\log(n)}{\lambda}\right) \tag{33}$$

- For $\mathcal{D}$:
  By Corollary 13:

$$\mathcal{D} \approx -\frac{2\ln\left(1 - \sqrt[(n+1)^2]{p}\right)}{\lambda}. \tag{34}$$

  Notice that $-\ln(1 - \sqrt[(n+1)^2]{p}) \in \Theta(\log(n))$ for any $0 < p < 1$:

$$\lim_{n\to\infty} \frac{-ln(1 - \sqrt[(n+1)^2]{p})}{\ln(n+1)} \underset{\text{Heine}}{=} \lim_{x\to\infty} \frac{-ln(1 - \sqrt[(x+1)^2]{p})}{\ln(x+1)} \tag{35}$$

$$\underset{\text{L'Hôpital}}{=} \lim_{x\to\infty} \frac{-\frac{1}{1-\sqrt[(x+1)^2]{p}} \cdot \frac{2\cdot\sqrt[(x+1)^2]{p}\ln(p)}{(x+1)^3}}{\frac{1}{x+1}} \tag{36}$$

$$= \lim_{x\to\infty} \frac{-\frac{2\ln(p)}{(x+1)^2}}{\frac{1-\sqrt[(x+1)^2]{p}}{\sqrt[(x+1)^2]{p}}} = \lim_{x\to\infty} \frac{-\frac{2\ln(p)}{(x+1)^2}}{-\sqrt[(x+1)^2]{p} - 1} \tag{37}$$

$$\underset{\text{L'Hôpital}}{=} \lim_{x\to\infty} \frac{\frac{4\ln(p)}{(x+1)^3}}{\frac{2\cdot\sqrt[(x+1)^2]{p}\ln(p)}{(x+1)^3}} = \lim_{x\to\infty} 2 \cdot \sqrt[(x+1)^2]{p} = 2. \tag{38}$$

- For $2\delta n$:
  By definition, $\delta = \frac{\ln(n)}{\lambda\sqrt{n}}$. Hence:

$$2\delta n = \frac{2\sqrt{n}\ln(n)}{\lambda} \in \mathcal{O}\left(\frac{\sqrt{n}\log(n)}{\lambda}\right). \qquad \blacktriangleleft$$

In conclusion, we have presented a concurrent Ordered Response protocol that guarantees a correct run with high probability in a sub-linear expected response time. However, the trade-off is increased message complexity. Due to the "redirect" broadcast that every worker agent performs, we have that the message complexity of the PA-CORE protocol is $\Theta(n)$ broadcasts.

5.5 The PA-CORE-Message-Chain Hybrid Protocol

The PA-CORE protocol guarantees *Ordered Response* with high probability in sub-linear expected response time, when $\Psi(n) \approx 1$. We know that $\Psi(n) \approx 1$ when the number of participating agents is very large. Consequently, we have not given any guarantees on the performance of the PA-CORE protocol for small n.

Similarly to the approach we have presented in the EDD model with a global clock, we propose concurrently running the PA-CORE and Message Chain protocols in EDD model without a global clock. The advantages of doing so are two-fold. Firstly, as before, the expected response time of the protocol will be no worse than that of the Message Chain protocol's, and for large n, the expected response time will be sub-linear in n. Secondly, our analysis for the expected response time and probability of a correct run are only accurate and informative for large n.

With the addition of a concurrent message chain, the performance of the hybrid protocol is well-known to us for all values of n. For small n, the message chain guarantees quick *Ordered Response w.p.* 1. For large n, the PA-CORE part of the hybrid protocol dominates, and guarantees *Ordered Response* with high probability in sub-linear expected time. As a result, the hybrid PA-CORE-Message-Chain protocol in a setting without a global clock, guarantees *Ordered Response* with high probability in sub-linear expected time.

