Causal Delivery Protocols in Real-time Systems: a Generic Model

Paulo Veríssimo
paulov@inesc.pt
INESC - IST
Technical University of Lisboa*

Abstract

The objective of this paper is to introduce a model for causal delivery protocols in real-time systems. We start by showing that temporal order properties of real-time protocols are independent of whether they are timer-driven or clock-driven, being instead related to their degree of synchronism, that we call steadiness. Then, we derive a set of correctness conditions for such protocols to secure causal delivery order. To achieve this objective, we use an extension of Lamport's model of time-stamp based order. We show that both timer- and clock-driven protocols have order correctness limits dictated by the environment and the target applications, and define those limits, through a set of working formulas. We show that in extremely adverse cases, timer-driven protocols will perform as well as clock-driven ones, due to the restrictions imposed on the operation of the latter, which is perhaps surprising. These results open the door to exploring new forms of communication in time-critical systems, for example, supporting clock- and time-driven communication, and event- and time-triggered operation. We expect that the results of this paper will give insight to that problem, and will be useful in real-life systems, such as distributed computer control.

1 Introduction

Communication protocols take different approaches to achieve order and reliability of message delivery. Reliable multi-participant— or multicast— protocols, for example, lie in two such classes, depending on the use they make of time:

- *Clock-driven* protocols [4, 6] are diffusion-based, and rely on the existence of a *clock*, in the sense of a global time-base— that is, an absolute global time reference;
- *Timer-driven*, or (global-) clock-less, protocols [3, 1] are acknowledgement-based protocols, that rely on local timers— i.e. relative time references.

*Instituto de Engenharia de Sistemas e Computadores, R. Alves Redol, 9 - 6^o - 1000 Lisboa - Portugal, Tel.+351-1-3100281. Fax: +351-1-525843. This work has been supported in part by the CEC, through Esprit Projects BROADCAST and DINAS, and JNICT, through Programs Ciência and STRIDE.

In a former paper [19], we have informally pointed out that timer-driven and clock-driven protocols are both able to serve distributed time-critical applications, not only from a time boundedness viewpoint, but also to secure temporal order. This point is relevant because the suitability of timer-driven protocols for real-time fault-tolerant applications has been sometimes questioned, mainly for two presumed reasons: their asynchrony; their inability to secure temporal (real-time) order[1].

In fact, most timer-driven protocols are asynchronous, in the sense of not having a known time bound to deliver messages. In contrast, their clock-driven counterparts are synchronous, since the afore-mentioned bound exists and is known. However, there is no fundamental reason for a timer-driven protocol not to be synchronous, if provided with the adequate structure and assumptions. The author was involved in the design of such a protocol in the DELTA-4 Esprit project, the AMp, which was formally specified and validated, and tested in real settings [22].

As for the second reason, it is well known that logical order protocols are blind to interactions made outside their control [10], as happens many often with distributed real-time systems. This created the notion of the necessity of clock-driven protocols and physical time-stamps to address the problem of cause-effect order. However, it will be shown that if a timer-driven protocol is synchronous, its ability to secure cause-effect order can be defined.

To finalise these opening remarks, one might question this concern with timer-driven protocols for real-time operation, given that clock-driven ones are around. First of all, it is not the author's intention to get rid of global clocks, a mandatory component in a time-critical system [9]. The idea is just not to oblige all ordered communication to be done via clock-driven protocols. The importance of this issue stems from the fact that synchronous timer-driven protocols are a useful alternative to clock-driven protocols in a number of situations, since they exhibit fast termination in absence of faults, and clock-driven behaviour can always be superimposed on top of a basic timer-driven communication infrastructure [13]. This allows to envisage time-critical systems with mixed behaviour, an interesting subject of research.

In consequence, the objective of this paper is to formalise a set of synchronism and ordering properties equally valid for timer-driven and clock-driven protocols. This model would confirm our point about suitability of timer-driven protocols from a theoretical viewpoint. On the other hand, it would contribute to a definition of correctness conditions, from the causal delivery viewpoint, valid for both classes of protocols (timer-driven and clock-driven). Then, applying those conditions to both classes of protocols in real-life time-critical settings should reveal the respective limits, which we do at the end of the paper.

To put the subject in context, we start, in the next section, by introducing some notation and briefly recapitulating a few notions about potential causality, temporal order and synchronism. Next we show, through our metrics of synchronism, the relation between synchronism of a protocol and its ability to secure temporal order. Having established that result, we show that timer-driven protocols are able to correctly order events and messages in time-critical systems, as their clock-driven counterparts. Then, in section 3, we establish the conditions for correctness of a distributed application, *vis-à-vis* time and order. In other words, given a certain system and execution environment, when

[1] We will explain these problems clearly in section 2.

does a given temporal order protocol support a given application correctly. In section 4, we derive the requirements imposed on the communication architecture to meet the correctness criteria, based on varying execution environment conditions. We show that in an adverse environment, the restrictions imposed on clock-driven protocols degrade their effectiveness to a point where timer-driven ones do just as well. On the other hand, for certain favourable environments, timer-driven protocols are sufficient. However, when high precision is necessary, only clock-driven protocols apply.

To the author's knowledge, this characterisation of protocol ordering ability— that is, the ability to preserve the order among events— with regard to the execution environment has not been addressed in previous papers. We consider this result relevant for two reasons: (i) it defines conditions for correct causal delivery with regard to application needs; (ii) it does it independently from protocol implementation, establishing the suitability of timer-driven protocols for temporal order. Besides any theoretical relevance that this work may have, these results open the door to exploring new forms of reliable communication in time-critical systems, using the best-fit protocol for a given problem. For example, supporting mixed event- and time-triggered operation. Up until now, it was not envisaged how to establish correctness conditions for these settings. We expect the results of this paper to give insight into that problem.

2 Order and synchronism

Ordering paradigms assume several forms in a system: total, potential causal, FIFO. Several protocols provide a total order, either real-time or not [4, 11, 3, 22]. The issue of providing a real-time total order is to define a bound on the time needed to secure it [7].

A potential causal order is a much more complex issue, and will be the focus of this paper (the FIFO order is a reduction of the potential causal order to a single sender). The natural order of events in a system is the *cause-effect* relation. Protocols, in order to secure it, tend to order messages that *may* be causally related. Before delving further into the discussion, let us present our system model.

System Model

We consider a system defined by a set P of participants noted p, q, r, ... A participant can be a computing process, a sensor or actuator controller, but also some component of an external physical system under control. The system is synchronous, in the sense that there are known bounds on: processing speed; message transmission delay; local clock rate drift.

We model the history of the participants as a sequence of events: e_p^i is the i^{th} event of participant $p \in P$. Event e_p^i can be a local event or an external event. We consider two types of external events, action, ACT, and observation, OBS:

- $ACT_p(a)$ is an event by which the participant p takes some external action a, be it an I/O actuation or a message send;
- $OBS_p(a)$ is an event by which the participant p observes some external action a, be it an observation of the environment or the delivery of a message sent by a

participant q.

An event *precedes* another event if it could have caused the latter, that is, if it is causally related to it. We introduce the precedence relation $\xrightarrow{p}$ between events for our model, which is a generalisation of the "happened before" relation first introduced by Lamport for message-based systems [10], to arbitrary events, including those taking place in physical processes (oven, steel mill, car ABS system, plane steering system, etc.):

Definition 1 Precedence. *An event precedes ($\xrightarrow{p}$) another event if one of the following conditions hold:*

- *for $j > i$, $e_p^i \xrightarrow{p} e_p^j$;*
- *for $p \neq q$, $e_p^i \xrightarrow{p} e_q^j$ if e_p^i is an external event $ACT_p(a)$ and e_q^j an external event $OBS_q(a)$;*
- *for $p \neq q$, $e_p^i \xrightarrow{p} e_q^j$ if there exists an event e such that $e_p^i \xrightarrow{p} e$ and $e \xrightarrow{p} e_q^j$.*

□

For the sake of representing their place in a (newtonian) time-line, whenever necessary, we associate physical time-stamps to events: $t(e_p^i)$ is the time-stamp of e_p^i, as defined by an omniscient external observer.

Causal Delivery

There are several ways of implementing *causal delivery*, i.e. guaranteeing that messages are delivered in their precedence order[2] in a distributed system. Given the system model of the previous section, we note $send_p(m)$ the event corresponding to the transmission of m by p, and $deliver_q(m)$ the delivery of m to q. For simplicity of notation, we may omit the subscript, when there is no risk of ambiguity. The $send(m)$ and $deliver(m)$ events are, respectively, ACT and OBS events.

Definition 2 Causal delivery. *Consider two messages m_1, m_2 sent by p, resp q, to the same destination participant r. Causal delivery ensures that if $send_p(m_1) \xrightarrow{p} send_q(m_2)$ then $deliver_r(m_1) \xrightarrow{p} deliver_r(m_2)$, i.e. m_1 is delivered to r before m_2.* □

Well-known causal delivery protocols are the "causal broadcast/multicast" protocols using: piggybacking [2], context graphs [12], or vector clocks [3, 14].

