On Using Time Without Clocks via Zigzag Causality

Asa Dan
Technion
asa.dan.2@gmail.com

Rajit Manohar
Yale University
rajit.manohar@yale.edu

Yoram Moses
Technion
moses@ee.technion.ac.il

ABSTRACT

Even in the absence of clocks, time bounds on the duration of actions enable the use of time for distributed coordination. This paper initiates an investigation of coordination in such a setting. A new communication structure called a *zigzag pattern* is introduced, and shown to guarantee bounds on the relative timing of events in this clockless model. Indeed, zigzag patterns are shown to be necessary and sufficient for establishing that events occur in a manner that satisfies prescribed bounds. We capture when a process can know that an appropriate zigzag pattern exists, and use this to provide necessary and sufficient conditions for timed coordination of events using a full-information protocol in the clockless model.

KEYWORDS

coordination; time; clocks; temporal ordering; time bounds

1 INTRODUCTION

Coordination is a fundamental task in distributed systems. The order in which events take place, and often also the relative timing of the events, can be of primary concern in many applications. Timing of actions can be useful, for example, when we wish to dispatch trains in a manner that ensures proper use of critical single-lane sections of the track, or schedule plane takeoffs to alleviate unnecessary congestion at the destination airports.

In asynchronous systems, processes have no access to clocks. Furthermore, they have no timing information except for what they can obtain based on the happens-before relation of [22]. In such systems, only the ordering of events can be determined, and not their relative timing. Using accurate clocks, it is possible to orchestrate much more finely-tuned temporal patterns of events at the different sites than in the asynchronous setting. Moreover, this can often be achieved using significantly less communication [5, 23]. Of course, clocks are not always available, and when they are, maintaining clocks accurate and in synchrony may be non-trivial. But even when processes do not have access to clocks, bounds on the timing of communication and of actions are routinely monitored, and system designers can often have access to reliable timing information [14].

This paper initiates an investigation of the use of time for distributed coordination when processes do not have clocks, but do have bounds on the duration of events and of communication. We call this the *bounded communication model without clocks* (or the "clockless model" for short), and denote it by bcm. It is not *a priori* obvious that the clockless model is any more powerful than the asynchronous model. We will show that it is. Throughout the text we will use notation borrowed from [31] that states timed precedence between events. We write $e \xrightarrow{x} e'$ to state that the event e takes place at least x time units before e' does.[1] In an asynchronous system, only the relative ordering of events can be coordinated, and not their timing. Thus, for example, the only way to ensure that an action $\underline{b}$ is performed by process B no more than 10000 time steps before $\underline{a}$ is performed by process A (in our notation, this is denoted by $\underline{a} \xrightarrow{-10000} \underline{b}$) is by having B perform $\underline{b}$ *after* $\underline{a}$. This is far from optimal, of course, as $\underline{b}$ could be performed way before $\underline{a}$. Similarly, the only way that B can be sure to act before A does, is if $\underline{a}$ cooperates, and waits until a message chain from B informs it that $\underline{b}$ has been performed. As we shall see, in the clockless model it is possible to allow B (or A) to act much earlier. In this paper, we will focus on two basic coordination problems in which B should act in a manner that is causally and temporally related to A's action, without requiring A to adjust its own decision to act, and often without requiring any communication between the two.

Definition 1.1 (Timed Coordination). Given processes A, B and C, suppose that A performs the action $\underline{a}$ when it receives a "go" message from C. Moreover, assume that C's decision to send this message is spontaneous. We define two coordination problems:

Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$, in which B should perform $\underline{b}$ at least x time units before $\underline{a}$ is performed; and

Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$, in which similarly B should perform $\underline{b}$ at least x time units after $\underline{a}$ is performed.

In both cases, $\underline{b}$ should be performed in a run only if $\underline{a}$ is performed.

In each of these coordination problems, A acts unconditionally when it receives C's message. Process B needs to perform $\underline{b}$ only if it can do so in a manner that conforms to the stated bounds.

Suppose that we wish to ensure that $\underline{a} \xrightarrow{0} \underline{b}$, i.e., that $\underline{a}$ occurs no later than $\underline{b}$. We can of course ensure this by creating a message chain from A to B, which starts at or after the occurrence of $\underline{a}$. Once the final message in this chain is received, $\underline{b}$ can safely be performed. In an asynchronous system, such a message chain

This research was supported in part by a Ruch grant from the Jacobs Institute at Cornell Tech, and by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF) under grant CCF 1617945. Yoram Moses is the Israel Pollak academic chair at the Technion.

PODC'17, July 25-27, 2017, Washington, DC, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-4992-5/17/07...$15.00
DOI: http://dx.doi.org/10.1145/3087801.3087839

[1]As discussed in [31], although $e \xrightarrow{x} e'$ states a lower bound of x on the time difference between the events (i.e., $t_{e'} \geq t_e + x$), the same notation can also be used to state upper bounds. Since $t_{e'} \leq t_e + y$ is equivalent to $t_e \geq t_{e'} - y$, we can capture an upper bound of y on how much later e' occurs by writing $e \xrightarrow{-y} e'$.

would be necessary. Is it possible to ensure in our model that $\underline{a}$ happens before $\underline{b}$ without creating a message chain from A to B?

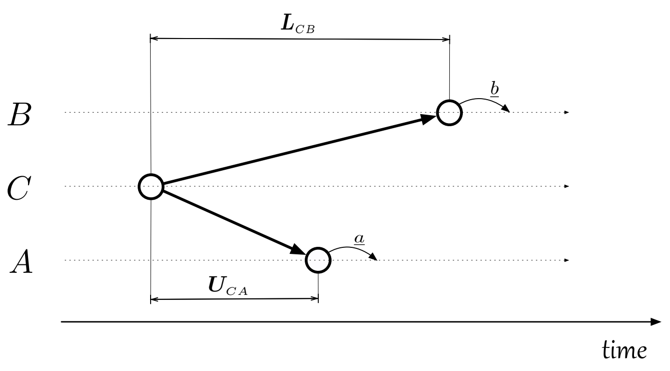


Figure 1: Coordination without direct communication

Let us return to the question of ensuring that $\underline{a} \xrightarrow{x} \underline{b}$ for a general value of x, and consider the example depicted in Figure 1. Here process C simultaneously sends messages to A and B. Let us denote by $\boldsymbol{U}_{CA}$ the upper bound on message transmission times for the channel CA, and by $\boldsymbol{L}_{CB}$ the lower bound for CB. It is easy to check that if $\boldsymbol{L}_{CB} \geq \boldsymbol{U}_{CA} + x$ then B is guaranteed to receive C's message no less than x time units after A receives it. In that case, as A is assumed to perform $\underline{a}$ upon receiving C's message, it is possible to ensure that $\underline{b}$ happens at least x time units later than $\underline{a}$, by having B perform $\underline{b}$ upon receiving C's message. Notice that this guarantees a (timed) causal connection between actions at A and at B even without any communication between A and B.

Clearly, the analysis underlying the example of Figure 1 remains valid if we replace each of the direct messages from C to A and B by a message chain, and replace the condition $\boldsymbol{L}_{CB} \geq \boldsymbol{U}_{CA} + x$ by a requirement that the sum of lower bounds along the chain from C to B exceeds x plus the sum of upper bounds along the chain from C to A. We remark that, in a precise sense, the asynchronous solution (for a simple happened-before requirement) is an instance of Figure 1 in which $C = A$, and $\boldsymbol{L}_{CB} > 0 = \boldsymbol{U}_{CA}$.

Note that if $\boldsymbol{L}_{CB} < \boldsymbol{U}_{CA} + x$ then, by waiting for more than $\delta = \boldsymbol{U}_{CA} - \boldsymbol{L}_{CB} + x$ time units before performing $\underline{b}$, process B would also ensure that $\underline{a}$ takes place x time units before $\underline{b}$ does. But we are assuming that B ***has no clock or timer*** that it can use to measure the passage of δ time steps. It can only use bounds on communication or internal actions to estimate the passage of time. We remark that in current-day technology, clocks and timers are often available. A vast portion of computer chip come with built-in clocks, and highly accurate clock synchronization algorithms are by now standard [18]. But this does not cover all distributed systems of interest. Indeed, it is becoming popular to consider biological systems such as the brain or human body as instances of distributed systems. There, no explicit clock can be found, although timing appears to play a role [6, 20]. Another setting that fits the bcm model is that of asynchronous (or self-timed) VLSI circuits, where there is no clock but there are bounds on data transfer along wires and on delays of gates [32].

