Proceedings Intl. Conf. on Dependable Systems and Networks (DSN'01), p. 57–66, Göteborg, Sweden, July 2001

How to Model Link Failures: A Perception-based Fault Model

ULRICH SCHMID
Technische Universität Wien
Department of Automation
Treitlstraße 1, A-1040 Vienna
Email: s@auto.tuwien.ac.at
Phone: ++43-1-58801-18325, FAX: ++43-1-58801-18391

Abstract

We propose a new hybrid fault model for clock synchronization and single-round (approximate) agreement in synchronous distributed systems[1]*, which accurately captures both node and link faults. Unlike conventional "global" fault models, which rest upon the total number of faulty nodes in the system, it solely relies upon the number of faults in any two non-faulty nodes' "perceptions" —conveyed by the messages from all other nodes— of the system. This way, arbitrary node and communication faults, including receiver-caused omission and time/value faults, can be modeled properly. As an example*[2]*, we show that the consistent broadcast primitive —and hence the clock synchronization algorithms— of Srikanth & Toueg can be analyzed under this model. As far as link faults are concerned, our analysis reveals that as few as* $4f_{\ell a} + 2f_{\ell s} + 2f_{\ell o} + 1$ *nodes are sufficient for tolerating at most* $f_{\ell a}$*,* $f_{\ell s}$*, and* $f_{\ell o}$ *asymmetric, symmetric, and omission link faults at any receiving node.*

Keywords: *Fault-tolerant distributed systems, fault models, link failures, consistent broadcasting, clock synchronization.*

1 Motivation

One of the most important abstractions facilitating a coherent view of the huge literature on distributed algorithms —and hence books like Lynch's [8]— is the "deterministic" fault model: To be able to reason deterministically about the execution of a distributed algorithm, it is usually assumed that at most f of the n nodes in the system may be faulty at any time. Apart from being simple and easy to handle, this model also captures both node and sender-caused link faults adequately.

Still, a link consists of the sender's network interface, the channel, and finally the receiver's network interface. As the latter unites messages from all other nodes, it is particularly susceptible to faults: Omission failures in modern wireline and, most notably, wireless networks are typically caused by receiver overruns (run out of buffers), unrecognized packets (synchronization errors), and CRC errors (data reception problems). Since any arriving message —even several consecutive ones— could be affected, the resulting receive omissions cannot be attributed to the "innocent" sending nodes.

Another example are receiver-caused timing failures in applications that disseminate time-dependent information, as in clock synchronization [20, 17]. Since the received information varies with the experienced end-to-end transmission delay here, excessive delays can easily produce incorrect —even inconsistent, i.e., Byzantine— information at the receiving end(s). The major sources of excessive delays, however, are network controller receive-FIFOs and device driver in-bound queues, see e.g. [16]. Again, blaming the sending nodes for those receiver-caused link faults is inappropriate.

Such link faults[3] must obviously be ascribed to the receiving nodes, and considered *independent of each other* at different receivers. Like node faults, they should be classified according to a hybrid fault model to ensure maximum fault-tolerance under realistic operating conditions. Unfortunately, however, there is no obvious way to accommodate link faults in a global fault model for synchronous[4] systems at all.

The major contribution of this paper is an alternative *perception-based* hybrid fault model for clock synchronization and single-round agreement algorithms, which provides all the required features. It has been stimulated by the fault model utilized for our analysis of interval-based clock syn-

[1]This research has been conducted in our SynUTC-project (see *http://www.auto.tuwien.ac.at/Projects/SynUTC/*) supported by the Austrian START programme Y41-MAT.

[2]Lacking space did not allow us to include the analysis of convergence-function-based algorithms in this paper; consult [13] for a more comprehensive treatment.

[3]Since sender-caused link faults can be counted as node faults, we will use the term *link fault* exclusively for receiver-caused ones.

[4]Note that it is relatively easy to handle link faults in asynchronous systems satisfying the "fair loss" property: If sending an infinite number of messages over a link causes an infinite number of messages to be received, a perfect link can be simulated by suitable retransmission schemes, see e.g. [1, 26]. Clearly, this approach cannot be used in synchronous systems.

chronization [14, 18]. Following an informal introduction in Section 2 and its formalization in Section 3, we show in Section 4 that a properly adapted implementation of the consistent broadcast primitive —and hence the clock synchronization algorithm— of Srikanth & Toueg [21] can be analyzed under this model. Some conclusions in Section 5 eventually round off the paper.

2 Informal Overview

We consider synchronous distributed systems consisting of n nodes, which are interconnected by a fully connected point-to-point or broadcast-type network. The basic assumptions are the following:

(a) Each receiving node can reliably identify the sender of a message,

(b) Flooding or jamming "foreign" links / the broadcast network is impossible.