These protocols, adequate for asynchronous systems, deliver messages according to a *logical* ordering [2]: a message m_1 is said to logically precede ($\xrightarrow{l}$) m_2 **if** *m_1 is sent before m_2, by the same participant* **or** *m_1 is delivered to the sender of m_2 before it sends m_2* **or** *there exists a message m_3, such that, $m_1 \xrightarrow{l} m_3$ and $m_3 \xrightarrow{l} m_2$.*

The fact that an implementation based on logical ordering can achieve causal delivery is based on a simple observation: if participants only exchange information by sending and receiving messages through a given protocol, any relation of potential causality must be created through those messages.

[2]Or *potential causal* order, sometimes only called causal, for simplicity.

Whenever there is information flowing between participants which is not controlled by the ordering protocol, so to speak in a *clandestine* manner, there may be ordering anomalies observable by the participants. That is, participants may have a perception of the system evolution that, for example, seems to violate the cause-effect relation. This was noted by Lamport [10], in the form of hidden communication channels.

However, as we have first pointed out in [21], this situation is very common in distributed real-time systems, under another more subtile albeit disturbing form: clandestine information paths (or hidden channels) through the physical environment (feedback loops, etc.), established by the input/output actions performed by the computer control system. An example of anomalous behaviour in a control system is given in [16]. Formally, these hidden channels are represented in our model by the existence of ACT and OBS events inside a causality chain which do not correspond to sends and deliveries of the logical order protocol. For that protocol, that chain appears broken.

Temporal order

One solution for these situations consists in using protocols that deliver messages according to their *temporal order*. A message m_1 is said to temporally precede ($\stackrel{t}{\rightarrow}$) m_2 **if** *the* send *event of* m_1 *physically occurs before that of* m_2*, in a newtonian time-frame, i.e.* $t(send(m_1)) < t(send(m_2))$. Given that for an event to cause another, it must happen before the latter, a clock-driven protocol time-stamping messages at send time and delivering them in temporal order will, under certain conditions, achieve causal delivery, as first proposed by Lamport [10].

However, our remark to this approach is that, given the precision achievable by most synchronised clocks, one will be ordering too much in most of the settings, *vis-à-vis* the primary objective of causal delivery, which is to represent precedence (potential causality). The reason is that in a distributed computer system or in a physical process, it takes a finite amount of time for an input event (OBS, eg. deliver) to *cause* an output event (ACT, eg. send): the time for an information to travel from one site to the other; the execution time of a computer process; the feedback time of a control loop in a physical process.

Supposing there is a known such minimum time δ_t for a given system, then only messages separated by more than δ_t may be causally related in that system. In consequence, it is of no utility to order messages with a smaller separation. We will formalise this conjecture in what follows. Namely, we will characterise the relevant ordering protocols, and the conditions to secure precedence.

Protocol synchronism

A propos the remark just made, we recall a definition that will assist us later in developing our reasoning [22]:

Definition 3 **δ_t-precedence** ($\stackrel{\delta_t}{\rightarrow}$)*: An event e is said to δ_t-precede an event e',* $e \stackrel{\delta_t}{\rightarrow} e'$, **if** $t(e') - t(e) > \delta_t$. □

Next, we need a metrics for protocol synchronism that is independent from protocol implementation, and thus applicable to any protocol, timer-driven or clock-driven, as

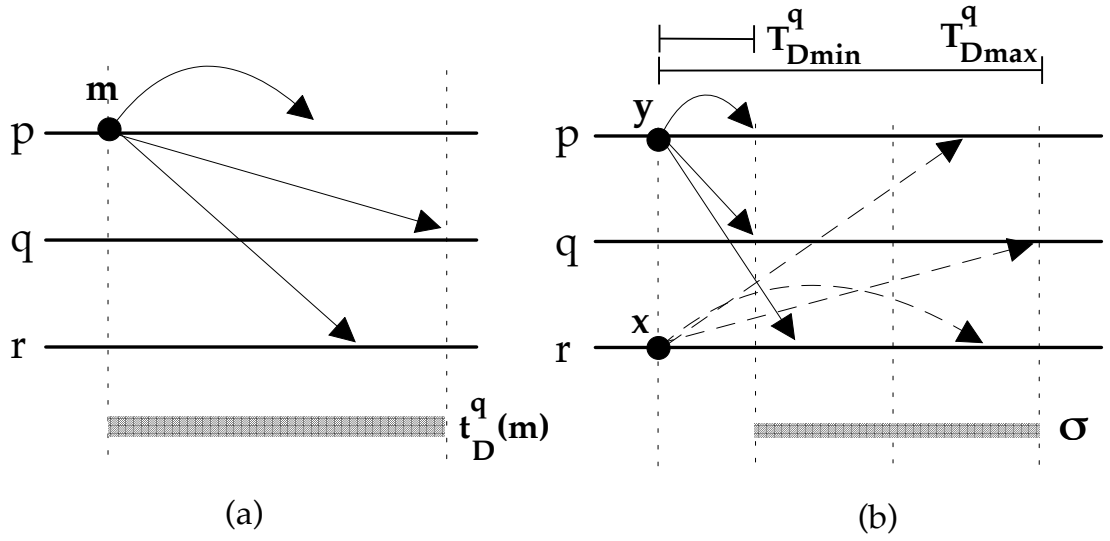

Figure 1: Protocol Synchronism metrics — (a) delivery time (t_D); (b) steadiness (σ)

proposed in the beginning. We recapitulate the metrics for synchronism appearing in [20], with a slight modification that makes it simpler to handle. We note $DELIV(p)$ the set of messages delivered to a participant p during an execution. Moreover:

- *Delivery time*, $t_D^p(m)$, of message $m \in DELIV(p)$, is defined as the interval between the $send(m)$ event, and the $deliver_p(m)$ event[3] at p, i.e. $t_D^p(m) = t(deliver_p(m)) - t(send(m))$;
- $T^p_{Dmax} = \max_{m \in DELIV(p)} (t_D^p(m))$, the maximum delivery time for participant p;
- $T^p_{Dmin} = \min_{m \in DELIV(p)} (t_D^p(m))$, the minimum delivery time for participant p;

We define the *steadiness* σ of a communication protocol[4]:

Definition 4 Steadiness *of a protocol, σ, is the greatest difference, for all participants p, between T^p_{Dmax} and T^p_{Dmin}:* $\sigma = \max_p (T^p_{Dmax} - T^p_{Dmin})$ □

These definitions are exemplified in figure 1, where supposedly the runs **x** and **y** in figure 1b yield respectively the maximum and minimum delivery times as per definition 4.

Steadiness should not be confused with *temporal uncertainty*, introduced in [7]. Temporal uncertainty is the absolute greatest difference $\max_{p,q} (T^p_{Dmax} - T^q_{Dmin})$. In some protocols (eg. Δ-protocols), steadiness is not equal to temporal uncertainty, because they never yield those extremes in the same protocol run or at the same participant. A proof and discussion of the implications is made in appendix (cf. § B.2).

[3] It is usual in reliable communication, to differentiate between `receive`, the event of a message arriving at a site, and `deliver`, the event of it being delivered to the upper layer.

[4] For readers aware of the work in [20], steadiness was defined in a different way, in terms of *execution* times, and ordering properties were defined with the help of a second variable, *tightness*, τ. $\sigma + \tau$ is formally equivalent to what is meant by σ in this paper. This slight modification makes it simpler to treat our problem.

3 Execution correctness conditions

A first result is that the temporal ordering properties of a protocol are related to its synchronism, that is, a protocol with σ steadiness delivers messages separated by more than σ according to their temporal order:

Theorem 1 *A protocol with σ steadiness delivers messages m_1 and m_2 such that $send(m_1) \xrightarrow{\sigma} send(m_2)$, in temporal order.*

PROOF

Consider two messages m_1 and m_2 sent to a common participant q, and assume $send(m_1) \xrightarrow{\sigma} send(m_2)$, i.e. $t(send(m_2)) - t(send(m_1)) > \sigma$ (property (1)).
By definition, $t(send(m_1)) + t_D^q(m_1) = t(deliver_q(m_1))$, and the same holds for m_2 (property (2)).
If $t_D^q(m_1) \leq t_D^q(m_2)$ we trivially have $t(deliver_q(m_1)) < t(deliver_q(m_2))$.
Assume thus $t_D^q(m_1) > t_D^q(m_2)$. Because of the σ steadiness of the protocol, we have $t_D^q(m_1) - t_D^q(m_2) \leq \sigma$. Together with property (1) this leads to $t(send(m_2)) - t(send(m_1)) > t_D^q(m_1) - t_D^q(m_2)$, and by property (2), to $t(deliver_q(m_1)) < t(deliver_q(m_2))$. □

Observe that theorem 1 guarantees the following implication: $send(m_1) \xrightarrow{\sigma} send(m_2) \Rightarrow deliver(m_1) \not\rightarrow deliver(m_2)$. However, that is not enough to guarantee causal delivery as per definition 2: there may be messages for which $send(m_1) \not\rightarrow send(m_2)$ holds, whereas $send(m_1) \xrightarrow{\sigma} send(m_2)$ does not. For example, ACT/OBS pairs through the environment may establish the first implication, while they are fast enough that the second does not hold[5]. In consequence, the protocol may not order these messages, violating causal delivery. So, we define below an execution model where the property $[(send(m_1) \not\rightarrow send(m_2)) \Rightarrow (send(m_1) \xrightarrow{\sigma} send(m_2))]$ holds.