A natural question at this point is whether the pattern depicted in Figure 1 is typical for coordinating actions based on transmission bounds. In other words, is this essentially the only way in which B can guarantee that $\underline{a}$ will be performed (sufficiently long) before $\underline{b}$ in the clockless model? Interestingly, the answer is No. Consider the scenario depicted in Figure 2a. In this case E sends a message to B and to D, while C sends a message to D and to A. Moreover, D receives C's message before it receives E's message. Finally, A performs $\underline{a}$ upon receiving C's message, and B performs $\underline{b}$ when it receives the message that E sends. As depicted in Figure 2a, denote the sending times of C and E's messages by t_c and t_e. Moreover, let t_a and t_d be the times at which A and D receive C's message, and let t_b the time at which B receives E's message. Clearly, $\underline{b}$ is performed no earlier than time $t_e + \boldsymbol{L}_{EB}$, yielding inequality (i) below. Similarly, the action $\underline{a}$ is performed no later than time $t_c + \boldsymbol{U}_{CA}$, yielding inequality (iv). However, the fact that E's message to D arrives after C's message arrives implies that t_e can not be pushed "too far" back relative to t_c. After all, the message along ED took no more than $\boldsymbol{U}_{ED}$ time units (inequality (ii)), and the one along CD took no less than $\boldsymbol{L}_{CD}$ (inequality (iii)). Altogether, we have:

$$\begin{array}{rlll} \text{(i)} & t_b & \geq & t_e + \boldsymbol{L}_{EB}, \\ \text{(ii)} & t_e & > & t_d - \boldsymbol{U}_{ED}, \\ \text{(iii)} & t_d & \geq & t_c + \boldsymbol{L}_{CD}, \text{ and} \\ \text{(iv)} & t_c & \geq & t_a - \boldsymbol{U}_{CA}. \end{array}$$

By substitution, we have that $t_b > t_a - \boldsymbol{U}_{CA} + \boldsymbol{L}_{CD} - \boldsymbol{U}_{ED} + \boldsymbol{L}_{EB}$. Thus, $t_b > t_a + x$ is guaranteed in this case if

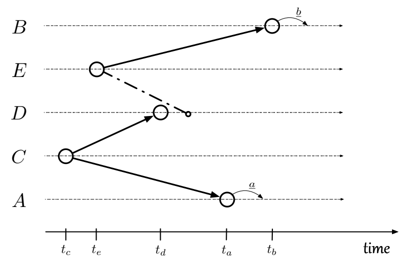
$$-\boldsymbol{U}_{CA} + \boldsymbol{L}_{CD} - \boldsymbol{U}_{ED} + \boldsymbol{L}_{EB} \geq x. \tag{1}$$

The reader may correctly suspect at this point that the zigzag pattern of Figure 2a can be extended by adding an arbitrary finite number of additional zigs and zags. Indeed, in that case a more elaborate condition in the style of Equation (1), based on a longer derivation, will ensure that $\underline{a}$ happens more than x time units before $\underline{b}$. The first result of our analysis is a proof that this is tight. We will show that, in a precise sense, the existence of an appropriate zigzag pattern is a necessary condition for B's performing the action $\underline{b}$ in Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$ or Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$.

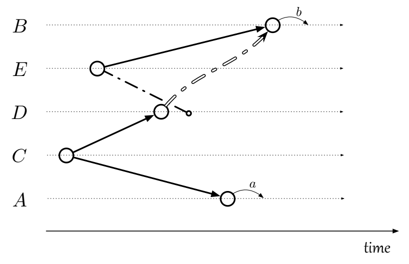
Since a zigzag pattern is necessary, we have by the *Knowledge of Preconditions principle* of [30] that B must *know* that a zigzag pattern exists when it performs $\underline{b}$. Interestingly, even if the processes follow a full-information protocol,[2] the mere existence of a zigzag pattern is not sufficient to enable process B to correctly coordinate its action. In a run containing the pattern of Figure 2a, for example, if B receives no messages from D, then B would not be able to detect the existence of the zigzag pattern, because in the eyes of B it may be possible that C will send its messages only in the far future.

There are, of course, cases in which B can detect that an appropriate zigzag pattern exists. In such a case, B can decide to perform its action $\underline{b}$ and be sure that $\underline{a}$ happens before $\underline{b}$. Consider Figure 2b. Suppose that the bounds satisfy the condition of Equation (1). Moreover, let's assume that every message contains a header specifying who its intended recipients are. Once B receives E's message (with an indication that it was also sent to D), and receives a message from D (denoted by a dashed line in Figure 2b) indicating that D

[2]A *full-information protocol* (*fip*) is one in which every message sent encodes the sender's complete history up to the point at which it is sent.



(a) Zigzag-based happens-before



(b) Zigzag-based coordination

Figure 2: A zigzag communication pattern

heard from C before D heard from E, then B can perform $\underline{b}$ and be guaranteed to satsify Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$. This is an instance of a *visible zigzag* pattern, which is a zigzag pattern that is extended by an appropriate set of message chains. Our analysis will identify particular visible zigzag patterns as necessary and sufficient for B's action in instances of Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ or Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$.

This paper is organized as follows: Section 2 introduces the bounded communication model, protocols, and standard aspects of causality. It also presents two ways of describing a point on a local timeline: Since processes do not have access to clocks, one way is in terms of the local state that the process is in, and another way is as the point at which a message chain arrives from a point of the first type. Section 3 introduces zigzag patterns and shows that they are necessary and sufficient for guaranteeing a precedence relation. In Section 4 the notion of a visible zigzag pattern is introduced, and it is shown to be necessary and sufficient for optimal behavior in a coordination task. Section 5 sketches the two variants of bounds graphs used in our technical analysis, and sketches the ideas underlying the proofs of our main theorems. Finally, Section 6 provides a discussion of our results, their implications, and direction for future work. Detailed proofs of our theorems and claims appear in [9].

1.1 Related Work

Lamport's seminal work [22] on the ordering of events in asynchronous systems introduced the happened before relation, and initiated an orderly account of the role of causality in the ordering of events. Roughly speaking, his work shows that the only way to implement instances of Late$\langle \underline{a} \xrightarrow{0} \underline{b} \rangle$ in an asynchronous setting is by constructing a message chain from A to B. Using the terminology of [22], one can consider the *causal past* of an event in an asynchronous setting to be the set of events from which it has received a message chain. In our analysis, this set also plays an important role. However, the bounds provide partial information on the timing of events in the past, and, moreover, the past guarantees the occurrence of events that are not seen by the process (i.e., they do not have explicit message chains to the process).

Clocks are very useful tools for coordinating actions in distributed systems (see, e.g., [8, 10, 17, 19, 26–28]). There is a vast literature on real-time systems and on time in multi-agent systems (see, e.g., [21]). Clock synchronization based on bounds on message transmission times was studied extensively in the 70's and 80's [1, 11, 12, 16, 24, 25, 33, 35]; see [34] for an early survey.

One important aspect that our work shares with the clock synchronization literature is the fact that bounds on the duration of events or on transmission times play an important role. Indeed, some of our technical analysis is based on bounds graphs that are strongly inspired by [33] and [31]. In particular, the notion of timed precedence we use comes from [31]. Our study diverges from the existing literature in the fact that no clocks or timers whatsoever are assumed, and the only timing information comes from the observed events and the guaranteed bounds.

An early suggestion to use knowledge to study time and coordination appeared in [29]. Knowledge theory has been used to investigate the protocols and communication patterns that can solve coordination problems in systems with global clocks or accurate timers in [2–5, 15]. In such settings, these works provide tools for a wide variety of coordination tasks.

2 MODEL AND PRELIMINARY DEFINITIONS

2.1 The Bounded Communication Model

We focus on a simple setting of a communication network Net = (Procs, Chans) modeled by a directed graph whose nodes are the processes Procs = $\{1, \ldots, n\}$ and whose edges are the communication channels among them. We identify time with the natural numbers, $\mathbb{N}$, where a single time step should be thought of as the minimal relevant unit of time. There are lower and upper bounds on message transmission times per channel, specified by a pair of functions $\boldsymbol{L}, \boldsymbol{U}\colon \mathsf{Chans} \to \mathbb{N}$, that satisfy $1 \le \boldsymbol{L}_{ij} \le \boldsymbol{U}_{ij} < \infty$ for all $(i, j) \in \mathsf{Chans}$.