6 Discussion and Conclusions

One of the goals of the theory of distributed systems is to understand how various properties of the system affect the solvability and efficiency of distributed problems. To this end, the impact that communicating over channels with probabilistic guarantees on message delivery times has on the efficiency of distributed protocols has received little attention in our community. A comprehensive study of the impact of probabilistic channels on coordination could ultimately involve a taxonomy of different probabilistic assumptions, a variety of coordination problems, tight upper and lower bounds, and an assessment of practical use cases and practical implementation details. This, of course, is a grand research program that goes well beyond the scope of a single conference paper. Our purpose in this paper is to take several initial steps in this direction. To this end, clarity and simplicity are central. Indeed, our goal is to illustrate that such questions can be formulated, and that probabilistic channels can provide a genuine advantage. As we have discussed, asynchronous protocols should be directly implementable in settings in which transmission over a probabilistic channel is guaranteed to succeed eventually with probability 1. In the setting of a natural coordination problem, Ordered Response, and a popular and mathematically accessible exponential distribution over transmission times, we showed is possible to improve over the asynchronous solution in a significant manner. This is true both if clocks are synchronized and when they are not. The solutions that we provide are novel, and their analysis is rigorous. Their simplicity is a feature, and not a bug. Being an initial foray into the topic, our treatment opens the door for many extensions, improvements and refinements, left for future research. Proving lower bounds and matching upper bounds is one. Exploring other probabilistic distributions, and possibly other coordination problems is another. Finally, we consider modifying the treatment for concrete practical settings an important open problem. Much is left to explore regarding coordination in this class of models. By establishing that the complexity of asynchronous solutions can be significantly improved on, we have provided strong motivation for such investigations.

References

1 Hisham S Abdel-Ghaffar. Analysis of synchronization algorithms with time-out control over networks with exponentially symmetric delays. *IEEE Transactions on Communications*, 50(10):1652–1661, 2002. doi:10.1109/TCOMM.2002.803979.

2 Koshal Arvind. Probabilistic clock synchronization in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):474–487, 1994. doi:10.1109/71.282558.

3 Ido Ben-Zvi and Yoram Moses. Beyond lamport's happened-before: On the role of time bounds in synchronous systems. In *International Symposium on Distributed Computing*, pages 421–436. Springer, 2010. doi:10.1007/978-3-642-15763-9_42.

4 Nicolas Bourbaki. *General Topology: Chapters 1–4*, volume 18. Springer Science & Business Media, 2013. doi:10.1007/978-3-642-61701-0.

5 K. Mani Chandy and Jayadev Misra. How processes learn. In *Proceedings of the fourth annual ACM symposium on Principles of Distributed Computing*, pages 204–214, 1985. doi:10.1145/323596.323615.

6 Qasim M Chaudhari, Erchin Serpedin, and Jang-Sub Kim. Energy-efficient estimation of clock offset for inactive nodes in wireless sensor networks. *IEEE Transactions on Information Theory*, 56(1):582–596, 2009. doi:10.1109/TIT.2009.2034817.

7 John H. Conway and Richard Guy. *The book of numbers*, pages 258–259. Springer Science & Business Media, 1998. doi:10.1007/978-1-4612-4072-3.

8 Flaviu Cristian. Probabilistic clock synchronization. *Distributed computing*, 3:146–158, 1989. doi:10.1007/BF01784024.

9 M Evans, N Hastings, and B Peacock. Erlang distribution. In *Statistical Distributions, 3rd ed.*, chapter 12, pages 71–73. Wiley, New York, 2000.

10 Daniel R Jeske. On maximum-likelihood estimation of clock offset. *IEEE Transactions on Communications*, 53(1):53–54, 2005. doi:10.1109/TCOMM.2004.840668.

11 Daniel R Jeske and Ashwin Sampath. Estimation of clock offset using bootstrap bias-correction techniques. *Technometrics*, 45(3):256–261, 2003. doi:10.1198/004017003000000078.

12 Leonard Kleinrock. Theory, volume 1, queueing systems, 1975.

13 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pages 179–196. Association for Computing Machinery, New York, 2019. doi:10.1145/3335772.3335934.

14 Mei Leng and Yik-Chung Wu. Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays. *IEEE Transactions on Signal Processing*, 59(10):4860–4870, 2011. doi:10.1109/TSP.2011.2160857.

15 Alberto Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1993.

16 Ariel Livshits and Yoram Moses. Probable approximate coordination. *arXiv preprint*, 2023. arXiv:2311.05368.

17 Nancy A Lynch. *Distributed algorithms*, page 466. Elsevier, 1996.

18 Alan Olson and Kang G Shin. Probabilistic clock synchronization in large distributed systems. *IEEE Transactions on Computers*, 43(9):1106–1112, 1994. doi:10.1109/12.312120.

19 Santashil PalChaudhuri, Amit Saha, and David B Johnson. Probabilistic clock synchronization service in sensor networks. *IEEE Transactions on Networking*, 2(2):177–189, 2003.