Before proceeding, note however that there is an important result from this section: from this point on, whatever we derive will apply equally to clock-driven and timer-driven protocols. What matters is that they must be synchronous and thus have known steadiness.

A few examples about temporal order

Let us exemplify what it means for an application supported by a temporal order protocol to execute correctly.
Consider the example of two participants P_A and P_B at different sites, competing for a resource controlled by P_C at a third site. They both try to grab it approximately at the same time, by sending messages m_A and m_B through a temporal order protocol. Suppose P_A sends first, but P_B would get the resource, because the protocol ordered m_B before m_A. The fact is that this would only be anomalous if P_A could know it had requested it first. This might only happen if $m_A \not\rightarrow m_B$, according to the definition of precedence (Def. 1). In other words, if there would be time for an information, departing from A when $send(m_A)$ occurs, to reach B, and be processed before $send(m_B)$ occurs.

This example shows that not all messages have to be ordered by the physical order of the $send$ requests. That is only required when two *send* events are time-like separated, by

[5]Because the length of the ACT/OBS chain is shorter than σ.

more than the time it takes to overcome their space-like separation[6].

Since in our model we make no assumptions about the way information is propagated— it can even travel through the physical process— a conservative figure to represent the above-mentioned space-like separation, is the absolute minimum **propagation delay** between any two participants in the system. As noted before, this delay is not necessarily related to network message passing, it can be concerned with propagation through a physical process under control— for example, from a computer controlling an actuator, to another computer reading from a sensor in that process loop.

Suppose now that the request from P_A, if received at P_B, should cause P_B *not* to request, after being interpreted by a processing step with a duration of $5ms$. If the participants were separated by, say, 100 meter, their space-like separation would thus be $\simeq 300ns$ at the speed of light (the minimum propagation delay). In consequence, P_A would have to request, in real time, more than $300ns + 5ms$ before P_B, to *cause* the inhibition of P_B. Otherwise, the message-passing subsystem could order m_B before m_A— against their physical order— and the system would still behave correctly, since no participant has means to confirm or infirm such an ordering.

In fact, this second example shows that it is of no use to have a protocol order two messages because they are just separated by more than the space-like separation— that is, $300ns$ in this case— since a process needs time to generate the causal relation between them— that is, $5ms$ in this case— and we may add that time to the causal chain. So, it is also useful to represent the time-like separation between processing steps made by the same participant, that we introduce as **local granularity**, the minimum interval between any two related consecutive events in the time-line of a participant.

The μ parameters

The examples above served to exemplify what happens with distributed real-time programs in general: they perform execution steps within bounded delays and exchange messages and/or perform input/outputs in result of those computations; they have a limited capability (in the time domain) of acquiring information and producing responses, formalised below.

Definition 5 Local granularity, μ_t, *of a system, is the minimum delay between two consecutive events in any participant* p, e_p^i *and* e_p^{i+1}: $\forall p, \forall i \quad t(e_p^{i+1}) - t(e_p^i) \geq \mu_t$ □

Definition 6 Propagation delay, μ_s, *of a system, is the minimum delay between an* $ACT_p(a)$ *event and the corresponding* $OBS_q(a)$ *event:* $\forall p, q, p \neq q \quad t(OBS_q(a)) - t(ACT_p(a)) \geq \mu_s$ □

Causal Delivery conditions

Finally, we are going to define the conditions under which a given temporal order protocol performs causal delivery, according to the definition of the precedence relation $\not\rightarrow$ made earlier. The problem as we see it has two facets:

[6] "Time-like" is measured in the time coordinate, "space-like" concerns the space coordinates, in Relativity jargon.

- the ability of the protocol proper to represent precedence, i.e. implement causal delivery of messages, ignoring hidden channels;
- the ability to do so with the additional problem presented by hidden channels.

We will call the environments where the protocol has to cope with hidden channels, *adverse*, and *favourable*, if otherwise. Please note two things. Firstly, in our model, a hidden channel can be *any form of information propagation* (not just fast network communications), either in the controlling computer system or in the controlled physical system. Secondly, a hidden channel only presents a problem *for the protocol* when it is faster than the latter. We formalise the two situations above:

- *adverse environments*— environments where the channels external to the protocol can deliver information faster than the protocol itself; in consequence, nothing can be said about the lower bound of the propagation delay μ_s, except that it is not defined by the protocol's fastest execution, but by the fastest hidden channel:
$$\mu_s < \min_p (T^p_{Dmin})$$
- *favourable environments*— environments where the channels external to the protocol are slower than or as fast as the latter, or do not exist at all; in consequence, the propagation delay is defined by the protocol's fastest execution[7]:
$$\mu_s = \min_p (T^p_{Dmin})$$

Adverse Environments

The following theorem is valid for the general case, that we called "adverse", where hidden channels exist and can have any speed[8].

Theorem 2 *Given a system with local granularity μ_t and propagation delay μ_s, a protocol such that $\mu_t > \max_u (T^u_{Dmax}) - \mu_s$, ensures causal delivery order.*

The proof of theorem 2 is made evident by figure 2.

PROOF

Consider a participant r and two messages m_1, m_2 received by r such that m_1 is received before m_2. Assume m_1 is sent by some participant q and m_2 by some participant p:

i) $q \neq p$.

Let $e^i_p \equiv send_p(m_2)$. If there exists a relation $send_q(m_1) \not\rightarrow send_p(m_2)$, then necessarily $send_q(m_1) \not\rightarrow e^{i-1}_p$.

Consider the causal chain $send_q(m_1) \not\rightarrow e^{i-1}_p \not\rightarrow send_p(m_2) \not\rightarrow deliver_r(m_2)$. Its shortest time length occurs when event e^{i-1}_p is $deliver_p(m_1)$. The length of the chain is thus greater or equal to $\mu_t+2\mu_s$, as shown in figure 2. The time length of the direct causal chain $send_q(m_1) \not\rightarrow deliver_r(m_1)$ is less or equal to T^r_{Dmax}, and thus, at most $\max_u (T^u_{Dmax})$. In consequence, by hypothesis it is less than $\mu_s + \mu_t$, i.e. less than $\mu_t + 2\mu_s$. Thus m_1 is received before m_2.

[7] At first glance, the equal sign might seem to ignore the "greater than" situation. However, note that μ_s is the minimum propagation delay, *including* message transmissions. "Favourable" just means that nothing in the system delivers faster than the fastest protocol execution.

[8] Well, up to the speed of light.

ii) q = p.

The time length of the direct causal chain $send_p(m_1) \xrightarrow{p} deliver_r(m_1)$ is less or equal to T^r_{Dmax}, and thus, at most $\max_u (T^u_{Dmax})$. In consequence, by hypothesis it is less than $\mu_s + \mu_t$. If m_2 is sent after m_1, the time length of the causal chain $send_p(m_1) \xrightarrow{p} send_p(m_2) \xrightarrow{p} deliver_r(m_2)$ is greater or equal to $\mu_s + \mu_t$. Message m_1 is thus received before message m_2. □

Favourable Environments

For favourable environments, the result of theorem 2 can be tightened by the introduction of condition $\mu_s = \min_u (T^u_{Dmin})$. This is beneficial since it lowers the allowed minimum value of μ_t.

Theorem 3 *Given a system with local granularity μ_t and propagation delay μ_s, and a protocol with steadiness σ and an absolute minimum delivery time of $\min_u (T^u_{Dmin})$, such that $\mu_s = \min_u (T^u_{Dmin})$ and $\mu_t > \sigma$, that protocol ensures causal delivery order.*

PROOF

The proof style follows that of theorem 2.

i) q ≠ p.

In this case (figure 2, left), the length of the causal chain L1, $send_q(m_1) \xrightarrow{p} deliver_p(m_1) \xrightarrow{p} send_p(m_2) \xrightarrow{p} deliver_r(m_2)$, is

$$L1 \geq T^p_{Dmin} + \mu_t + T^r_{Dmin}$$

Considering $\mu_t > \sigma$ by hypothesis,

$$L1 > T^p_{Dmin} + \sigma + T^r_{Dmin}$$

The length of the direct causal chain L2, $send_q(m_1) \xrightarrow{p} deliver_r(m_1)$, is

$$L2 < T^r_{Dmax}$$

Comparing L1 and L2,

$$L1 - L2 > \sigma + T^p_{Dmin} + T^r_{Dmin} - T^r_{Dmax}$$

The highest value of $T^r_{Dmax} - T^r_{Dmin}$ is σ (definition 4) and the smallest value of T^p_{Dmin} is $\min_u (T^u_{Dmin})$:

$$L1 - L2 > \min_u (T^u_{Dmin})$$

So, L1 is longer than L2, and causal delivery is secured (m_1 is received before m_2).

ii) q = p.