Paths in Net are specified by sequences of process names. We denote a singleton sequence $[i]$ simply by i, and the concatenation of sequences p and q by $p \cdot q$. We define the composition of two sequences $p = [i_1, \ldots, i_k, j]$ and $q = [j, h_1, \ldots, h_m]$ in which the last element of p coincides with the first element of q by $p \odot q \triangleq [i_1, \ldots, i_k, j, h_1, \ldots, h_m]$. We extend the notation of upper and lower bounds on message transmission times to paths $p = [i_1, \ldots, i_d]$ in the network graph by defining

$$\boldsymbol{L}(p) \triangleq \sum_{i_k=1}^{d-1} \boldsymbol{L}_{i_k i_{k+1}} \quad \text{and} \quad \boldsymbol{U}(p) \triangleq \sum_{i_k=1}^{d-1} \boldsymbol{U}_{i_k i_{k+1}}.$$

For ease of exposition, actions are assumed to be instantaneous.[3] A global state (or a snapshot of the system) will have the form $\boldsymbol{g} = (\ell_e, \ell_1, \ldots, \ell_n)$, consisting of a state ℓ_e for the environment e, and one local state ℓ_i for every process $i \in$ Procs. A tuple $\gamma = \big((\mathsf{Net}, \boldsymbol{L}, \boldsymbol{U}), \mathcal{G}_0\big)$ whose first component is a time-bounded network as described above, and $\mathcal{G}_0$ is a set of possible initial global states, is called the ***context*** in which a protocol operates.

A run r is an infinite sequence of global states. Thus, $r(m)$ is a global state for every $m \in \mathbb{N}$. We denote by $r_i(m)$ process i's local state in $r(m)$. Processes can perform (application-dependent) local actions and send messages along their outgoing edges. For simplicity, we will assume that the local state of a process consists of an initial state followed by the sequence of events (local actions, message sends and message receives) that the process has observed.

The bounds on message delivery are enforced by assuming the existence of a scheduler, which we call the *environment*. The environment's local state contains the current contents of all channels in Chans, and for every message in a channel it also records the time at which the message was sent. At any point in time, the environment can deliver messages to each of the processes. It can nondeterministically choose whether or not to deliver a message μ in a channel $(i, j) \in$ Chans at time t if the sending time t_μ of μ satisfies $\boldsymbol{L}_{ij} \le t - t_\mu < \boldsymbol{U}_{ij}$. The environment ***must*** deliver μ to j at time t if $t - t_\mu = \boldsymbol{U}_{ij}$. We remark that if a message μ is delivered to i at time t in the run r, then i's local state at time t, $r_i(t)$, will record the fact that i received μ.

We assume a set $\mathbb{E}$ of ***external*** messages, where the environment may nondeterministically choose at any point ($t > 0$) whether to deliver messages from $\mathbb{E}$ to an arbitrary process. Such delivery is spontaneous, and is independent of other (past or present) events in the run. For simplicity we assume that a particular external message of $\mathbb{E}$ can be delivered to at most one process in a given run. Since processes in the bcm model have no clocks, we assume that their actions are event based. A process is scheduled to move only when it receives messages (either external or internal).[4] It can then perform a finite sequence of actions.

Recall that we assumed in Definition 1.1 that C's decision to send a "go" message in an instance of Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ or Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$ is spontaneous. Formally, we will assume that there is a message $\mu_{go} \in \mathbb{E}$ such that C will send the "go" message to A when it receives μ_{go}.

Processes follow a protocol $\mathsf{P} = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$, where P_i, process i's protocol, is a deterministic function of i's local state. A specific class of protocols we use in this paper are what we call ***flooding full-information protocol***s (FFIP). An FFIP is a protocol in which each process that receives a message immediately sends a message, containing its entire local state, to all of its neighbors. In a precise sense, FFIP's are general protocols for bcm: Just as with standard full-information protocols in the synchronous model (see, e.g., [7]), it is possible to simulate any given protocol in the bcm model by one that communicates according to the FFIP.

[3]Our analysis will apply even in the case in which actions extend over time. Such an action will be modeled as a special channel from the process to itself, with lower and upper bounds for the channel. The invocation and completion of the action would each be instantaneous events.

[4]In particular, processes do not spontaneously perform actions at time 0.

Given a protocol P and a bounded context $\gamma = \big((\mathsf{Net}, \boldsymbol{L}, \boldsymbol{U}), \mathcal{G}_0\big)$, we denote by $\mathcal{R} = \mathcal{R}(\mathsf{P}, \gamma)$ the set of runs of P in context γ. We call it the system representing P in γ. A run r belongs to $\mathcal{R}$ exactly if (1) $r(0) \in \mathcal{G}_0$, and (2) for all $m > 0$, $r(m)$ is obtained from $r(m-1)$ following the rules described above. A more formal definition appears in [9]. Henceforth, whenever a system is mentioned, it is assumed to have this form. We say that a given protocol P ***implements*** Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ (resp. Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$) if in all runs $r \in \mathcal{R}(\mathsf{P}, \gamma)$ process A performs $\underline{a}$ when it receives the "go" message, and process B performs $\underline{b}$ in r only if $\underline{a}$ is performed in r, and only at a time that is consistent with the specification of Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ (resp. the specification of Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$).

2.2 Reasoning about bcm Systems

In the coordination problems Early$\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ and Late$\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$ specified in Definition 1.1, process B needs to decide whether and when to perform a particular action $\underline{b}$. In particular, it needs to estimate the relative time difference between points on different processes' timelines: It's current point, and the point at which A performs $\underline{a}$. Because processes have no clocks, formally defining points on a timeline is somewhat subtle. Rather than distinguishing the points along the timeline of a given process according to the times at which they arise, to which processes have no access, one useful way is to identify a local point with the local state of the process. We call a pair $\sigma = (i, \ell)$ consisting of a process name and a local state for this process a *basic node*. In order to emphasize its site i, we sometimes call such a node an ***i-node***. We say that a basic node $\boldsymbol{\sigma} = (i, \ell)$ ***appears in r*** if $r_i(m) = \ell$ holds for some time m.

While the local state of a process in the FFIP protocol does not repeat twice in non-contiguous intervals of the same run, a local state can remain constant along some time interval. During such an interval the process cannot observe the passage of time; it observes only the state transitions. For a basic node $\sigma = (i, \ell)$ that appears in a run r, we define $\mathbf{time}_{\boldsymbol{r}}\,(\boldsymbol{\sigma})$ to be the minimal m such that $r_i(m) = \ell$. This allows us to treat a basic node as specifying a particular (externally observable) time in the run.[5] While a run r can be uniquely determined by the set of its basic nodes and their respective times, different runs can possess the same set of basic nodes, and differ in their timing.

For a given site i, an i-node σ' is called a *successor* of another i-node σ in r if $\mathsf{time}_r(\sigma) < \mathsf{time}_r(\sigma')$ and there is no i-node σ'' such that $\mathsf{time}_r(\sigma) < \mathsf{time}_r(\sigma'') < \mathsf{time}_r(\sigma')$. If σ' is the successor of σ, then we call σ the *predecessor* of σ'.

Definition 2.1. Given a run r, we define Lamport's *happens-before* relation among basic nodes that appear in r, denoted by $\boldsymbol{\sigma' \rightsquigarrow_r \sigma}$, to be the minimal transitive relation that satisfies (i) Locality: If both σ' and σ are i-nodes and $\mathsf{time}_r(\sigma') \le \mathsf{time}_r(\sigma)$, then $\sigma' \rightsquigarrow_r \sigma$, and (ii) if a message is sent in the run r at σ' and delivered to σ, then $\sigma' \rightsquigarrow_r \sigma$. We say that σ' is ***in the past of σ*** in r if $\sigma' \rightsquigarrow_r \sigma$, and we define $\mathbf{past}(\boldsymbol{r}, \boldsymbol{\sigma}) \triangleq \{\sigma' : \sigma' \rightsquigarrow_r \sigma\}$.[6]

[5]Since processes act in an event-driven fashion, the time at which i acts in r when in local state ℓ is precisely $\mathsf{time}_r(\sigma)$.

[6] The $\rightsquigarrow_r$ condition is defined w.r.t. a specific run r. However, since we restrict attention to full-information protocols, the run does not play an essential role. This is because if $\sigma \rightsquigarrow_r \sigma'$ holds, then $\sigma \rightsquigarrow_{r'} \sigma'$ holds for every run r' in which both nodes appear.

General Nodes. We view a process as having access to its local state, and hence to its current basic node, at any point. Indeed, since processes are assumed to be following a full-information protocol, it also has access to all the basic nodes that appear in its past. But the points with which B should coordinate its action (the points where A performs $\underline{a}$) are often not in its past. So B is aware neither of the real time at which they occur, nor of the basic nodes, since it cannot identify their local states. Recall, however, that processes are assumed to follow an FFIP protocol, in which whenever a process receives a message, it broadcasts (its local state) to all of its neighbors. So if $\sigma' \in \mathsf{past}(r,\sigma)$ and σ' is not an initial node (i.e. not a node from time 0), then there are typically many message chains starting at σ'. Along these message chains appear new nodes, and it is with such nodes that it may need to coordinate, and about whose timing we need to reason. We now define the class of general nodes, which can be defined as being at the end of a path in the network from a given basic node. We proceed as follows.