Nodes and links may fail according to various fault models, ranging from a simple fail-stop model up to a Byzantine one, where faulty nodes/links can take arbitrary steps and transmit (and "receive") any number of arbitrary messages.

Conventional fault models, like the one that at most f nodes may behave Byzantine at any time, usually rest upon the total number of faults in the entire system. The usual assumption of entire nodes as fault containment regions, however, makes it difficult to accommodate link faults in such models: In [5], for example, link faults are simply mapped to sender node faults. If we grant each receiving node i, $1 \le i \le n$, just a single independent receive omission, it may e.g. be the case that each node $i > 2$ (resp. node 1) drops the message from sender $i-1$ (resp. sender n), see Figure 1. Hence, all n nodes must be considered faulty in this particular execution.

One might of course argue that the scenario portrayed in Figure 1 is a quite unlikely one. Still, receiver-caused link errors often affect multiple messages, which means that even less "exotic" fault patterns could lead to the above situation. In any case, we cannot rule out the possibility of executions where any reasonable $f < n$ is exceeded in a conventional global fault model. This observation suggests that approximate agreement algorithms cannot be used in presence of link faults.

However, for single round agreement algorithms, there is in fact no need for relying upon the total number faults in the system: Showing (approximate) agreement of all non-faulty nodes amounts to showing, in a pairwise manner, that any two non-faulty nodes achieve (almost) the same result. For example, the usual clock synchronization condition just requires that any two non-faulty nodes' clocks satisfy $|C_p(t) - C_q(t)| \le \pi$, *without regard to the other nodes in the system.* Hence, only non-faulty receiving nodes p

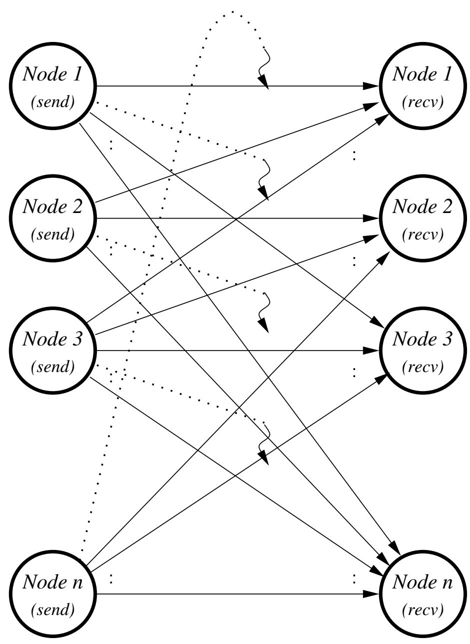

Figure 1. *System-wide broadcast with $f_\ell = 1$ receive omissions per receiver. Any node must be considered faulty in a conventional fault model here.*

and q need to be considered, and their sets of received messages can be considered in isolation. Global fault assumptions —like the one that <u>all</u> receivers perceive a certain fault consistently— are indeed not required here.

It is hence quite natural to move to a *perception-based fault model* as introduced in our analysis of interval-based clock synchronization [17, 14, 12, 18], where the number of faults in any two non-faulty receivers' *perception vectors* $\mathcal{V}_p = (V_p^1, \dots, V_p^n)$ resp. $\mathcal{V}_q = (V_q^1, \dots, V_q^n)$ —containing p's resp. q's local perceptions V_p^s resp. V_q^s of node s, $1 \le s \le n$, as conveyed by its messages— is considered. Apart from disposing of the system-wide number of faults, this approach also facilitates refined fault containment regions, namely, basic nodes (computation and emission), links, and node receiving units, since the respective faults are easily mapped to faulty perceptions. As shown in Figure 2, a node fault of sender s usually affects the corresponding perception V_r^s at all receiving nodes r in a correlated fashion, although probably inconsistently in case of a Byzantine fault. On the other hand, a link fault hitting the link from node s to p affects V_p^s in p's perception vector only, see Figure 3.

Consequently, recalling the above example of $f_\ell = 1$ receive omissions per node, it is obvious that any two local views can differ only in at most $2f_\ell = 2$ perceptions, namely, the ones where either receiving node experienced its omission. Moreover, only at most $f_\ell = 1$ of the non-faulty perceptions present at some non-faulty node can be missing at any other non-faulty node.