In this case (figure 2, right), by hypothesis, the length of the direct causal chain $send_p(m_1) \xrightarrow{p} deliver_r(m_1)$ is

$$L2 \leq T^r_{Dmax}$$

The length of the causal chain $send_p(m_1) \xrightarrow{p} send_p(m_2) \xrightarrow{p} deliver_r(m_2)$ is

$$L1 \geq \mu_t + T^r_{Dmin}$$

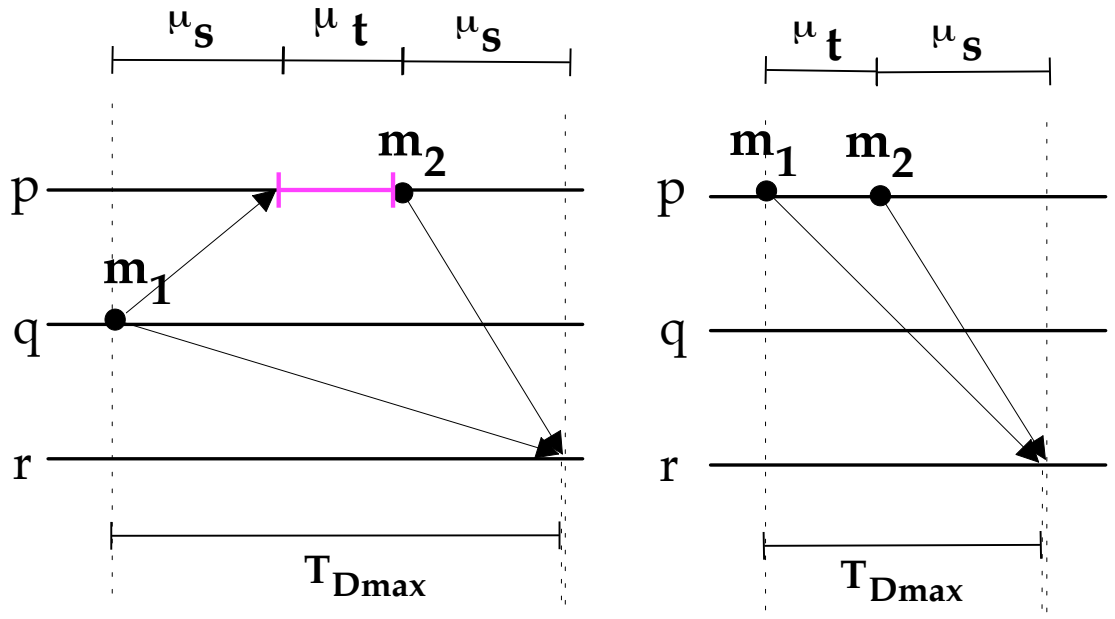

Figure 2: Correctness of temporal order implemented by protocols with σ steadiness (T_{Dmax} stands for $\max_u\ (T^u_{Dmax})$)

Considering $\mu_t > \sigma$ by hypothesis,

$$L1 > \sigma + T^r_{Dmin}$$

Comparing L1 and L2,

$$L1 - L2 > \sigma + T^r_{Dmin} - T^r_{Dmax}$$

The highest value of $T^r_{Dmax} - T^r_{Dmin}$ is σ (definition 4):

$$L1 - L2 > 0$$

So, L1 is longer than L2, and causal delivery is secured (m_1 is received before m_2). □

To duly appreciate the general result of theorems 2 and 3, and just for the sake of example, consider a system exhibiting a propagation delay of $\mu_s = 100\mu s$, and a distributed real-time control program with a granularity of $\mu_t = 100ms$, requiring temporal order to be secured. A protocol with a steadiness as coarse as $\sigma = 100ms$ and $T^i_{Dmin} \leq 100\mu s$ would do. An implication of the results of this section, is that the door is open for synchronous timer-driven protocols to implement temporal orderings and causal delivery in real-time applications. It remains to be seen, in the next section, what the limits to this ability are. Before that, a final word about our model:

> We have shown that, for environments where both I/O operations and message exchanges coexist, these message exchanges respect causal delivery. Nothing was said about the order in which ACT and OBS take place, only about the speed with which they propagate. For example, if an actuation $ACT(a)$ takes

place at a participant p, who later issues a message m, $OBS(a)$ can perfectly take place after $deliv(m)$ at another participant q, even if $ACT(a) \not\xrightarrow{p} send(m)$.

However, this is not disturbing. Firstly, it is easy to see that the fact that an event is observed late can never disturb causal delivery of messages. Secondly, if we were concerned about an anomaly caused by q observing a before m, then it would be possible to think of a protocol where m could convey information about the events that caused it. In fact, that is the classical scenario in distributed control, where subsequent to an actuation, the actuator informs a supervisor, through the network, as exemplified in [20][9]. The supervisor, upon eventually observing the effect of that actuation, would already know about it.

The really anomalous situation, as we pointed out in the beginning of this paper, would consist in $send(m)$ taking place before $ACT(a)$ at p, and being delivered later than $OBS(a)$ at q. This would indeed violate causal delivery of messages. Its avoidance was the subject of this section, namely the conditions of theorems 2 and 3. We believe this model can be extended with adequate protocols, towards a global causal order of events (messages and I/O), but that is a subject of future research.

4 Requirements on communication

Clock-driven protocols normally yield tight-synchrony (small σ, with regard to to T_{Dmax}), whereas their timer-driven counterparts yield loose-synchrony (moderate to large σ, which may be of the order of T_{Dmax}). In consequence, it is important to discuss the practical implications of the results of the previous sections. The intention is to show the reader and/or implementor that the results of the paper have indeed practical relevance, and to provide him/her with a set of working formulas. We do not present proofs of the working formulas we will derive, since they are easy to follow from theorems 2 and 3, and the steadiness derivations below.

Steadiness of some types of protocols

From steadiness one can establish ordering properties of protocols. In consequence, we determine the steadiness of a few synchronous protocols. We refer to their operation in a sketchy way, assuming the reader is familiar with the cited protocols. A list of the steadiness values is given in the table of figure 3, and the proof of their derivation is given in appendix (cf. § B.1).

Since messages are scheduled for transmission immediately after the request, the clock-driven protocol in [4] is event-triggered, as opposed to TTP, analysed next, which is time-triggered. Such a clock-driven protocol, also called Δ-protocol, delivers messages time-stamped $T(m)$ at $C_r = T(m) + \Delta$ on the recipient's clock. Transmission reliability is achieved by diffusion of the message through several paths. The constant Δ is system-dependent [4]: $\Delta = T_{Xmax} + \pi$, where T_{Xmax} is the maximum network delivery delay.

[9] In page 475.

STEADINESS	
Clock-driven Event-Triggered [AAS]	$\sigma = \pi + g$
Clock-driven Time-Triggered [MARS]	$\sigma = \pi$
Timer-driven Event-Triggered [DELTA-4]	$\sigma = 10.T^i_{Dmin}$

Figure 3: Steadiness of a few known real-time protocols

Introducing granularity of the clocks [8], the constant Δ becomes $\Delta = T_{Xmax} + \pi + g$ (Cristian [4] assumed continuous clocks, so in his paper, $g = 0$). Since clocks may be as much as π apart (the precision), two runs of the protocol from different senders may not have the same delivery time to a given recipient. Steadiness, as discussed in the appendix, is $\sigma = \pi + g$.

Time-triggered protocols, such as the TTP protocol described in [6], are clock-driven protocols whereby participants only transmit at pre-specified periodic instants, driven by the global clock. TTP is thus a TDMA protocol. What the protocol does exactly is to write directly on memory, immediately data arrives at a site, and this, taking into account that every message is sent twice, and one transmission may fail. TTP being so embedded in the specialised approach of the MARS system, makes it difficult to establish general results. However, given that MARS is completely time-triggered, the new data is used in synchrony by all sites at the start of the next "computing period", defined over a periodic time-base called action lattice. In consequence, and for the benefit of generality, we derive a model of time-triggered, clock-driven protocols:

- transmission requests are served at the start of the next period of the action lattice;
- the protocol delivers the messages sent in period G_i of the lattice, time-stamped $T(G_i)$, at $C_r = T(G_i) + \Delta$ on the recipient's clock;
- the constant Δ is $\Delta = T_{Wmax} + T_{Xmax} + \pi + g$, where: T_{Wmax} is the maximum waiting time for a transmission, equal to a lattice period; T_{Xmax} is the maximum network delivery delay (for example, extremely short in MARS— two LAN round-trip times); π and g, the precision and the granularity of the global clock;

Based on this model, we derive the parameters for the protocol (cf. § B.1), amongst which steadiness, whose expression is $\sigma = \pi$.