Definition 2.2. Let σ be a basic i-node, and p be a path in Net that begins at i, then $\langle\sigma,p\rangle$ is a (general) node that describes the basic node that will receive the message chain that goes along p starting at σ. We say that ***$\theta = \langle\sigma,p\rangle$ appears in*** a run $\boldsymbol{r}$ if both σ appears in r, and p is a path in Net (so that there is a message chain in r that leaves σ and goes along p).

Note that if p is a singleton (i.e. $p = [i]$) and σ is an i-node, then $\theta = \langle\sigma,p\rangle$ denotes σ itself. However, if p is not a singleton, then θ corresponds to a basic node whose identity depends on the run in question. The correspondence is defined as follows.

Definition 2.3. Let $\theta = \langle\sigma,p\rangle$ be a node that appears in the run $r \in \mathcal{R}$. The basic node that corresponds to θ in r, $\mathtt{basic}(\theta,r)$, is defined inductively as follows:

(a) If $\theta = \langle\sigma,j\rangle$ (so p is a singleton), then $\mathtt{basic}(\theta,r) = \sigma$. Otherwise,
(b) Let $p = p' \cdot j$ be a non-singleton, and let $\mathtt{basic}(\langle\sigma,p'\rangle,r) = \sigma'$. If the message sent in r from σ' to process j is delivered at σ'', then $\mathtt{basic}(\langle\sigma,p\rangle,r) = \sigma''$.

General nodes will inherit properties from their corresponding basic nodes. Thus, we define $\mathsf{time}_r(\theta) \triangleq \mathsf{time}_r(\mathtt{basic}(\theta,r))$, we write $\theta \rightsquigarrow_r \theta'$ iff $\mathtt{basic}(\theta,r) \rightsquigarrow_r \mathtt{basic}(\theta',r)$, and call θ an i-node if $\mathtt{basic}(\theta,r)$ is an i-node. For a j-node $\theta = \langle\sigma,p\rangle$ of r and a path q in Net, where q begins at process j, it will be convenient to write $\theta \odot q$ as shorthand for the node $\langle\sigma,p \odot q\rangle$.

Clearly, a node $\theta' = \langle\sigma',p'\rangle$ can appear in a run r only if σ' appears in r. However, if σ appears in r and $\sigma' \not\rightsquigarrow_r \sigma$, then σ might not be able to distinguish whether $\theta' = \langle\sigma',p'\rangle$ indeed appears in the current run. We shall say that a general node $\theta' = \langle\sigma',p'\rangle$ is ***σ-recognized*** iff $\sigma' \rightsquigarrow_r \sigma$ and either (i) p' is a singleton, or otherwise (ii) $\langle\sigma',p''\rangle \not\rightsquigarrow_r \sigma$ for every non singleton prefix p'' of p'. Note that in an FFIP protocol, σ actually "knows" that every σ-recognized node appears in the run. Moreover, such nodes have a unique representation using σ-recognized nodes. More formally, in a given run r, if node θ can be represented in the form $\langle\sigma',p'\rangle$ where $\sigma' \rightsquigarrow_r \sigma$, then there is a unique σ-recognized node σ'' and path p'' such that $\theta = \langle\sigma'',p''\rangle$. Intuitively, in the eyes of σ, every node has a well-defined "σ-recognized" representation.

3 ZIGZAG PATTERNS AND TIMED PRECEDENCE

We adapt the notion of timed precedence from [31] to nodes in our setting. Formally, given a run $r \in \mathcal{R}$, we say that a run ***$\boldsymbol{r}$ satisfies*** $\theta \xrightarrow{x} \theta'$, and write $(\mathcal{R},r) \models \theta \xrightarrow{x} \theta'$, iff both (i) the nodes θ and θ' appear in r, and (ii) $\mathsf{time}_r(\theta) + x \leq \mathsf{time}_r(\theta')$. (While the system $\mathcal{R}$ does not play a role in this definition, it is included here because it will play a role in our later analysis.)

Our discussion in Section 1 shows that communication as in Figure 1 ensures that $\underline{a} \xrightarrow{L_{CB}-U_{CA}} \underline{b}$, and similarly that a pattern as in Figure 2a ensures a precedence as captured in Equation (1). We now define general zigzag patterns and relate them to timed precedence. The basic building block is a *two-legged fork* (see Figure 3):

Definition 3.1. A ***two-legged fork*** in r is a triple $F = \langle\theta_0,\theta_1,\theta_2\rangle$ of nodes of r, such that $\theta_1 = \theta_0 \odot p_1$ and $\theta_2 = \theta_0 \odot p_2$, for process sequences p_1 and p_2. We denote $\mathsf{base}(F) = \theta_0$, $\mathsf{head}(F) = \theta_1$, and $\mathsf{tail}(F) = \theta_2$.

In a two-legged fork, there are direct message chains (possibly empty) from the base node to the head and to the tail of the fork. Figure 1 is an example of a two-legged fork in which the message chains consist of single messages, while Figure 3 depicts one with longer paths from the base node to head and tail nodes.

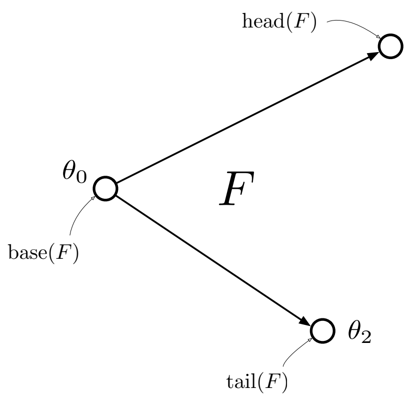


Figure 3: A two-legged fork $F = \langle\theta_0,\theta_1,\theta_2\rangle$**.**

Let θ be an i-node, and let $F = \langle\theta,\theta \odot p_1,\theta \odot p_2\rangle$ be a two-legged fork in r. We define the ***weight*** of F to be

$$\mathsf{wt}(F) \triangleq \boldsymbol{L}(p_1) - \boldsymbol{U}(p_2).$$

In Figure 1, for example, $\mathsf{wt}(F) = \boldsymbol{L}_{CB} - \boldsymbol{U}_{CA}$. The existence of a two-legged fork F in r with tail θ_1 and head θ_2 implies that $(\mathcal{R},r) \models \theta_1 \xrightarrow{\mathsf{wt}(F)} \theta_2$.

A Zigzag pattern is made of a sequence of suitably composed two-legged forks. Roughly speaking, the head of each fork should be on the same timeline as, but appear no later than, the tail of the next fork in the sequence. If they coincide at the same basic node the forks are called *joined.* Otherwise, the tail will be at least one time unit later than the preceding head. More formally, we define

Definition 3.2. A ***zigzag pattern*** from node θ to θ' in the run r is a sequence $Z = (F_1, \ldots, F_c)$ of two-legged forks in r, with $c \geq 1$, such that $\mathsf{tail}(F_1) = \theta$ and $\mathsf{head}(F_c) = \theta'$. Moreover, if $c > 1$ then for every $k = 1, \ldots, c-1$ there is a process j such that both $\mathsf{head}(F_k)$ and $\mathsf{tail}(F_{k+1})$ correspond to j-nodes, and $\mathsf{time}_r(\mathsf{head}(F_k)) \leq \mathsf{time}_r(\mathsf{tail}(F_{k+1}))$.

Figure 2a depicts a zigzag pattern consisting of $c = 2$ forks, which are not joined, since the head of the lower fork and the tail of the upper one correspond to distinct nodes on D's timeline.

The notion of weight extends to zigzag patterns. Consider a zigzag pattern $Z = (F_1, \ldots, F_c)$, and denote by $S(Z)$ the number of forks $F_k \in Z$ that are not joined to their successor (i.e., $\mathsf{head}(F_k)$ strictly precedes $\mathsf{tail}(F_{k+1})$). The ***weight*** of Z is defined by

$$\mathsf{wt}(Z) \triangleq \sum_{k=1}^{c} \mathsf{wt}(F_k) + S(Z).$$

We can thus justify the claim that zigzag patterns are sufficient for establishing timed precedence in bcm systems:

Theorem 3.3 (Zigzag Sufficiency). *Let Z be a zigzag pattern from node θ_1 to θ_2 in the run $r \in \mathcal{R}$. Then $(R, r) \models \theta_1 \xrightarrow{\mathsf{wt}(Z)} \theta_2$.*

The intuition is that each fork implies a timed precedence between its tail and its head, and the concatenation of forks in the zigzag pattern introduce a simple timed precedence between the head of one fork and the tail of its successor. Recall that, by assumption, if the successive forks are not joined, then they are separated by at least one time unit.