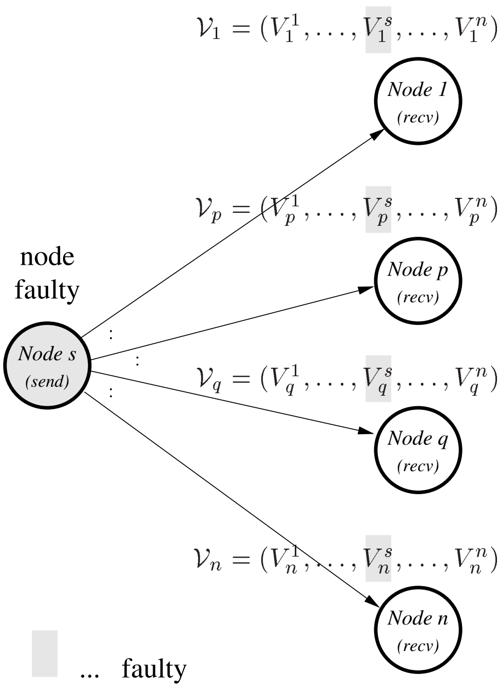

Figure 2. *Effect of a node fault of sender s upon the perception vectors of all receivers r. All perceptions V_r^s are usually faulty here, although probably inconsistently affected in case of an arbitrary fault.*

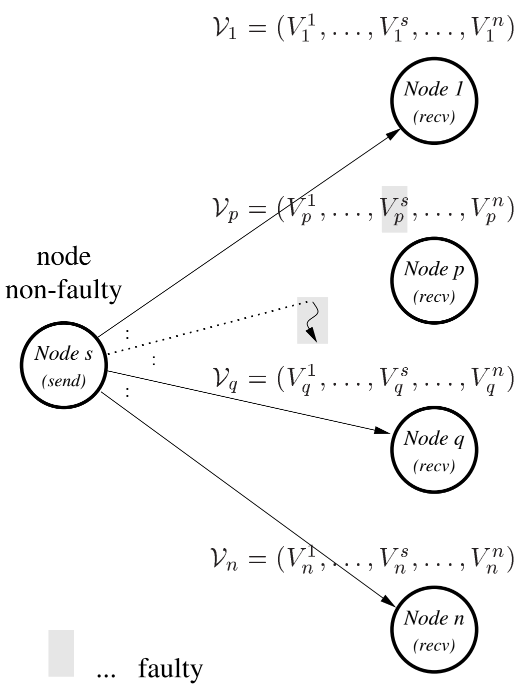

Figure 3. *Effect of a link fault between sender s and receiver p upon the perception vectors of all receivers r. Only V_p^s is faulty here.*

In sharp contrast to the pessimistic perspective derived from the global fault model, this suggests that approximate agreement algorithms should work well in presence of link faults. And indeed, the result of Rushby's formal verification of the Interactive Convergence Algorithm (ICA) [11] shows that as few as $n \geq 2f_\ell + 1$ nodes are sufficient for ICA to cope with the scenario of Figure 1. Note that, apart from our work on interval-based clock synchronization [17, 14, 12, 18], this paper seems to be the first and only one where something like a perception-based fault model has been employed before. Still, it considers only locally detectable ("manifest") link faults, which do not cover more severe incidents like timing faults. Our paper incorporates arbitrary link faults and shows that algorithms other than ICA can be analyzed in this framework.

Some recent papers [13, 19] reveal that other distributed algorithms also have fault-tolerance capabilities that are better than traditional analysis suggests. It is important to note, though, that improving the fault-tolerance w.r.t. node faults is impossible: After all, a global fault assumption like "at most f nodes may appear Byzantine" also implies "at most f pairs of perceptions may appear Byzantine", for any pair of non-faulty receiving nodes. A perception-based fault model thus "preserves" the corresponding global one, which means that classic impossibility results like [4, 3] remain valid.

3 Formal Framework

In order to formalize our perception-based fault model, we assume that all nodes s, $1 \leq s \leq n$, are somehow provided with a (virtual) *event* $V^s[t^s]$ that occurs at some *event generation time* t^s specified in any suitable global time scale. Besides its pure occurrence, $V^s[t^s]$ may or may not carry additional information, given by its *value* V^s. Note that V^s is often used as an abbreviation for $V^s[t^s]$ when the event generation time t^s is clear from the context.

$V^s[t^s]$ is a virtual event in the sense that it does not usually correspond to a physical event at node s. Rather, it is just the hypothesized trigger of the local execution of the particular distributed algorithm at node s, which eventually informs all nodes q (including $q = s$) of the fact that $V^s[t^s]$ has occured. By allowing an arbitrary behavior of the algorithm's local execution, including false triggering at any time, we can always assume that $V^s[t^s]$ itself occurs correctly[5] (i.e., with correct value V^s and correct event generation time t^s) even when node s is faulty.

A few particular examples of virtual events[6] are

(1) $V^s[t^s] \models$ "node s should start round k",

[5]Note that the term "correctly" makes sense here due to the fact that event generation times at different nodes may be externally related, as in the local detection of a global predicate, for example.

[6]Of course, many distributed algorithms will involve several different virtual events, which must be modeled separately.

(2) $V^s[t^s] \models$ "node s should detect crash of node x",

(3) $V^s[t^s] \models$ "node s should propose *attack*".

The respective event values are $V^s = k$ in (1), $V^s = x$ in (2), and $V^s =$ *attack* in (3); the event generation times are of vital importance (and close to each other) in clock synchronization (1) and other time-dependent applications (2), but (almost) irrelevant in classic agreement (3).