The last example concerns a timer-driven protocol, such as the AMp protocol used in the DELTA-4 system [22]. The protocol executes in two phases. Only one component may fail during a run. The network delays may vary with load. The sender, in the first phase, sends the message using transmit-with-reply rounds, with duration T_{XWR}, and is repeated at most $k + 1$ times, to tolerate up to k omission faults. In the second, it sends the decision to deliver the message to all recipients, when the first phase succeeds— i.e. when all replies come— in a non-acknowledged datagram, with duration T_X. After the first phase concludes, recipients set-up a timeout, $T_{WaitDecision}$, to detect a missing

decision frame. If the timeout expires, they request a decision from the sender, which they do up to $k+1$ times, after which they suspect sender failure. A two-phase group membership protocol, accumulating the function of termination, is immediately started, which will finally secure message delivery: *Step1* collects the pending message context; *Step2* terminates and establishes the new group view. A delivery time analysis of AMp, discussed in the appendix, reveals that there is a ratio of 1:11 between the AMp minimum and maximum delivery times, which yields $\sigma = 10.T^i_{Dmin}$. This may vary in other timer-driven protocols, depending on assumptions about failures and load variation.

4.1 Protocol behaviour in several environments

Before we proceed, it is important to stress two points:

- The framework of this paper is the absence of any restriction to the semantics of participants interactions. As such, an ordering securing precedence must be observed, such as that provided by protocols implementing causal delivery (Def. 2). The fact that real-life problems can sometimes benefit from assumptions tending to relax ordering requirements [16] does not invalidate the need for addressing the cases where that cannot be done.

- One important characteristic of the timeliness of a real-time process is its minimum response time, i.e. the shortest delay to produce an output (eg. ACT), given an input (eg. OBS). This is represented by μ_t, in our model. As such, the main objective of the sequel of this section is to find expressions for μ_t, in order to find out, for each communication environment, the kind of applications it can serve.

- For example, we show that in an adverse environment, the restrictions imposed on clock-driven protocols degrade their effectiveness to a point where timer-driven ones do just as well. On the other hand, for certain favourable environments, timer-driven protocols are sufficient. However, when high precision is necessary, only clock-driven protocols apply.

Clock-driven protocols in adverse environments

Let us recall that an environment is adverse to the ordering protocol, if it has hidden communication channels that are faster than the protocol itself.

a) Considering a worst-case scenario of $\mu_s = 0$, from theorem 2 we will have in general $\mu_t > T^i_{Dmax}$. So, adverse environments are only tractable if the applications are "slower" than the protocol. From the appendix (cf. § B.1), we obtain $T^i_{Dmax} = \Delta + \pi$, for clock-driven protocols. In consequence, a general ($\mu_s = 0$) working formula for clock-driven protocols in adverse environments is:

$$\boxed{\mu_t > \Delta + \pi}$$

b) Let us consider a slightly less pessimistic scenario, where though the physical environment may exhibit undefined propagation delay, we have an I/O subsystem which takes care of handling observations and actuations correctly, with regard to temporal order. In the computing part, we can make the assumption that hidden channels are all network-based, having as a lower bound, the propagation delay of the fastest datagram:

E-T CLOCK-DRIVEN PROTOCOLS	WORK. FORMULAS
adverse environments ($\mu_s = 0$)	$\mu_t > \Delta + \pi$
adverse environments and all-network hidden channels ($\mu_s = T_{Xmin}$)	$\mu_t > 3\pi + g - k$
using very precise clocks	$\mu_t > 3\pi + g - k$ $\pi \ll T_{Xmin}$
favourable environments ($\mu_s = \min_u (T^u_{Dmin})$)	$\mu_t > \pi + g$

Figure 4: Working formulas for clock-driven protocols

$\mu_s = T_{Xmin}$ (but still $T_{Xmin} < T^i_{Dmin}$). From theorem 2, we obtain $\mu_t + T_{Xmin} > T^i_{Dmax}$. On the other hand, we have $\Delta = T_{Nmax} + \pi + g$, where T_{Nmax} is the maximum generic network delivery delay— T_{Xmax}, for event-triggered systems, or $T_{Wmax} + T_{Xmax}$, for time-triggered systems (cf. beginning of section 4). Thus:

$$\mu_t + T_{Xmin} > \Delta + \pi$$

$$\mu_t > T_{Nmax} - T_{Xmin} + 2\pi + g$$

Remember that T_{Nmax} includes T_{Xmax}. We know that precision is related to the clock reading error, dictated by the variance in network delivery delay, $\pi = T_{Xmax} - T_{Xmin} + k$, where k is a constant dependent on the synchronisation protocol [15], which includes T_{Wmax} in time-triggered systems :

$$\mu_t > \pi - k + 2\pi + g$$

So, a working formula for clock-driven protocols in adverse environments where hidden channels are all network-based, is:

$$\boxed{\mu_t > 3\pi + g - k}$$

c)In some systems, π may be too large to render the optimisation above usable. In these cases, the only solution lies in using clock synchronisation protocols overcoming the precision limitation, either with hardware support [8], or taking advantage of LAN properties [17]. In this case, we can easily have $\pi \ll T_{Xmin}$, yielding a significant improvement of the utility of the working formula of (b). This shows high-precision clock synchronisation as being advantageous in adverse environments with network-based hidden channels, confirming the views expressed in [8].

TIMER-DRIVEN PROTOCOLS	WORK. FORMULAS
adverse environments	$\mu_t > \max_u (T^u_{Dmax})$ T-driven good vs. C-driven if C-driven $(T_{Xmax}) \simeq$ T-driven (T_{Dmax})
all-network hidden-channels	$\mu_t > \max_u (T^u_{Dmax}) - T_{Xmin}$ T-driven good vs. C-driven if C-driven $(T_{Xmax}) \simeq$ T-driven (T_{Dmax})
favourable environments	$\mu_t > \sigma$ σ of C-driven normally better than σ of T-driven

Figure 5: Compared behaviour of timer-driven protocols

Clock-driven protocols in favourable environments

One may consider, however, that environments are normally better behaved. Let us recall that an environment is favourable to the ordering protocol, if no hidden communication channel is faster than the protocol itself.

From theorem 3 we will have in general $\mu_t > \sigma$. In consequence, from the steadiness derivations (cf. § B.1), a general working formula for clock-driven protocols in favourable environments is:

$$\boxed{\mu_t > \pi + g}$$

Timer-driven protocols

It is harder to define working formulas for timer-driven protocols, since their performance depends on a number of factors. However, from what was derived above, we can establish a comparison with clock-driven protocols, exemplified in the table of figure 5.

a) In adverse environments without restrictions, timer-driven protocols are as good (or as bad) as clock-driven ones: the correctness condition is the same (cf. theorem 2), and although a comparison is difficult, in similar architectures T_{Xmax} of clock-driven protocols should not be much different from T_{Dmax} of timer-driven ones. Given that T_{Xmax} is the dominant term in the message delivery time of clock-driven protocols, in the above-mentioned situation both types have comparable performance, with regard to causal delivery.

b) In adverse environments with only network-based hidden channels the situation is different in absolute terms. However, clock-driven protocols may again be no better than timer-driven ones, if again for similar architectures, the same condition of (a) holds, and T_{Xmin} is the same in both.

c) In favourable environments, there are no hidden channels to worry about: the allowed pace of computation, μ_t, depends on steadiness, not on maximum delivery time. Clock-driven protocols are extremely effective here, mainly if high-precision protocols are

used, since steadiness is practically given by precision. Timer-driven protocols have normally a coarse steadiness, which does not make them appropriate for "fast" applications.

In spite of these remarks, in a number of settings, for example, continuous processes, valves, slow-motion discrete processes, etc., both μ_s (concerning physical actuator/sensor feedback), and μ_t (concerning processing and actuation times), are large enough for the system to fall under the favourable environment assumption and furthermore, to allow low-precision clock-driven or even timer-driven operation, from the point of view of order. As an example, consider a system with a synchronous timer-driven protocol for LANs, with relatively loose synchronism, $T^i_{Dmax} = 50ms$, $T^i_{Dmin} = 10ms$, and hidden channels with $\mu_s = 12ms$. Then, $\sigma = 50 - 10 = 40ms$, and the environment is favourable, so, $\mu_t > \sigma = 40ms$. In consequence, such a setting would support control programs of granularity greater than $40ms$.

5 An evaluation of our approach

Let us analyse the situation under the classical model [10], to assess whether there is any real advantage in the notions introduced in this paper.

Lamport started by defining the "happened before relation", and proposed a solution based on logical order (with logical clocks).

HB [Lamport]:

- for $j > i$, $e^i_p \rightarrow e^j_p$;
- for $p \neq q$, $e^i_p \rightarrow e^j_q$ if e^i_p is the sending of a message and e^j_q is the receipt of that message;
- $e^i_p \rightarrow e^j_q$ if $e^i_p \rightarrow e$ and $e \rightarrow e^j_q$.

To solve the problem of anomalous behaviour, he introduced a "strong clock condition", whose purpose is to preserve precedence relations among events, even if created by events external to the ordering protocol. Then he proposed an implementation based on physical clocks, to guarantee the correctness of his message-based formulation of the HB relation. With the same purpose, we propose a formulation which generalises the HB relation (Definition 1), and then propose solutions based on that new relation, which are in consequence more general, as we will see below.