20 Athanasios Papoulis. *Probability, random variables and stochastic processes*. McGraw-Hill, Columbus, OH, USA, 3rd edition, 1991.

21 Davide Zennaro, Aitzaz Ahmad, Lorenzo Vangelista, Erchin Serpedin, Hazem Nounou, and Mohamed Nounou. Network-wide clock synchronization via message passing with exponentially distributed link delays. *IEEE transactions on communications*, 61(5):2012–2024, 2013. doi:10.1109/TCOMM.2013.021913.120595.

A Probability Space Formulation

We define a probability space $\mathcal{S} = (\Omega, \mathcal{F}, \Pr)$ that supports a countably infinite number of exponential random variables. The sample space $\Omega \triangleq \mathbb{R}_+^{\mathbb{N}}$ is the space of all non-negative sequences. The σ-algebra $\mathcal{F} \triangleq \mathbb{B}(\mathbb{R}_+^{\mathbb{N}}) = \sigma(\Pi_{i=1}^{m}(a_i, b_i) \times \Pi_{m+1}^{\infty}\mathbb{R}_+ : m \in \mathbb{N}, a_i, b_i \in \mathbb{R}_+)$ is the Borel sigma-algebra generated by the Tychonoff topology (see [4]). The probability measure of an event $E = \Pi_{i=1}^{m}(a_i, b_i) \times \Pi_{m+1}^{\infty}\mathbb{R}_+$, is then defined as $\Pr(E) = \Pi_{i=1}^{m} \int_{a_i}^{b_i} \lambda e^{-\lambda t} dt$.

We have chosen this probability space due to the continuous nature of time in our model. In simple terms, $E = \Pi_{i=1}^{m}(a_i, b_i) \times \Pi_{m+1}^{\infty}\mathbb{R}_+$ describes the event that the i-th random variable, for $i \leq m$, takes on a value in the interval (a_i, b_i), and that for all $i > m$, it takes on some arbitrary non-negative value. The number m represents the number of messages sent in runs in which the event E occurs. The event E itself is an m-dimensional box in a countably infinite dimensional space. $\mathcal{F}$ is the σ-algebra generated by closure under complement, and under countable unions and intersections of such events E for any $m \in \mathbb{N}$ and $a_i, b_i \in \mathbb{R}_+$. Additionally, we have chosen the probability measure $\Pr(E) = \Pi_{i=1}^{m} \int_{a_i}^{b_i} \lambda e^{-\lambda t} dt$, i.e., the product of m exponential probability measures, since we assume message arrival times are exponential random variables that are statistically independent.

Let $r \in R(\mathcal{P})$ be a run. We map each run to an element ω in the sample space Ω. We do this by enumerating the messages sent in r as $m_{i,j}^{(1)}, m_{i,j}^{(2)}, \ldots$ and so forth, for all agent-pairs $i, j \in \mathbb{P}$, where message $m_{i,j}^{(k)}$ is the k-th message sent from agent i to agent j. Every message $m_{i,j}^{(k)}$ is associated with an exponential random variable $e_{i,j}^{(k)}$ representing its delay. In the run r, these random variables take on values $\nu_{i,j}^{(k)}$. Thus, the run r is mapped to an $\omega \in \Omega$ that is the sequence of $\nu_{i,j}^{(k)}$ for all $i, j \in \mathbb{P}$ and $k \in \mathbb{N}$. Notice that the number of messages in r must be finite. However, the sequence ω is infinitely long and contains values for message delays that are not sent at all in r. So, for the values of ω that are not associated with a realization of a message delay in r, we choose some arbitrary value. For instance, we may choose the value 1 for all of them, or any other set of non-negative real numbers.

We will mostly use simpler notations whenever the context allows. For example, we define the probability of a run $r \in R(\mathcal{P})$ in which m messages are sent:

▶ **Definition 16.** *Fix a run $r \in R(\mathcal{P})$. Let $e_1, e_2, \ldots, e_m$ be the set of* i.i.d. *exponential random variables associated with the messages sent in r, let $\nu_1, \nu_2, \ldots, \nu_m$ be their realizations in r, and let f_{e_k} be the exponential probability density function (PDF) of e_k. Then, the probability of the run r is $\Pr(r) \triangleq \Pi_{k=1}^{m} f_{e_k}(\nu_k)$.*