For our general framework, there is obviously no need to fully specify the particular distributed algorithm employed. We only assume that it disseminates all nodes' virtual events $V^s[t^s]$ throughout the system by means of messages, in a way that any correctly received message conveying V^s can be taken as representative. Note that this property is trivially fulfilled if only a single message is sent to any receiver.

Any non-faulty receiving node q can hence build a *perception vector*

$$\mathcal{V}_q(t) = (V_q^1[t_q^1], V_q^2[t_q^2], \ldots, V_q^n[t_q^n]) \qquad (1)$$

of *perceptions* $V_q^s[t_q^s]$, $1 \leq s \leq n$, collected by real-time t. A perception's value V_q^s is the result of reception and optional preprocessing of the message from node s that conveyed V^s to q (including $s = q$), and its *occurrence time* $t_q^s \leq t$ is the time when V_q^s is made available at node q. Note carefully that, unlike V_q^s, the explicit value of t_q^s is usually not available. If no message from s arrived by time t or if preprocessing has not yet completed, $V_q^s = \emptyset$ for the unique empty value $\emptyset$ and $t_q^s := \infty$.

Most distributed algorithms, including the clock synchronization algorithm of Section 4 and classic agreement algorithms, do not need advanced preprocessing. In this case, $V_q^s = V^s$ and $t_q^s = t^s + \delta_q^s$ with δ_q^s denoting the *end-to-end computational + transmission delay* between sender s and receiver q, provided there was no fault. Advanced preprocessing is used in convergence-function based clock synchronization to "simulate" $t_q^s = t^s$, by moving the dependency upon transmission delays etc. to V_q^s, see [13]. Our framework thus makes explicit the relation between t_q^s and t^s (resp. V_q^s and V^s in the latter case) via the end-to-end delay, which is particularly beneficial for clock synchronization applications. Note that this feature distinguishes our approach from the otherwise very similar one of [10], which stipulates a (somewhat magic) allowed range for t_q^s instead.

Our assumptions on the distributed algorithm ensure that a non-faulty node's perception vector is monotonic in time during any single execution, in the sense that the number $|\mathcal{V}_q(t)|$ of non-empty perceptions $V_q^s \neq \emptyset$ in $\mathcal{V}_q$ can only increase; we denote this by

$$|\mathcal{V}_q(t)| \leq |\mathcal{V}_q(t')| \qquad \text{for } t' \geq t. \qquad (2)$$

Even more, since only one message from node s is accepted for V_q^s, a perception vector is also "static" in the sense that all non-empty perceptions present in $\mathcal{V}_q(t)$ are also present in $\mathcal{V}_q(t')$, denoted by

$$\mathcal{V}_q(t) \subseteq \mathcal{V}_q(t') \qquad \text{for } t' \geq t. \qquad (3)$$

Let us now consider the perception vectors belonging to the same execution of any two non-faulty nodes p and $q \neq p$ at some arbitrary real-time t, that is,

$$\begin{aligned}\mathcal{V}_p(t) &= (V_p^1[t_p^1], V_p^2[t_p^2], \ldots, V_p^n[t_p^n]),\\ \mathcal{V}_q(t) &= (V_q^1[t_q^1], V_q^2[t_q^2], \ldots, V_q^n[t_q^n]).\end{aligned}$$

In order to characterize the effect of faults upon p's and q's execution up to time t, it is —at least for the algorithms considered here— sufficient to know the number $f(t)$ of faulty pairs of perceptions $\{V_p^s, V_q^s\} \in \{\mathcal{V}_p(t), \mathcal{V}_q(t)\}$. Recall that a link fault between s and p affects $V_p^s[t_p^s]$ in $\mathcal{V}_p(t)$ only, whereas a sender node fault at node s affects $V_r^s[t_r^s]$ in all $\mathcal{V}_r(t)$, $1 \leq r \leq n$, in a correlated —though probably inconsistent— fashion.

Consequently, our *perception-based fault model* just constrains $f(t)$ for any p, q in any admissible execution. Still, for finite t, the perception vectors may involve "phantom faults" ($t_p^s < t$ but $t_q^s > t$ for some node s) caused by the fact that $V_q^s[t_q^s]$ may not yet be available at node q by time t. A phantom fault may vanish/change type later on, at time t_q^s, when the result of an eventually received message is made available. Given the static nature of perception vectors, however, we can get rid of this problem by just considering $f(t)$ for $t = \infty$; we will denote this value by $f := f(\infty)$.