Lamport, for his implementation: defined correctness conditions specifically for clock-driven systems; was only concerned with hidden message passing channels, and made correctness depend only on the absolute minimum space-like separation. Recapitulating:

C1 [Lamport]: "Let μ be a number such that if event a occurs at physical time t and event b in another process satisfies $a \rightarrow b$, then b occurs later than physical time $t + \mu$. In other words, μ is less than the shortest transmission time for interprocess messages"

C2 [Lamport]: "Anomalous behaviour is avoided if $\epsilon \leq \mu$"

[In our terminology, this means: $\mu < \mu_s$, $\epsilon \equiv \pi$, $\mu_s > \mu \geq \epsilon$, that is, $\mu_s > \pi$. The original paper stated **C2** as $\epsilon \,/\, (1 - \kappa) \leq \mu$, where κ is the accuracy preservation of the clock set, that is, the rate at which they deviate from real time. We have ignored it, since it is a second order factor [15].]

So, while Lamport's definition of correctness depends on the absolute minimum space-like separation, μ_s, in our model we introduce the minimum time-like separation, i.e. the

minimum time it takes to process events, as the local granularity μ_t (Def. 5), together with μ_s (Def. 6).

Additionally, Lamport's definitions were limited by implementation-dependent parameters: he stipulated the minimum space-like separation, μ_s, to be greater than π, the precision of clocks. For the time-like separation, he just proposed that events local to a participant be separated by at least one clock tick. We are proposing a formulation, expressed by theorems 2 and 3, which is independent of protocol implementation— it may also be used with timer-driven protocols— and of environment— the ACT and OBS external events address network message passing as well as physical process feedback loops. On the other hand, it covers any speed of channels (Lamport's is only valid for $\mu_s > \pi$).

Finally, Lamport's results for clock-driven systems have to be revised with the introduction of clock granularity. As explained in [8, 5], clocks are granular, and it is not relevant to use a finer granularity for local clocks than the precision achieved by the global clock. This makes ordering properties dependent on clock granularity, besides precision. Our formulation is perfectly in line with this evolution [16].

We now compare our formulation with Lamport's, in detail.

Under Lamport's model, no solution would exist for adverse environments. In the limit situation, the expression of theorem 2 would become ($\mu_t = 0$): $\max_u (T^u_{Dmax}) < \mu_s$. Since, by hypothesis, $\mu_s < \min_u (T^u_{Dmin})$, we would have $\max_u (T^u_{Dmax}) < \mu_s < \min_u (T^u_{Dmin})$, which is clearly impossible, showing that there is not a solution for adverse environments under Lamport's model.

For favourable environments, the situation according to Lamport's model is, introducing granularity, $\mu_s > \pi + g$. This ensures that the reception time-stamp is higher than the transmission time-stamp. However, this is not enough to build a protocol. Lamport gives additional conditions that lead to correct operation, and which are precursors of Cristian's protocol [4]. To this effect, see the discussion of clock-driven protocols under favourable environments, in the last section.

6 Conclusions and related work

There has been some work related to the temporal ordering properties of systems. Earlier on, Lamport established the fundamental theory behind it [10]. Later, the impact of finite precision [4] and granularity [8] have been pointed out as disturbance factors of theoretical models based on fully-synchronous systems and perfect and continuous clocks. A model for real-time objects and a discussion on the impact of temporal uncertainties on total order protocols has been advanced in [7]. To the author's knowledge, the characterisation of protocol ability to perform causal delivery, with regard to the execution environment, had not been addressed in previous papers.

We started by recalling [22] that a protocol with a given degree of synchronism is capable of ordering messages separated by a given interval (δ_t-precedence). A recent work in this line [5] discusses how to match sparse time-bases to δ_t-precedent sets of events. In doing so, we have shown that there is no fundamental reason why synchronous timer-driven protocols should not be able to provide temporal order.

Then, we proposed a model for causal delivery that generalises that of Lamport [10]:

external events address physical process feedback loops as well as network message passing; *local granularity* represents the time it takes to process events; *propagation delay* represents the time it takes for information to be propagated between nodes, for any speed of channels.

Finally, we evaluated the capabilities of real protocols in several kinds of environments. One side-effect of our model, besides confirming the suitability of timer-driven protocols for a number of settings, has been to show unexpected limitations of clock-driven protocols, despite their general ability to provide fine-grained orderings.

We intend to experiment with these concepts, to explore new forms of communication in distributed real-time systems, in the scope of adaptive time-critical systems: systems reliably supporting mixed event- and time-triggered operation. As discussed in [9], there is a duality between these two styles of system, and its understanding may help pursuing the afore-mentioned objective.

Acknowledgements

The author wishes to thank André Schiper for his collaboration in rendering this material clearer and formally sounder. Vassos Hadzilacos and Sam Toueg made useful suggestions with regard to terminology. A warm acknowledgement also goes to Andreas Krueger, for the many detailed comments, and to the anonymous reviewers, whose contributions have improved the paper.

A Consolidated List of New Results

Definition 1: Precedence. An event precedes ($\overset{p}{\rightarrow}$) another event if one of the following conditions hold:

- for $j > i$, $e_p^i \overset{p}{\rightarrow} e_p^j$;
- for $p \neq q$, $e_p^i \overset{p}{\rightarrow} e_q^j$ if e_p^i is an external event $ACT_p(a)$ and e_q^j an external event $OBS_q(a)$;
- for $p \neq q$, $e_p^i \overset{p}{\rightarrow} e_q^j$ if there exists an event e such that $e_p^i \overset{p}{\rightarrow} e$ and $e \overset{p}{\rightarrow} e_q^j$.

Definition 3: δ_t-precedence ($\overset{\delta_t}{\rightarrow}$): An event e is said to δ_t-precede an event e', $e \overset{\delta_t}{\rightarrow} e'$, **if** $t(e') - t(e) > \delta_t$.

Definition 4: Steadiness of a protocol, σ, is the greatest difference, for all participants p, between T^p_{Dmax} and T^p_{Dmin}: $\sigma = \max_p (T^p_{Dmax} - T^p_{Dmin})$

Definition 5: Local granularity, μ_t, of a system, is the minimum delay between two consecutive events in any participant p, e_p^j and e_p^{i+1}: $\forall p, \forall i \quad t(e_p^{i+1}) - t(e_p^i) \geq \mu_t$

Definition 6: Propagation delay, μ_s, of a system, is the minimum delay between an $ACT_p(a)$ event and the corresponding $OBS_q(a)$ event: $\forall p, q, p \neq q \quad t(OBS_q(a)) - t(ACT_p(a)) \geq \mu_s$

Theorem 1: A protocol with σ steadiness delivers messages m_1 and m_2 such that $send(m_1) \overset{\sigma}{\rightarrow} send(m_2)$, in temporal order.

Theorem 2: Given a system with local granularity μ_t and propagation delay μ_s, a protocol such that $\mu_t > \max_u (T^u_{Dmax}) - \mu_s$, ensures causal delivery order.

Valid for any environment (adverse or favourable): $\mu_s < \min_p (T^p_{Dmin})$

Theorem 3: Given a system with local granularity μ_t and propagation delay μ_s, and a protocol with steadiness σ and an absolute minimum delivery time of $\min_u (T^u_{Dmin})$, such that $\mu_s = \min_u (T^u_{Dmin})$ and $\mu_t > \sigma$, that protocol ensures causal delivery order.

Valid for favourable environments: $\mu_s = \min_p (T^p_{Dmin})$

Application of the theorems to E-triggered C-driven protocols

E-T CLOCK-DRIVEN PROTOCOLS	WORK. FORMULAS
adverse environments ($\mu_s = 0$)	$\mu_t > \Delta + \pi$
adverse environments and all-network hidden channels ($\mu_s = T_{Xmin}$)	$\mu_t > 3\pi + g - k$
using very precise clocks	$\mu_t > 3\pi + g - k$ $\pi \ll T_{Xmin}$
favourable environments ($\mu_s = \min_u (T^u_{Dmin})$)	$\mu_t > \pi + g$

Application of the theorems to T-driven protocols

TIMER-DRIVEN PROTOCOLS	WORK. FORMULAS
adverse environments	$\mu_t > \max_u (T^u_{Dmax})$ T-driven good vs. C-driven if C-driven (T_{Xmax}) $\simeq$ T-driven (T_{Dmax})
all-network hidden-channels	$\mu_t > \max_u (T^u_{Dmax}) - T_{Xmin}$ T-driven good vs. C-driven if C-driven (T_{Xmax}) $\simeq$ T-driven (T_{Dmax})
favourable environments	$\mu_t > \sigma$ σ of C-driven normally better than σ of T-driven

B Informal proof of ancillary results

B.1 Derivation of protocol parameters

For each protocol, we start by deriving the absolute worst-case bounds on delivery time, T_{Dmin} and T_{Dmax}. Then we derive steadiness, which in some cases is not the difference of the latter, but smaller, due to common mode effects that occur on transmissions to a single recipient (used to measure σ). $C_u(t_i)$ denotes the value of the clock of site u, during the clock interval that starts at real time t_i. A clock interval lasts g, the clock granularity.