What is perhaps more instructive than Theorem 3.3 is that, in a precise sense, the only way to guarantee a timed precedence relation is via a zigzag pattern of this type. More formally, we say that a system $\mathcal{R}$ ***supports*** the statement $\theta_1 \xrightarrow{x} \theta_2$ if, for all $r \in \mathcal{R}$, if one of the nodes θ_1 or θ_2 appears in r, then both nodes appear in r, and $(\mathcal{R}, r) \models \theta_1 \xrightarrow{x} \theta_2$. We can show:

Theorem 3.4 (Zigzag Necessity). *Suppose that $\mathcal{R}$ supports $\theta_1 \xrightarrow{x} \theta_2$. Moreover, assume that θ_1 and θ_2 both appear in a run $r \in \mathcal{R}$, with $\mathsf{time}_r(\theta_1) > 0$ and $\mathsf{time}_r(\theta_2) > 0$. Then there is a zigzag pattern Z in r from θ_1 to θ_2 with $\mathsf{wt}(Z) \geq x$.*

Suppose that a protocol guarantees a particular time precedence constraint among a given pair of actions. Then it must ensure the existence of an appropriate zigzag pattern in the run. We remark that the requirement that $\mathsf{time}_r(\theta_2) > 0$ in the theorem ensures that the node θ_2 is not an initial node. In our model, protocols cannot perform actions at initial nodes, and precedence among initial nodes can be obtained without the existence of zigzags.

4 USING ZIGZAG CAUSALITY FOR COORDINATION

Theorems 3.3 and 3.4 show that zigzag patterns are necessary and sufficient for ensuring that a precedence relation between two nodes holds. It follows, for example, that B can act in an instance of $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ only at a node that is the tail of a zigzag pattern of weight x whose head is the node at which A performs $\underline{a}$. (Similarly, the roles of head and tail need to be reversed for an instance of $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$.) However, as discussed in the introduction, it is not guaranteed that a node at either end of the zigzag pattern is able to detect the existence of the pattern, and such endpoint node might not know that the necessary precedence condition holds.

We will show that in order to act in one of the two coordination tasks we are considering, B must know that $\theta_1 \xrightarrow{x} \theta_2$ holds, for the two nodes at which A and B act. Next, we will characterize the communication patterns that give rise to such knowledge of a timed precedence, and thus are necessary for coordinating A and B's actions. We start by defining an appropriate notion of knowledge for the bcm model, which will allow us to formulate and prove these results.

4.1 Reasoning About Coordination

Our focus is on coordinating actions at different sites in a manner that satisfies temporal constraints. We use the notion of knowledge of [13] to reason about what a process knows about the relevant aspects of the timing of events. We now describe just enough of the logical framework to support our analysis.

Two runs $r, r' \in \mathcal{R}$ are said to be ***indistinguishable*** at the basic node σ, which we denote by $r \sim_\sigma r'$, if σ appears both in r and in r'. Intuitively, if $\sigma = (i, \ell)$ appears in both runs, then when i's local state is ℓ, it cannot distinguish whether the run is r or r'. Knowledge is the dual of indistinguishability. I.e., a fact is known at a node if it is true of all indistinguishable runs. In particular, in this paper, we focus on knowledge of precedence statements at basic nodes. We write $(\mathcal{R}, r) \models K_\sigma(\theta_1 \xrightarrow{x} \theta_2)$ to state that in the run $r \in \mathcal{R}$ the precedence statement is known at the basic node σ. It is formally defined as follows:[7]

$$(\mathcal{R}, r) \models K_\sigma(\theta_1 \xrightarrow{x} \theta_2) \quad \text{iff} \quad (\mathcal{R}, r') \models \theta_1 \xrightarrow{x} \theta_2 \text{ holds for all } r' \in \mathcal{R} \text{ such that } r \sim_\sigma r'.$$

When performing the action $\underline{b}$ in solving a coordination problem such as $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$ or $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$, process B must know that its current basic node satisfies the required precedence condition with respect to the node θ' at which A performs its action $\underline{a}$. This is formalized as follows.

Theorem 4.1. *Suppose that C sends A a "go" message at basic node σ_C in run $r \in \mathcal{R} = \mathcal{R}(\mathsf{P}, \gamma)$, and that B performs $\underline{b}$ at node σ in r. If P implements $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b} \rangle$ then $(\mathcal{R}, r) \models K_\sigma(\sigma_C \cdot A \xrightarrow{x} \sigma)$. Similarly, $(\mathcal{R}, r) \models K_\sigma(\sigma \xrightarrow{x} \sigma_C A)$ if P implements $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a} \rangle$.*

By Theorem 4.1, B cannot perform $\underline{b}$ in a protocol solving one of the coordination tasks of Definition 1.1 unless it knows that it is at a node satisfying an appropriate temporal precedence to the one at which A performs $\underline{a}$. Since such knowledge is also a sufficient condition for B's action, we can obtain an optimal solution for the coordination tasks by characterizing when the corresponding knowledge statements hold. So, an optimal protocol for B when performing the coordination tasks of Definition 1.1, is:

[7]It will suffice to define knowledge at basic nodes, here, since our analysis does not concern knowledge about what is known at other nodes. For a more general treatment, it is possible to define $(\mathcal{R}, r) \models K_\theta p$ to hold precisely if $(\mathcal{R}, r) \models K_\sigma p$ holds at $\sigma = \mathtt{basic}(\theta, r)$.

Protocol 4.2. *In local state ℓ, denoting $\sigma \triangleq (B,\ell)$: If B has not performed $\underline{b}$ yet, and C sends a "*go*" message at a basic node $\sigma_C \rightsquigarrow_r \sigma$, then:*

- *For* $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b}\rangle$*: If* $(\mathcal{R},r) \models K_\sigma(\sigma_C \cdot A \xrightarrow{x} \sigma)$*, then perform* $\underline{b}$.
- *For* $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a}\rangle$*: If* $(\mathcal{R},r) \models K_\sigma(\sigma \xrightarrow{x} \sigma_C \cdot A)$*, then perform* $\underline{b}$.

This description of the optimal protocols is made in terms of B's knowledge about timed precedence between nodes. Our goal is to translate this into a more concrete description, in terms of the communication pattern that is recorded in B's local state. We will do so at once for both problems by solving a more general problem. I.e., we will characterize the communication patterns that determine when $(\mathcal{R},r) \models K_\sigma(\theta_1 \xrightarrow{x} \theta_2)$ holds, for general nodes θ_1 and θ_2.

It can be shown (and will follow from our results) that in order to know that $\theta_1 \xrightarrow{x} \theta_2$, a node σ must know that a zigzag pattern of weight at least x connects these two nodes. In contrast to the case of message chains in asynchronous systems, information does not flow along a zigzag pattern. Indeed, it does not pass from the tail of a fork to its head, or vice-versa. The shape and existence of a zigzag pattern depends on whether or not the head of one fork occurs before the tail of its successor (e.g., at node D in Figure 2a). Thus, roughly speaking, the only way in which σ can observe that a zigzag pattern exists is by being informed of the ordering among adjacent forks. Moreover, if σ does not belong to the top fork in the pattern, then it must also be informed of the existence of this fork. We thus define:

Definition 4.3 (Visible Zigzag). Let σ be a node of run $r \in \mathcal{R}$, and let $Z = (F_1,\ldots,F_c)$ be a zigzag pattern in r. Then Z is called ***σ-visible in r*** if both (i) $\mathsf{head}(F_k) \rightsquigarrow_r \sigma$ for all $1 \le k \le c-1$, and (ii) $\mathsf{base}(F_c) = \langle \sigma', p'\rangle$ for a node $\sigma' \rightsquigarrow_r \sigma$.