Apart from being perception-based, our fault model will also be a *hybrid* one, for both node and link faults. In a hybrid fault model [9, 22, 25, 2, 24], different classes of faults (manifest/crash/symmetric/asymmetric) are distinguished to account for the fact that less severe faults can be handled with fewer nodes than more severe ones. For example, masking f symmetric faults usually requires only $n \geq 2f+1$ nodes, whereas $n \geq 3f+1$ is needed if all faults are asymmetric ones. Since a large number of asymmetric faults is quite unlikely in practice, this effectively leads to a smaller n for tolerating a given number of faults. This, in turn, has a positive effect upon dependability by reducing the number of components that could be faulty, cf. [10].

We start our classification with the faults of a single perception V_p^s, which is more subtle than it meets the eye. It is straightforward for classic agreement algorithms, where value faults are the only source of problems: V_p^s is faulty iff $V_p^s \neq V^s$. In clock synchronization and related applications, however, both the value V_p^s and, in particular, the occurrence time t_p^s is essential. This means that $V_p^s[t_p^s]$ could be value and/or time faulty in this case. In addition, since the actual end-to-end computational + transmission delay δ_p^s that determines $t_p^s = t^s + \delta_p^s$ is unknown, a whole range of values must be considered correct here.

For any two non-faulty nodes s, q connected by a non-faulty link, let $\boldsymbol{\tau}_p^s$ be a set of admissible values of the end-to-end computational + transmission delay that ensures $\delta_p^s \in$

$\boldsymbol{\tau}_p^s$. Usually, $\boldsymbol{\tau}_p^s = [\tau_p^{s,-}, \tau_p^{s,+}]$ will just be an interval defined by some lower and upper bound on δ_p^s. In any case, $t_p^s \in t^s + \boldsymbol{\tau}_p^s$, where the operation $+$ is defined as

$$x + \boldsymbol{S} = \{v \mid v = x + \sigma \text{ for all } \sigma \in \boldsymbol{S}\}.$$

Therefore, $V_p^s[t_p^s]$ will be considered time faulty iff $t_p^s \notin t^s + \boldsymbol{\tau}_p^s$:

Definition 1 (Single Faults: Time) *Given a set of admissible delays $\boldsymbol{\tau}_p^s$, a node p's perception $V_p^s[t_p^s]$ of the event $V^s[t^s]$ disseminated by node s can suffer from the following mutually exclusive faults:*

- Omission fault*: $t_p^s = \infty$ [and $V_p^s := \emptyset$],*
- Time fault*: $t_p^s < \infty$ but $t_p^s \notin t^s + \boldsymbol{\tau}_p^s$.*

An analogous argument applies to the relation between V^s and V_q^s in algorithms like convergence-function-based clock synchronization [14, 12, 18] revisited in [13], where preprocessing translates the above timing range into a value range. Let $\boldsymbol{\nu}_p^s$ be a suitable set of admissible offsets ensuring $V_p^s \in V^s + \boldsymbol{\nu}_p^s$ for any two non-faulty nodes s, p connected by a non-faulty link. Note that we assume here that the values are suitably encoded such that the operation + is well-defined. In most cases, however, the only admissible (correct) value will be V^s and hence $\boldsymbol{\nu}_p^s = \{0\}$.

Definition 2 (Single Faults: Value) *Given a set of admissible value offsets $\boldsymbol{\nu}_p^s$, a node p's perception $V_p^s[t_p^s]$ of the event $V^s[t^s]$ disseminated by node s can suffer from the following mutually exclusive faults:*

- Omission fault*: $V_p^s = \emptyset$ [and $t_p^s := \infty$],*
- Value fault*: $V_p^s \neq \emptyset$ but $V_p^s \notin V^s + \boldsymbol{\nu}_p^s$.*

A single perception is called faulty iff it is time faulty, value faulty, or both.

Remarks:

1. Recall that event generation times t^s and values V^s (as well as the specifications $\boldsymbol{\tau}_p^s$ and $\boldsymbol{\nu}_p^s$) are always considered correct, even in case of a faulty node s. Only t_p^s and/or V_p^s may be faulty.
2. Time faults can be caused by node timing failures or excessive transmission delays.
3. Value faults can be produced by faulty nodes or apparently correct but in fact corrupted messages, and could also be the consequence of a time fault if advanced preprocessing is applied.
4. $V_p^s = \emptyset$ in $\mathcal{V}_p(t)$ for $t < \infty$ does not necessarily imply a real omission but could just mean that node s did not yet send its message.
5. Except for manifest faults like code errors [10], it is usually impossible to decide locally whether a perception is faulty or not.

For our perception-based fault model, the single-perception faults of Definition 1 and 2 must be extended to faults of pairs of perceptions $\{V_p^s, V_q^s\}$. As before, a pair of perceptions is considered faulty if it is either time faulty, value faulty, or both. For our definitions, we need the operation $-$ defined by

$$x - \boldsymbol{S} = \{v \mid v = x - \sigma \text{ for all } \sigma \in \boldsymbol{S}\},$$

which boils down to $t_p^s - \boldsymbol{\tau}_p^s = [t_p^s - \tau_p^{s,+}, t_p^s - \tau_p^{s,-}]$ if $\boldsymbol{\tau}_p^s = [\tau_p^{s,-}, \tau_p^{s,+}]$ is just an interval. The following Definition 3 specifies all possible classes of time faults.