Event-triggered clock-driven protocols

We recall that Cristian's protocol delivers messages time-stamped $T(m)$ at $C_r = T(m) + \Delta$ on the recipient's clock, and that the constant Δ is $\Delta = T_{Xmax} + \pi + g$.

Intuitively, we should seek for delivery time extremes, in the cases where the worst-case separation of clocks (precision π) occurs.

Thus, $\min_u(T^u_{Dmin})$ must occur in transmissions from a site q to a site r where C_r leads C_q by π. Message m time-stamped $T(m)$, is delivered at the beginning of clock interval $C_r(t_{dlv}) = \Delta + T(m)$, at real time t_{dlv}. The latest that message can be sent by q with time-stamp $T(m)$, is just before the start of the clock interval $C_q(t_{snd}) = T(m) + 1$, at real time t_{snd}. So, this will yield the shortest delivery time:

$$\min_u(T^u_{Dmin}) = t_{dlv} - t_{snd}$$

If t_0 is the beginning of clock interval $C_q(t_0) = T(m)$, and t_1 is the beginning of clock interval $C_r(t_1) = T(m)$, then, given precision,

$$t_0 - t_1 = \pi$$

So,

$$t_{snd} - g - t_1 = \pi$$

$$t_{snd} = \pi + g + t_1$$

Delivery at r occurs Δ ticks later than the beginning of clock interval $C_r(t_1) = T(m)$:

$$t_{dlv} = t_1 + \Delta$$

Then,

$$\min_u(T^u_{Dmin}) = t_{dlv} - t_{snd} = t_1 + \Delta - \pi - g - t_1$$
$$\boxed{\min_u(T^u_{Dmin}) = \Delta - \pi - g}$$

□

As for $\max_v(T^v_{Dmax})$, it must occur in transmissions from a site p to a site q where C_p leads C_q by π. Message m time-stamped $T(m)$, is delivered at the beginning of clock interval $C_q(t_{dlv}) = \Delta + T(m)$, at real time t_{dlv}. The earliest that message can be sent by p with time-stamp $T(m)$, is just at the start of the clock interval $C_p(t_{snd}) = T(m)$, at real time t_{snd}. So this will yield the longest delivery time:

$$\max_v(T^v_{Dmax}) = t_{dlv} - t_{snd}$$

If t_1 is the beginning of clock interval $C_q(t_1) = T(m)$, then, given precision,

$$t_1 - t_{snd} = \pi$$

$$t_{snd} = t_1 - \pi$$

Delivery at q occurs Δ ticks later than the beginning of clock interval $C_q(t_1) = T(m)$:

$$t_{dlv} = t_1 + \Delta$$

Then,

$$\max_v(T^v_{Dmax}) = t_{dlv} - t_{snd} = t_1 + \Delta - t_1 + \pi$$
$$\boxed{\max_v(T^v_{Dmax}) = \Delta + \pi}$$

□

Next we derive steadiness. Since steadiness is measured as observed at one site, then it is impossible to simultaneously observe the two limit situations above, that is: transmission from a

site p to a site r where C_p leads C_r by π; transmission from a site q to a site r where C_r leads C_q by π. In fact, this would lead to a precision of 2π.

It is easy to see that whether we start the proof by the absolute maximum or by the absolute minimum delivery times, the conclusion will be the same: if the sending site clock must be π far apart from the recipient clock, the other extreme must occur from a site in phase with the recipient clock. We go back to our proof of the absolute minimum delivery time, and from then on, we derive the maximum delivery time for that same site, to achieve steadiness.

The absolute minimum delivery time must occur in transmissions from a site q to a site r where C_r leads C_q by π, as shown in the afore-mentioned proof. Consider again message m time-stamped $T(m)$, delivered at the beginning of clock interval $C_r(t_{dlv}) = \Delta + T(m)$, at real time t_{dlv}. We saw that the latest that message m can be sent by q with time-stamp $T(m)$, is just before the start of the clock interval $C_q(t_{snd}) = T(m) + 1$, at real time t_{snd}, yielding:

$$T^r_{Dmin} = \Delta - \pi - g$$

Conversely, given a message m' also delivered at r at the beginning of a given clock interval, the earliest that message can be sent from another site p to r, will yield the maximum delivery time at r. Given $C_r(t_{dlv'}) = \Delta + T(m')$, the earliest that message can be sent by p with time-stamp $T(m')$, is just at the start of the clock interval $C_p(t_{snd'}) = T(m')$, at real time $t_{snd'}$. The more p's clock leads r's clock, the earlier $t_{snd'}$ will occur, and the limit situation will yield the longest delivery time at r:

$$T^r_{Dmax} = t_{dlv'} - t_{snd'}$$

Given that by the hypothesis made, q's clock lags r's clock by π, r's clock is among the fastest clocks in the system, so p's clock cannot be faster than q's. In consequence, $C_r(t) = C_p(t)$, and thus, if t_1 is the beginning of clock interval $C_r(t_1) = T(m')$, then,

$$t_{snd'} = t_1$$

Delivery at r occurs Δ ticks later than the beginning of clock interval $C_r(t_1) = T(m')$:

$$t_{dlv'} = t_1 + \Delta$$

Then,

$$T^r_{Dmax} = t_{dlv'} - t_{snd'} = t_1 + \Delta - t_1$$

$$T^r_{Dmax} = \Delta$$

In consequence, steadiness of clock-driven protocols is

$$\sigma = \max_r\ (T^r_{Dmax} - T^r_{Dmin}) = \Delta - (\Delta - \pi - g)$$
$$\boxed{\sigma = \pi + g}$$

□

Time-triggered clock-driven protocols

We now derive the parameters of time-triggered, clock-driven protocols, such as the TTP time-triggered protocol [6]. We follow the model presented in section 4. A benefit of this model is that it shows that these protocols are indeed similar to the event-triggered, clock-driven ones. In

consequence, we are going to derive our informal proof of the parameters of a time-triggered, clock-driven protocol, by recapitulation of the proof of event-triggered, clock-driven protocols, just made, and induction for the present case.

Seeking for delivery time extremes, $\min_u(T^u_{Dmin})$ must occur in transmissions from a site q to a site r where C_r leads C_q by π. Message m time-stamped $T(m)$, is delivered at the beginning of clock interval $C_r(t_{dlv}) = \Delta + T(m)$, at real time t_{dlv}. A message from q with time-stamp $T(m)$, is sent (time-triggered) at the start of the clock interval $C_q(t_{snd}) = T(m)$, at real time t_{snd}. So, this will yield the shortest delivery time:

$$\min_u(T^u_{Dmin}) = t_{dlv} - t_{snd}$$

If t_1 is the beginning of clock interval $C_r(t_1) = T(m)$, then, given precision,

$$t_{snd} - t_1 = \pi$$

$$t_{snd} = \pi + t_1$$

Delivery at r occurs Δ ticks later than the beginning of clock interval $C_r(t_1) = T(m)$:

$$t_{dlv} = t_1 + \Delta$$

Then,

$$\min_u(T^u_{Dmin}) = t_{dlv} - t_{snd} = t_1 + \Delta - \pi - t_1$$
$$\boxed{\min_u(T^u_{Dmin}) = \Delta - \pi}$$

□

As for $\max_v(T^v_{Dmax})$, it must occur in transmissions from a site p to a site q where C_p leads C_q by π. Message m time-stamped $T(m)$, is delivered at the beginning of clock interval $C_q(t_{dlv}) = \Delta + T(m)$, at real time t_{dlv}. The earliest that message can be sent (time-triggered) by p with time-stamp $T(m)$, is obviously at the start of the clock interval $C_p(t_{snd}) = T(m)$, at real time t_{snd}. So this will yield the longest delivery time:

$$\max_v(T^v_{Dmax}) = t_{dlv} - t_{snd}$$

If t_1 is the beginning of clock interval $C_q(t_1) = T(m)$, then, given precision,

$$t_1 - t_{snd} = \pi$$

$$t_{snd} = t_1 - \pi$$

Delivery at q occurs Δ ticks later than the beginning of clock interval $C_q(t_1) = T(m)$:

$$t_{dlv} = t_1 + \Delta$$

Then,

$$\max_v(T^v_{Dmax}) = t_{dlv} - t_{snd} = t_1 + \Delta - t_1 + \pi$$
$$\boxed{\max_v(T^v_{Dmax}) = \Delta + \pi}$$

□

Next we derive steadiness. As for event-triggered clock-driven protocols, steadiness is measured as observed at one site, so the same impossibility holds. In consequence, we follow the same approach: use the absolute minimum delivery time, and from then on, derive the maximum delivery time for that same site, to achieve steadiness.