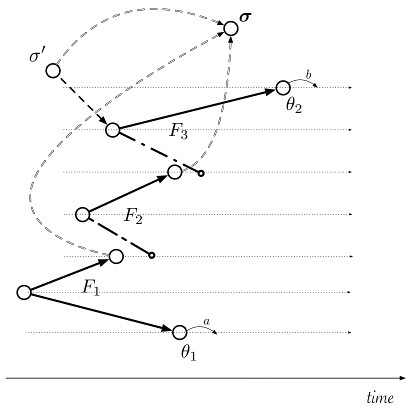


Figure 4: A σ-visible zigzag pattern from θ_1 to θ_2

In Figure 4 we see a σ-visible zigzag $Z = (F_1,F_2,F_3)$ from θ_1 to θ_2. Note that $\mathsf{head}(F_1) \rightsquigarrow_r \sigma$ and $\mathsf{head}(F_2) \rightsquigarrow_r \sigma$, and so σ knows that $\mathsf{tail}(F_2)$ doesn't appear before $\mathsf{head}(F_1)$, and $\mathsf{tail}(F_3)$ doesn't appear before $\mathsf{head}(F_2)$. We remark that the definition of a σ-visible zigzag does not require a path from the base of other forks to σ, because for all forks except the top one, there is a path consisting of a message from the base to the head and, by condition (i), a path from the fork's head to σ. We can now show:

Theorem 4.4 (Visible Zigzag Theorem). *Let $\mathcal{R} = \mathcal{R}(\mathsf{P},\gamma)$ and suppose that P is an FFIP. Moreover, let σ be a basic node of $r \in \mathcal{R}$, and let θ_1 and θ_2 be σ-recognized nodes in r, such that both $\mathsf{time}_r(\theta_1) > 0$ and $\mathsf{time}_r(\theta_2) > 0$. Then $(\mathcal{R},r) \models K_\sigma(\theta_1 \xrightarrow{x} \theta_2)$ iff there exists a σ-visible zigzag pattern Z from θ_1 to θ_2 in r with $\mathsf{wt}(Z) \ge x$.*

The Visible Zigzag Theorem provides a precise characterization of the pattern of communication that is necessary and sufficient for knowledge at σ of precedence among timepoints at distinct sites of the system. This is a fundamental aspect of information flow in bcm systems. The fact that a σ-visible pattern is sufficient for such knowledge appears reasonable given our analysis so far. The main technical challenge is to prove the converse: that such a pattern is also necessary.

We can now rephrase the optimal protocol defined before (Protocol 4.2), in terms of concrete communication patterns:

Protocol 4.5. *In local state ℓ, denoting $\sigma \triangleq (B,\ell)$: If B has not performed $\underline{b}$ yet, and C sends a "*go*" message at a basic node $\sigma_C \rightsquigarrow_r \sigma$, then:*

- *For* $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b}\rangle$*: If there is a σ-visible zigzag pattern Z in r from $\sigma_C \cdot A$ to σ with* $\mathsf{wt}(Z) \ge x$*, then perform* $\underline{b}$.
- *For* $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a}\rangle$*: If there is a σ-visible zigzag pattern Z in r from σ to $\sigma_C \cdot A$ with* $\mathsf{wt}(Z) \ge x$*, then perform* $\underline{b}$.

The visible zigzag patterns of Protocol 4.5 are instances of Figure 4, in which one of the endpoints of the pattern is σ in itself. The pattern for $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b}\rangle$ is illustrated in Figure 5. Note that there is no need for a separate message chain from $\mathsf{base}(F_c)$ to σ in this pattern, because $\mathsf{base}(F_c) \rightsquigarrow_r \sigma = \mathsf{head}(F_c)$ holds for the two-legged fork F_c, and so condition (ii) of Definition 4.3 is trivially guaranteed. In the pattern for the case of $\mathsf{Early}\langle \underline{b} \xrightarrow{x} \underline{a}\rangle$ we have that $\sigma = \mathsf{tail}(F_1) = \theta_1$. It contains all of the message chains depicted in Figure 4.

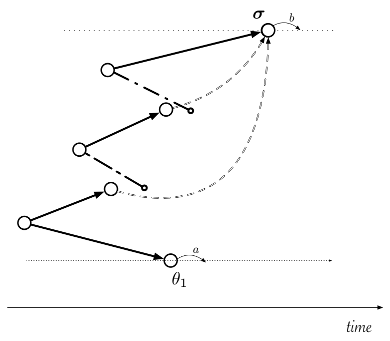


Figure 5: A visible zigzag pattern for $\mathsf{Late}\langle \underline{a} \xrightarrow{x} \underline{b}\rangle$

While the conditions described above for the optimal protocol are more figurative (communication patterns), there are actually simple algorithms to check for their truth (using a structure that is described next), but this is beyond the scope of this paper.

5 HIGHLIGHTS OF THE ANALYSIS

In this section we survey the general approach used for proving our main results. Of course, the essence of the analysis has to do with extracting knowledge about timing from the actual communication in a run, given the *a priori* bounds on message transmission times. This has been considered in the literature, for example, in the work on clock synchronization [1, 11, 12, 16, 24, 25, 31, 33, 35].

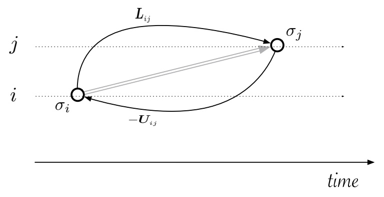


Figure 6: The bound edges created by a direct message

Inspired by [31, 33], we use a weighted graph to capture the timing guarantees provided by the system, and to reason about time differences between local timepoints in a given run r.

Definition 5.1. Given a run $r \in \mathcal{R}$, the ***basic bounds graph for r*** is a graph $G_B(r) = (V_B, E_B, w)$, where V_B are the basic nodes that appear in r. The edges of E_B are defined as follows: (a) If σ and σ' are i-nodes (for the same process i) and σ' is the successor of σ, then $(\sigma, \sigma') \in E_B$, and $w(\sigma, \sigma') = 1$. (b) If some message sent at an i-node σ_i in r is received at a j-node σ_j, then both $(\sigma_i, \sigma_j) \in E_B$ and $(\sigma_j, \sigma_i) \in E_B$, with $w(\sigma_i, \sigma_j) = \boldsymbol{L}_{ij}$ and $w(\sigma_j, \sigma_i) = -\boldsymbol{U}_{ij}$.

The basic bounds graph captures timed precedence information about the temporal relation among basic nodes: (a) is justified by the fact that successive nodes are at least one time step apart, while (b) embodies the upper and lower bounds on message transmission times. Figure 6 illustrates the edges of G_B that are induced according to case (b) by a single message delivery. It is straightforward to check (see, e.g., [31]) that

Lemma 5.2. *Let p be a path connecting nodes σ and σ' in $G_B(r)$. If $w(p) = x$, then $(\mathcal{R}, r) \models \sigma \xrightarrow{x} \sigma'$.*

Figure 7 highlights a path in the bounds graph that captures the timing guarantees implied by the zigzag pattern of Figure 2a via Lemma 5.2.

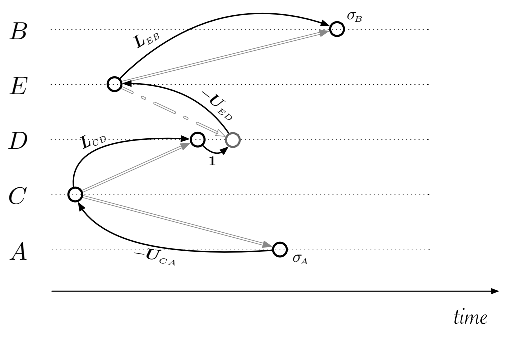


Figure 7: A path in the bounds graph justifying Equation (1)

In addition to imposing a precedence constraint, a path in $G_B(r)$ induces a zigzag pattern in the run r. More precisely:

Lemma 5.3. *If p is a path connecting nodes σ and σ' in $G_B(r)$, then there exists a zigzag pattern Z in r, from σ to σ', with* $\mathsf{wt}(Z) = \mathsf{wt}(p)$.

The faint lines in Figure 7 show the zigzag communication pattern underlying the path in $G_B(r)$, which is depicted by the solid lines.

We now give a sketch for the proof of Theorem 3.4. Assume that $\mathcal{R}$ supports $\sigma_1 \xrightarrow{x} \sigma_2$, for two basic nodes σ_1 and σ_2 that appear in r. By Lemma 5.2, each path between σ_1 and σ_2 in $G_B(r)$ defines a constraint on the difference of their times. I.e., if p is a path between σ_1 and σ_2, then $(\mathcal{R}, r) \models \sigma_1 \xrightarrow{\mathsf{wt}(p)} \sigma_2$. The longer the path is, the stronger the constraint. Thus, we are interested in finding the longest path from σ_1 to σ_2.

Assume that p is the longest path (between σ_1 and σ_2). Our main claim is that there exists a run $r' \in \mathcal{R}$ such that $G_B(r) = G_B(r')$, both σ_1 and σ_2 appear in r', and $\mathsf{time}_{r'}(\sigma_2) = \mathsf{time}_{r'}(\sigma_1) + \mathsf{wt}(p)$. This means that the constraint dictated by the longest path p is tight (a similar argument appears in [33]). By definition of *supports*, we obtain that $\mathsf{time}_{r'}(\sigma_2) \geq \mathsf{time}_{r'}(\sigma_1) + x$ and so $\mathsf{wt}(p) \geq x$. By Lemma 5.3 there exists in r a zigzag pattern Z from σ_1 to σ_2, with $\mathsf{wt}(Z) = \mathsf{wt}(p) \geq x$, just as stated in Theorem 3.4.