Definition 3 (Pairwise Faults: Time) *A pair of perceptions $\{V_p^s[t_p^s], V_q^s[t_q^s]\}$ of the event $V^s[t^s]$ disseminated by node s can suffer from the following mutually exclusive time faults:*

- Symmetric omission fault *($\mathcal{SO}$): $t_p^s = t_q^s = \infty$.*
- Symmetric commission fault *($\mathcal{SC}_t$): $t_p^s < \infty$ and/or $t_q^s < \infty$ are/is time faulty and*
$$(t_q^s - \boldsymbol{\tau}_q^s) \cap (t_p^s - \boldsymbol{\tau}_p^s) \neq \emptyset. \tag{4}$$
- Asymmetric omission fault *($\mathcal{AO}$): W.l.o.g. $t_p^s = \infty$ and $t_q^s < \infty$ is not time faulty.*
- Asymmetric commission fault *($\mathcal{AC}_t$): W.l.o.g. $t_p^s < \infty$ is time faulty, t_q^s is arbitrary, and*
$$(t_q^s - \boldsymbol{\tau}_q^s) \cap (t_p^s - \boldsymbol{\tau}_p^s) = \emptyset. \tag{5}$$

Similarly, Definition 4 specifies all possible value faults:

Definition 4 (Pairwise Faults: Value) *A pair of perceptions $\{V_p^s[t_p^s], V_q^s[t_q^s]\}$ of the event $V^s[t^s]$ disseminated by node s can suffer from the following mutually exclusive value faults:*

- Symmetric omission fault *($\mathcal{SO}$): $V_p^s = V_q^s = \emptyset$.*
- Symmetric commission fault *($\mathcal{SC}_V$): $V_p^s = V_q^s$ is value faulty.*
- Asymmetric omission fault *($\mathcal{AO}$): W.l.o.g. $V_p^s = \emptyset$ and $V_q^s[t_q^s] \neq \emptyset$ is non-faulty.*
- Asymmetric commission fault *($\mathcal{AC}_V$): W.l.o.g. V_p^s is value faulty and $V_p^s \neq V_q^s$.*

Remarks:

1. The "classic" asymmetric fault [24] is one that is perceived inconsistently at p and q. This could occur, for example, when the transmission delay to p resp. q is excessively slow resp. fast or if the sending node exhibits Byzantine behavior. In our particular context, an asymmetric commission fault occurs iff[7] the sets $\boldsymbol{I}_p^s = t_p^s - \boldsymbol{\tau}_p^s$ and $\boldsymbol{I}_q^s = t_q^s - \boldsymbol{\tau}_q^s$, which both should contain t^s, have an empty intersection $\boldsymbol{I}_p^s \cap \boldsymbol{I}_q^s = \emptyset$. Hence, at least one node could not deduce the correct value of t^s from its local information in this case. Note that one of the two perceptions may be arbitrary (empty, correct or faulty) here.

[7] But see Remark 1 on Theorem 1.

2. For the consistent broadcast algorithm analyzed in Section 4, all nodes and links are assumed to be the same and hence $\forall s,p : \tau_p^{s,-} = \tau^-$ and $\tau_p^{s,+} = \tau^+$, with the resulting maximum uncertainty $\varepsilon = \tau^+ - \tau^-$. It is easy to see that (4) resp. (5) boils down to $|t_q^s - t_p^s| \le \varepsilon$ resp. $|t_q^s - t_p^s| > \varepsilon$ in this case.
3. The "classic" symmetric fault [24] is caused by faulty information that is perceived similarly at p and q. This type of fault is usually caused by a sender (clock) that runs too slow or too fast, but may also occur if one node experiences a slightly excessive delay in an otherwise correct broadcast.
4. An asymmetric omission is visible both in the time and in the value domain, since it is charactercterized by $t_p^s = \infty$ and $V_p^s = \emptyset$ simultaneously. Consequently, we defined only one class $\mathcal{AO}$ (and $\mathcal{SO}$) for both Definition 3 and 4.
5. A symmetric omission fault is caused by a sender node crash or a receive omission at both p and q. Note carefully, though, that it is impossible for either node to decide locally —without further information e.g. from on-line diagnosis [23]— whether its omission belongs to the class $\mathcal{SO}$, $\mathcal{AO}$ or even $\mathcal{AC}_t$/$\mathcal{AC}_V$.