We saw that the absolute minimum delivery time is:

$$T^r_{Dmin} = \Delta - \pi$$

Conversely, given a message m' also delivered at r at the beginning of a given clock interval, the earliest that message can be sent from another site p to r, will yield the maximum delivery time at r. Given $C_r(t_{dlv'}) = \Delta + T(m')$, the earliest that message can be sent by p with time-stamp $T(m')$ is, by the hypothesis about time-triggered behaviour, at the start of the clock interval $C_p(t_{snd'}) = T(m')$, at real time $t_{snd'}$. The more p's clock leads r's clock, the earlier $t_{snd'}$ will occur, and the limit situation will yield the longest delivery time at r:

$$T^r_{Dmax} = t_{dlv'} - t_{snd'}$$

We saw in the event-triggered proof that, in order to follow the π precision hypothesis, $C_r(t) = C_p(t)$, and thus, if t_1 is the beginning of clock interval $C_r(t_1) = T(m')$, then,

$$t_{snd'} = t_1$$

Delivery at r occurs Δ ticks later than the beginning of clock interval $C_r(t_1) = T(m')$:

$$t_{dlv'} = t_1 + \Delta$$

Then,

$$T^r_{Dmax} = t_{dlv'} - t_{snd'} = t_1 + \Delta - t_1$$

$$T^r_{Dmax} = \Delta$$

In consequence, steadiness of time-triggered clock-driven protocols is

$$\sigma = \max_r\,(T^r_{Dmax} - T^r_{Dmin}) = \Delta - (\Delta - \pi)$$
$$\boxed{\sigma = \pi}$$

□

Timer-driven protocols

Timer-driven protocols are harder to analyse, because they are normally more complex than the protocols just analysed. An analytical performance study of the AMp, done in the DELTA-4 project [18], showed a few interesting results.

Maximum and minimum bounds are denoted by the subscripts $_{min}$ and $_{max}$. For example, $T_{XWRmin}(m)$ accounts for the minimum duration of a transmit-with-reply round. $T_{Xmin}(\langle message \rangle)$ is the minimum transmission time of a given message. $T_{WaitDecision}$ is the duration of the timer started by recipients waiting for a decision of a just-arrived message. The study presented the following results:

- the smallest delivery time, corresponding to the no-failure, minimum network load case,

$$T^i_{Dmin} = T_{XWRmin}(m) + T_{Xmin}(Decision)$$

- the largest delivery time, corresponding to the worst-case network load and failure scenario (i.e. the sender fails),

$$T^i_{Dmax} = T_{XWRmax}(m) + (k+1).T_{WaitDecision} + T_{XWRmax}(Step1) + T_{XWRmax}(Step2)$$

The best case corresponds to the completion of the transmit-with-reply round in the minimum time (transmission of the data message followed by the reply), followed by the transmission of the decision message to terminate the protocol. The worst case corresponds to the failure of the sender, in the worst network load conditions. That is, maximum duration of the transmit-with-reply of the data message, followed by $k+1$ expirations of the $T_{WaitDecision}$ timer (since the sender is dead), and subsequent run of the group membership protocol (again under worst-case network load), in two steps, finally having the message delivered at all sites.

The absolute performance figures of this kind of protocols are affected by a number of factors, which vary from setting to setting. However, the relative numbers do not vary much from setting to setting. In fact, we determined that 1:11 is a suitable bound for the ratio between the AMp minimum and maximum delivery times, which yields $\sigma = 10.T^i_{Dmin}$. □

B.2 Difference between steadiness and temporal uncertainty

Let us recall the difference between steadiness (σ) as we define it, and temporal uncertainty (Tu), as defined by Kopetz:

$$\sigma = \max_p \, (T^p_{Dmax} - T^p_{Dmin})$$
$$Tu = \max_{p,q} \, (T^p_{Dmax} - T^q_{Dmin})$$

To substantiate the remark we made earlier, we claim that in theorems 1 and 3, the tightest result is obtained with σ, and not with Tu. The informal proof is as follows:

(i) Suppose there exists a protocol such that $Tu > \sigma$. In consequence, if we replace σ by Tu in theorem 1, though still correct, it would be less tight. As for theorem 2, the same substitution would yield a $\mu_t > Tu$ condition, also coarser.

(ii) We now prove that such a protocol where $Tu > \sigma$ exists. From section B.1, an event-triggered clock-driven protocol has

$$\max_p(T_{Dmax}) = \Delta + \pi$$
$$\min_q(T_{Dmin}) = \Delta - \pi - g$$

In consequence,

$$Tu = \max_{p,q} \, (T^p_{Dmax} - T^q_{Dmin}) = 2\,\pi + g$$

whereas, from the same section,

$$\sigma = \pi + g$$

so,

$$Tu > \sigma$$

□

References

[1] Yair Amir, Danny Dolev, Shlomo Kramer, and Dalia Malki. Transis: A Communication Sub-System for High-Availability. In *Digest of Papers, The 22nd International Symposium on Fault-Tolerant Computing Systems*, pages 76–84. IEEE, 1992.

[2] K. Birman and T. Joseph. Reliable Communication in the Presence of Failures. *ACM, Transactions on Computer Systems*, 5(1), February 1987.

[3] Kenneth Birman, Andre Schiper, and Pat Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3), August 1991.

[4] F. Cristian, Aghili. H., R. Strong, and D. Dolev. Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement. In *Digest of Papers, The 15th International Symposium on Fault-Tolerant Computing*, Ann Arbor-USA, June 1985. IEEE.

[5] H. Kopetz. Sparse Time versus Dense Time in Distributed Systems. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, Yokohama, Tokyo, June 1992. IEEE.

[6] H. Kopetz and G. Grunsteidl. TTP - a Time-Triggered Protocol for Fault-Tolerant Real-Time Systems. In *Digest of Papers, The 23th International Symposium on Fault-Tolerant Computing*, pages 524–533, Toulouse, France, June 1993. IEEE.

[7] Hermann Kopetz and K.H.(Kane) Kim. Temporal Uncertainties in Interactions among Real-time Objects. In *Proceedings of the Ninth Symposium on Reliable Distributed Systems*, pages 165–174, Huntsville, Alabama, October 1990. IEEE.

[8] Hermann Kopetz and Wilhelm Ochsenreiter. Clock Syncronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, C-36(8):933–940, August 1987.

[9] Hermann Kopetz and Paulo Veríssimo. Real-time and Dependability Concepts. In S.J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, pages 411–446. Addison-Wesley, 1993.

[10] Leslie Lamport. Time, Clocks and the Ordering of Events in a Distributed System. *CACM*, 7(21), July 1978.

[11] P.M. Melliar-Smith and L.E. Moser. Fault-Tolerant Distributed Systems Based on Broadcast Communication. In *Proceedings of the 9th Internacional Conference on Distributed Computing systems*, pages 129–133. IEEE, June 1989.

[12] Larry L. Peterson, Nick C. Buchholz, and Richard D. Schlichting. Preserving and Using Context Information in Interprocess Communication. *ACM Transactions on Computer Systems*, 7(3), August 1989.

[13] L. Rodrigues and P. Veríssimo. xAMp: a Multi-primitive Group Communications Service. In *Proceedings of the 11th Symposium on Reliable Distributed Systems*, pages 112–121, Houston, Texas, October 1992. INESC AR/66-92.

[14] A. Schiper, J. Eggli, and A. Sandoz. A New Algorithm to Implement Causal Ordering. In *Proceedings of the 3rd Int Workshop on Distributed Algorithms*, volume LNCS 392, pages 219–232, Nice - France, September 1989. Springer Verlag.

[15] Fred B. Schneider. Understanding Protocols for Byzantine Clock Synchronization. Technical report, Cornell University, Ithaca, New York, August 1987.

[16] P. Veríssimo. Ordering and Timeliness Requirements of Dependable Real-Time Programs. *Journal of Real-Time Systems, Kluwer Eds.*, 7(2):105–128, September 1994. Also as INESC AR/14-94.

[17] P. Veríssimo and L. Rodrigues. A posteriori Agreement for Fault-tolerant Clock Synchronization on Broadcast Networks. In *Digest of Papers, The 22th International Symposium on Fault-Tolerant Computing*, Boston - USA, July 1992. INESC AR/65-92.

[18] P. Veríssimo, J. Rufino, H. Fonseca, and L. Rodrigues. The performance of the xAMp protocol on token-bus and fddi nac's. Technical Report RT/109-91, INESC, Lisboa, Portugal, November 1991.

[19] Paulo Veríssimo. Real-time Data Management with Clock-less Reliable Broadcast Protocols. In *Proceedings of the Workshop on the Management of Replicated Data*, Houston, Texas-USA, November 1990. IEEE. also as INESC AR/25-90.

[20] Paulo Veríssimo. Real-time Communication. In S.J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, pages 447–490. Addison-Wesley, 1993.

[21] Paulo Veríssimo, P. Barrett, P. Bond, A. Hilborne, L. Rodrigues, and D. Seaton. The Extra Performance Architecture (XPA). In D. Powell, editor, *Delta-4 - A Generic Architecture for Dependable Distributed Computing*, ESPRIT Research Reports, pages 211–266. Springer Verlag, November 1991.

[22] Paulo Veríssimo, L. Rodrigues, and J. Rufino. The Atomic Multicast protocol (AMp). In D. Powell, editor, *Delta-4 - A Generic Architecture for Dependable Distributed Computing*, ESPRIT Research Reports, pages 267–294. Springer Verlag, November 1991.