But what if $G_B(r)$ does not contain a path from σ_1 to σ_2? In such case we can show that there is a run $r'' \in \mathcal{R}$ containing σ_2, in which σ_1 doesn't appear. This contradicts the assumption that $\mathcal{R}$ supports $\sigma_1 \xrightarrow{x} \sigma_2$, since by definition of *supports* σ_1 and σ_2 must either both appear in r'', or neither should appear.

The proof shows that, for every node σ_2 in $G_B(r)$, there is a single run, r', in which, intuitively, every node of $G_B(r)$ is "delayed" as much as possible, relative to $\mathsf{time}_{r'}(\sigma_2)$. In other words, for every node σ' that has a path to σ_2 in $G_B(r)$ we will have that $\mathsf{time}_{r'}(\sigma_2) = \mathsf{time}_{r'}(\sigma') + \mathsf{wt}(p')$, where p' is the longest path from σ' to σ_2. Moreover, every node that doesn't have a path to σ_2 will not appear in r'. This proves Theorem 3.4.

5.1 The Extended Bounds Graph

The proof of Theorem 4.4 is similar in its nature to the proof of Theorem 3.4, but is much more complex. While Theorem 3.4 states the existence of a zigzag pattern following a general run property (*supports*), Theorem 4.4 deals with the knowledge of a specific node. In the previous proof we used $G_B(r)$. Essentially everything that can be deduced about the timing of events in a run r based on the combined information in all processes' histories is captured by $G_B(r)$. Figure 7, for example, presents a path in the bounds graph that justifies the analysis leading to Equation (1). However, $G_B(r)$ is defined by the entire run, and a process at a given basic node $\sigma = \langle i, \ell \rangle$ observes only a portion of this information that is generated by the nodes in $\mathsf{past}(r, \sigma)$, which we denote by $G_B(r, \sigma)$. This subgraph of $G_B(r)$ does not completely capture the timing information available to σ, however. For example, assume that an i-node σ_i and a j-node σ_j are both in $\mathsf{past}(r, \sigma)$, and assume that a message sent at σ_i to process j isn't received at any node in $\mathsf{past}(r, \sigma)$. We know that the node at which this message will be received, i.e. $\langle \sigma_i, [i, j] \rangle$, must appear in $G_B(r)$ later than σ_j. We also have that $\mathsf{time}_r(\sigma_i) + \boldsymbol{U}_{ij} \geq \mathsf{time}_r(\langle \sigma_i, [i, j] \rangle)$ by definition of $\boldsymbol{U}_{ij}$. Combining this with the requirement that $\mathrm{time}_r(\langle\sigma_i,[i,j]\rangle) \geq \mathrm{time}_r(\sigma_j)+1$ we have that $\mathrm{time}_r(\sigma_i) - \mathrm{time}_r(\sigma_j) \geq 1-\boldsymbol{U}_{ij}$, and thus $(\mathcal{R},r) \models \sigma_j \xrightarrow{1-\boldsymbol{U}_{ij}} \sigma_i$. In our setting processes follow an FFIP, and so the contents of $\mathrm{past}(r,\sigma)$ depend only on σ and not on r. So this precedence holds for any run r' containing σ. Such a run satisfies $r' \sim_\sigma r$, and we thus obtain that $(\mathcal{R},r) \models K_\sigma(\sigma_j \xrightarrow{1-\boldsymbol{U}_{ij}} \sigma_i)$. This time precedence does not correspond to a path in $G_B(r,\sigma)$, and so $G_B(r,\sigma)$ misses important information.

In order to fully capture the information available to a node σ based on its partial view of the run, we define an extended bounds graph based on the nodes of $\mathrm{past}(r,\sigma)$, to which we add n auxiliary nodes $\{\psi_1,\ldots,\psi_n\}$, one per process timeline. Intuitively, each node ψ_j represents the earliest among the nodes on j's timeline that are beyond view (intuitively "over the horizon") for σ. Messages that are sent to process j and not received at nodes from $\mathrm{past}(r,\sigma)$, will be received at ψ_j or later. This extended graph is denoted by $\boldsymbol{G_E(r,\sigma)}$. Three sets of edges E', E'' and E''' are added to the induced subgraph $G_B(r,\sigma)$ of $G_B(r)$ to obtain the extended graph: (a) E' consists of edges (σ_i,ψ_i) from the latest i-node in $\mathrm{past}(r,\sigma)$ to i's auxiliary node, with weight $w(\sigma_i,\psi_i)=1$; (b) if a message was sent from an i-node σ_i in $\mathrm{past}(r,\sigma)$ to process j and not delivered to a node in $\mathrm{past}(r,\sigma)$, then an edge (ψ_j,σ_i) is added to E'' with weight $-\boldsymbol{U}_{ij}$ as in the basic bounds graph. Finally, (c) the set E''' consists of edges (ψ_j,ψ_i) with weight $-\boldsymbol{U}_{ij}$ that are added for every channel $(i,j) \in \mathsf{Chans}$. (Intuitively, these edges are justified by the fact that the processes follow an FFIP, and so when a message will be delivered at a node beyond the view of σ, it will be sent to all neighbors in the Net graph.) Figure 8 illustrates the extended bounds graph $G_E(r,\sigma)$, for an i-node σ. The underlying network includes three processes, i, j and k, and communication channels (in both directions) between processes j and i, and between processes i and k. The figure highlights the four types of edges that appear in $G_E(r,\sigma)$. The shaded area depicts the $\mathrm{past}(r,\sigma)$ region. On the right are the auxiliary nodes ψ_i, ψ_j and ψ_k, one per process. Note that the bound edges to and from auxiliary nodes handle upper bounds only.

Now, the graph $G_E(r,\sigma)$ plays a similar role in the proof of Theorem 4.4 to the role of $G_B(r)$ in the proof of Theorem 3.4. Indeed, $G_E(r,\sigma)$ exhibits similar (albeit more complex) features to those of $G_B(r)$. For example, any path in $G_E(r,\sigma)$ whose endpoints are both basic nodes from $\mathrm{past}(r,\sigma)$ (and not auxiliary nodes), still defines a constraint between its endpoints (in any run $r' \sim_\sigma r$). It also defines a σ-visible zigzag in r with the same weight. Note the small differences: (1) The constraint here holds in any run $r' \sim_\sigma r$ (as for any such run, $G_E(r,\sigma) = G_E(r',\sigma)$), instead of any run r' with the same complete bounds graph (i.e $G_B(r') = G_B(r)$), and (2) the zigzag pattern is a *σ-visible* zigzag. Paths that start at, or end in, auxiliary nodes also exhibit important features, which are essential for the proof. The full details are beyond the scope of our presentation here and are available in [9].

Crucially, the extended bounds graph, $G_E(r,\sigma)$, can be used to construct valid runs of $\mathcal{R}$ with desirable properties. This is based on a careful assignment of times to nodes of $G_E(r,\sigma)$, in the following manner:

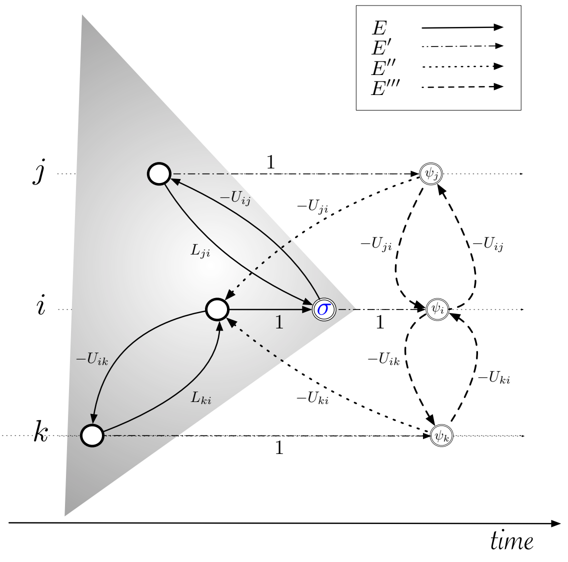


Figure 8: An illustration of the extended bounds graph $G_E(r,\sigma)$

Definition 5.4. Let σ be a basic node appearing in $r \in \mathcal{R}$. A valid timing function for $G_E(r,\sigma) = (V_E,E_E,w)$ is a function $T: V_E \to \mathbb{N}$, such that $T(\theta_1)+w(\theta_1,\theta_2) \leq T(\theta_2)$ holds for each $(\theta_1,\theta_2) \in E_E$.