In order to pin down the essential characteristics of our different classes of faults, the specifications given in Definition 3 and 4 are mutually exclusive. In practice, however, it is more common to define failure modes that are partially ordered, see e.g. [10]. After all, an algorithm that tolerates an asymmetric commision usually tolerates (even easier) just a symmetric commission as well. Figure 4 shows the partial order of time resp. value faults implied by algorithms like the one of Section 4. The "fork/join" structure is a consequence of the fact that a symmetric commission is both more severe than an asymmetric omission since it involves a time/value-faulty perception, and less severe since it is perceived consistently at both p and q, cp. Remark 2 on Theorem 1.

When considering time and value faults simultaneously, the cartesian product of the individual partial orders has to be constructed. Due to the fact that omissions are perceived identically in both domains, recall Remark 4 above, the resulting partial order has exactly the same structure as Figure 4. More specifically, recalling the classes of faults specified in Definition 3 and 4, our final failure modes are defined as follows:

Definition 5 (Pairwise Faults) *A pair of perceptions* $\{V_p^s[t_p^s], V_q^s[t_q^s]\}$ *of the event* $V^s[t^s]$ *disseminated by node* s *has the following failure modes:*

- Crash fault: $\mathcal{C} = \mathcal{SO}$
- Symmetric fault: $\mathcal{S} = \mathcal{C} \cup (\mathcal{SC}_t \cap \mathcal{N}_V) \cup (\mathcal{SC}_t \cap \mathcal{SC}_V) \cup (\mathcal{N}_t \cap \mathcal{SC}_V)$
- Omission fault: $\mathcal{O} = \mathcal{C} \cup \mathcal{AO}$
- Asymmetric fault: $\mathcal{A} = \mathcal{O} \cup \mathcal{S} \cup \mathcal{AC}_t \cup \mathcal{AC}_V$

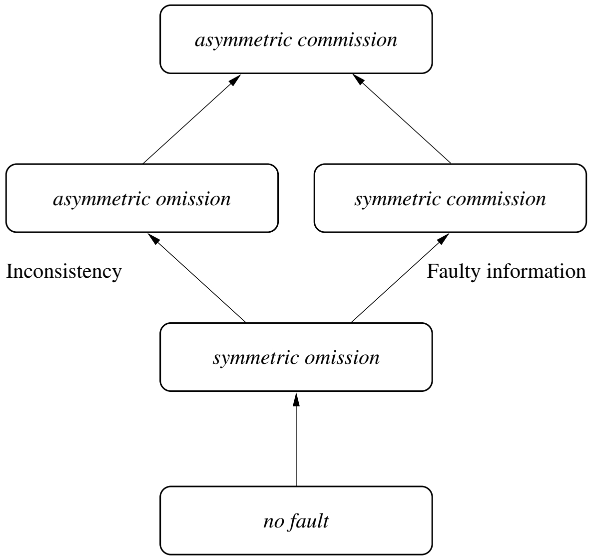

Figure 4. *Partial order of pure time and pure value faults of a pair of perceptions.*

Accordingly, a fault must be considered as asymmetric in case of an asymmetric time or value commission, but could just be any weaker type of fault as well. Note that we could have employed a notation like "asymmetric commission (or weaker)" as used in [14, 18] instead of overloading the standard term "asymmetric fault". In fact, our definition of symmetric and asymmetric fault seems to even contradict the classic definition of those terms in some cases. Nevertheless, we think that overloading is justified by the fact that our extensions preserve the essentials of their original meaning: The meaning of symmetric / asymmetric fault is basically received consistently / inconsistently at different nodes. In our context, however, we had to relax the definition of "received identically" since we cannot assume identical information at different nodes, even in the faultless case.

4 Consistent Broadcasting

In this section, we will prove that (a variant of) the non-authenticated consistent broadcast primitive —and hence the clock synchronization algorithms— of [21] can be employed in distributed systems under a perception-based hybrid fault model. Consistent broadcasting is implemented by means of two functions, *broadcast* and *deliver*, which can be used to disseminate a message (*msg*) throughout the system, in a way that guarantees properties P1–P3 defined in Theorem 1 below.

The pseudo-code description of the consistent broadcast primitive is shown in Figure 5. It relies upon unreliably sending two kinds of messages to all nodes (including itself) in the system: *init_msg* is used by a node to signal that

To broadcast msg*, each correct node:*
send *init_msg* to all;

To deliver msg*, each correct node:*
if received *init_msg* from $f_{\ell a} + f_a + f_s + f_{\ell s} + U + 1$ distinct nodes
  $\rightarrow$ send *echo_msg* to all [once]; /* sufficient evidence */
$\Box$ received *echo_msg* from $f_{\ell a} + f_a + f_s + f_{\ell s} + 1$ distinct nodes
  $\rightarrow$ send *echo_msg* to all [once]; /* sufficient evidence */
fi
if received *echo_msg* from $3f_{\ell a} + 2f_a + f_o + f_{\ell o} + f_s + f_{\ell s} + 1$ distinct nodes
  $\rightarrow$ deliver *msg*; /* accept */
fi

Figure 5. *Consistent broadcast primitive for the hybrid fault model of Definition 6*

broadcast *msg* has been called locally, *echo_msg* is emitted (at most once) by any node that received at least one non-faulty *init_msg* or *echo_msg*. The numbers f_s, $f_{\ell s}$, f_o, $f_{\ell o}$, f_a, and $f_{\ell a}$ give the maximum tolerated number of faults as specified in Definition 6 below, and U is the unforgeability threshold given in Theorem 1.