Based on a valid timing function T for $G_E(r,\sigma) = (V_E,E_E,w)$, we can define a run $r' \sim_\sigma r$ of $\mathcal{R}$ in which the nodes of $\mathrm{past}(r,\sigma)$ appear at the prescribed times, and all the other nodes appear no earlier than the time of the auxiliary node ψ_j that belongs to their timeline j. This result is achieved by the fact that the bounds associated with auxiliary nodes make sure that nodes outside $\mathrm{past}(r,\sigma)$ won't appear too early relative to nodes from $\mathrm{past}(r,\sigma)$. (That, in turn, could force a message sent outside $\mathrm{past}(r,\sigma)$ to be received inside $\mathrm{past}(r,\sigma)$, which would modify σ's past and cause $r' \not\sim_\sigma r$).

6 DISCUSSION

The principles underlying coordination in purely asynchronous systems are by now fairly well understood, based on [22] and the four decades since it was published. Message chains play a central role in determining the ordering of events and coordinating their timing. More recently, the study of coordination in systems with global clocks was initiated by [5]. The current paper considers yet another timing model, the bcm model, in which there are no built-in timers and clock. Nevertheless, timing information can be gleaned from observed events, because there are upper and lower bounds on the message transmission times among processes. A direct use of bounds in such a model is the one illustrated in Figure 1: Given two message chains that start from the same point, if the sum of lower bounds on one is greater than the sum of upper bounds on the other, then the first message chain is guaranteed to end later than the second one. Indeed, the bounds can be used to provide a quantitative estimate of the time difference between these two events. We introduced the notion of a zigzag message pattern and showed that it provides another way to deduce the time precedence between events. The existence of an appropriate zigzag pattern was shown to be necessary and sufficient for the message pattern of an execution of the system to ensure that a given timed precedence among events is satisfied.

Interestingly, whereas it is possible to ensure that the existence of a message chain will be observed by the process receiving the chain's final message, this is not the case with general zigzag patterns. Information about the pattern's existence is distributed among the processes. In order to use a zigzag pattern in coordination, it is necessary for its relevant endpoints to obtain information about the order in which pivotal intermediate messages were delivered. Only then can a process know that the pattern exists, and hence to know that the precedence that the zigzag pattern implies is satisfied. Our analysis provides a characterization of when a precedence statement is known by a process at a given local state. This requires a *visible zigzag*, consisting of an appropriate zigzag pattern, as well as message chains informing the node about the pivotal parts of the zigzag pattern. A corollary of this is a characterization of patterns that allow coordinating actions according to Early and Late specifications.

The main mathematical structures underlying our analysis are the basic bounds graph and the extended bounds graphs presented in Section 5. In these, the start and end points of events are nodes, and the bounds are represented by weighted edges among these nodes. While the basic bounds graph has appeared in the analysis of clock synchronization (see, e.g., [33], in which it is used to capture synchronization even in the presence of clock drift), the extended bounds graph seems to be novel. It allows an analysis of the timing information at a node based on its subjective view of the computation. Events in its direct causal past, as well as the fact that events do not appear there, provide information on the timing and ordering of events.

As remarked in the Introduction, the bcm model can easily be adapted to capture bounds on the duration of other events as well. A natural setting that fits the bcm model is that of asynchronous, or self-timed, VLSI circuits, which are circuits that operate without clocks. In such settings, time bounds are often used to coordinate actions and ensure correctness of the computation. The typical way to do so is by using a simple fork as in Figure 1. Such forks are also the basis for correct operation in synchronous circuits, where extreme care is taken to ensure that clock inputs to different flip-flops are arranged to have very similar delays from a common source for the implementation of sequencing [36]. To the best of our knowledge, it is an open problem whether zigzag causality and our characterization of solutions to the Early and Late coordination problems may facilitate the design of new circuits.

REFERENCES

[1] H. Attiya, A. Herzberg, and S. Rajsbaum. Optimal clock synchronization under different delay assumptions. *SIAM J. Comput.*, 25, February 1996.

[2] I. Ben-Zvi and Y. Moses. On interactive knowledge with bounded communication. *Journal of Applied Non-Classical Logics*, 21(3-4):323–354, 2011.

[3] I. Ben-Zvi and Y. Moses. Agent-time epistemics and coordination. In *Indian Conference on Logic and Its Applications*, pages 97–108. Springer, 2013.

[4] I. Ben-Zvi and Y. Moses. The shape of reactive coordination tasks. In *Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2013), Chennai, India, January 7-9, 2013*, 2013.

[5] I. Ben-Zvi and Y. Moses. Beyond Lamport's *happened-before*: On time bounds and the ordering of events in distributed systems. *J. ACM*, 61(2):13:1–13:26, 2014.

[6] G.-q. Bi and M.-m. Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[7] B. A. Coan. A communication-efficient canonical form for fault-tolerant distributed protocols. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 63–72. ACM, 1986.

[8] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):8, 2013.

[9] A. Dan, R. Manohar, and Y. Moses. On using time without clocks via zigzag causality. arXiv.org article, http://arxiv.org/abs/1705.08627, 2017.

[10] B. Dickerson. Time in the power industry: how and why we use it. Arbiter Systems, technical report, http://www.arbiter.com/ftp/datasheets/TimeInThePowerIndustry.pdf, 2010.

[11] D. Dolev, J. Y. Halpern, B. B. Simons, and H. R. Strong. A new look at fault-tolerant network routing. *Information and Computation*, 72(3):180–196, 1987.

[12] D. Dolev, J. Y. Halpern, and H. R. Strong. On the possibility and impossibility of achieving clock synchronization. *Journal of Computer and System Sciences*, 32(2):230–250, 1986.

[13] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 2003.

[14] P. Giusto, G. Martin, and E. Harcourt. Reliable estimation of execution time of embedded software. In *Proceedings of the conference on Design, automation and test in Europe*, pages 580–589. IEEE Press, 2001.

[15] Y. A. Gonczarowski and Y. Moses. Timely common knowledge. In *Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2013), Chennai, India, January 7-9, 2013*, 2013.

[16] J. Y. Halpern, N. Megiddo, and A. Munshi. Optimal precision in the presence of uncertainty. *Journal of Complexity*, 1:170–196, 1985.

[17] K. Harris. An application of IEEE 1588 to industrial automation. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, 2008.

[18] IEEE TC 9. 1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2. *IEEE*, 2008.

[19] ITU-T G.8271/Y.1366. Time and phase synchronization aspects of packet networks. *ITU-T*, 2012.

[20] M. Konishi. Centrally synthesized maps of sensory space. *Trends in Neurosciences*, 9:163–168, 1986.

[21] H. Kopetz. *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.

[22] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[23] L. Lamport. Using time instead of timeout for fault-tolerant distributed systems. *ACM Trans. Program. Lang. Syst.*, 6(2):254–280, 1984.

[24] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, 1985.

[25] J. Lundelius and N. Lynch. An upper and lower bound for clock synchronization. *Information and control*, 62(2-3):190–204, 1984.

[26] N. A. Lynch and N. Shavit. Timing-based mutual exclusion. In *Proceedings of the Real-Time Systems Symposium - 1992, Phoenix, Arizona, USA, December 1992*, pages 2–11, 1992.

[27] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer. White rabbit: Sub-nanosecond timing distribution over ethernet. In *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, pages 1–5. IEEE, 2009.

[28] A. Morrison and Y. Afek. Temporally bounding TSO for fence-free asymmetric synchronization. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '15, Istanbul, Turkey, March 14-18, 2015*, pages 45–58, 2015.

[29] Y. Moses. Knowledge and communication (a tutorial). In Y. Moses, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. Fourth Conference*, pages 1–14. Morgan Kaufmann, San Francisco, Calif., 1992.

[30] Y. Moses. Relating knowledge and coordinated action: The knowledge of preconditions principle. In *Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015, Carnegie Mellon University, Pittsburgh, USA, June 4-6, 2015.*, pages 231–245, 2015.

[31] Y. Moses and B. Bloom. Knowledge, timed precedence and clocks. In *Proc. 13th ACM Symp. on Principles of Distributed Computing*, pages 294–303, 1994.

[32] C. J. Myers. *Asynchronous circuit design*. John Wiley & Sons, 2004.

[33] B. Patt-Shamir and S. Rajsbaum. A theory of clock synchronization (extended abstract). In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 810–819, New York, NY, USA, 1994. ACM.

[34] B. Simons. An overview of clock synchronization. In *Fault-Tolerant Distributed Computing*, pages 84–96. Springer, 1990.

[35] T. K. Srikanth and S. Toueg. Optimal clock synchronization. In *Proc. 4th ACM Symp. on Principles of Distributed Computing*, pages 71–86, 1985.

[36] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective (4th Edition)*. Pearson, 2010.