For our system model, we assume that n (already booted[8]) nodes are interconnected by a fully connected point-to-point network, with end-to-end computational + transmission delay $\delta_p^s \in [\tau^-, \tau^+]$ if the path from sender s to receiver p (including $s = p$) is non-faulty. Let $\varepsilon = \tau^+ - \tau^-$ be the resulting uncertainty and $t_{del} = 2\tau^+$ the worst-case end-to-end round trip delay.

In order to justify the fault model given in Definition 6 below, we first have to map the event-based notion of Section 3 to to the algorithm of Figure 5. We need two different events and the appropriate perceptions for this purpose:

- $I^s[t^s] \models$ "node s entered broadcast *msg*",
- $I_p^s[t_p^s] \models$ "node p received *init_msg* from s".
- $E^s[t^s] \models$ "node s achieved sufficient evidence",
- $E_p^s[t_p^s] \models$ "node p received *echo_msg* from s".

There is only a single correct value $I^s = I_p^s = \textit{init\_msg}$ and $E^s = E_p^s = \textit{echo\_msg}$, such that the set of admissible offsets in Definition 2 is $\boldsymbol{\nu}_p^s \equiv \{0\}$ for both events. The admissible delays $\boldsymbol{\tau}_p^s = [\tau_p^{s,-}, \tau_p^{s,+}]$ in Definition 1 are defined by $\forall s, p : \tau_p^{s,-} = \tau^-$ and $\tau_p^{s,+} = \tau^+$; note that those bounds must also be valid for $p = s$. Let $\mathcal{V}_p(t)$ denote either $\mathcal{I}_p(t)$ or $\mathcal{E}_p(t)$.

Definition 6 (Hybrid Fault Model) *Consider the pairs of perception vectors $\{\mathcal{I}_p(\infty), \mathcal{I}_q(\infty)\}$ and $\{\mathcal{E}_p(\infty), \mathcal{E}_q(\infty)\}$ for any admissible execution of any two non-faulty nodes p and q. Among the n pairs of perceptions*

$$\{V_p^s[t_p^s], V_q^s[t_q^s]\} \in \{\mathcal{V}_p(\infty), \mathcal{V}_q(\infty)\}, \quad 1 \leq s \leq n,$$

there may be at most

- *f_c crash faults,*
- *$f_s + 2f_{\ell s}$ symmetric faults, $2f_{\ell s}$ of those caused by at most $f_{\ell s}$ link faults (slightly excessive delays) ascribed to p resp. q,*
- *$f_o + 2f_{\ell o}$ omission faults, $2f_{\ell o}$ of those caused by at most $f_{\ell o}$ link faults (receive omissions) ascribed to p resp. q,*
- *$f_a + 2f_{\ell a}$ asymmetric faults, $2f_{\ell a}$ of those caused by at most $f_{\ell a}$ link faults (excessive delays) ascribed to p resp. q,*

where the numbers are such that

$$n \geq 4f_{\ell a} + 3f_a + 2(f_o + f_{\ell o} + f_s + f_{\ell s}) + f_c + 1. \quad (6)$$

Remarks:

1. We do not consider the class of systemwide consistently perceived manifest faults [24] explicitly, since they are simple to accommodate in our context: To tolerate f_b manifest faults, $n \geq f_b + 1$ is sufficient.

2. A receive omission that "hits" an otherwise symmetric faulty (early) pair of perceptions produces an asymmetric fault, since an omission's effect is equivalent to that of a late time fault. However, two faults (one omission + one symmetric fault) are turned into a single asymmetric one here, which implies that sufficiently many non-faulty nodes for tolerating this situation are always available.

In order not to exceed the maximum allowed number of faults in the perception vectors, at most $f_a + f_o + f_s + f_c$ sending nodes —although not necessarily the same for different pairs $\{\mathcal{V}_p(\infty), \mathcal{V}_q(\infty)\}$— may behave faulty during the whole execution, and only $f_a + f_s$ of those may emit time- or value-faulty information. In fact, due to additional link faults, a single perception vector $\mathcal{V}_p(\infty)$ may contain at most

- $f_a + f_{\ell a} + f_s + f_{\ell s}$ time and/or value-faulty $V_p^s[t_p^s]$,
- $f_a + f_{\ell a} + f_o + f_{\ell o} + f_s + f_{\ell s} + f_c$ omissions.

Moreover, it is not difficult to show the following consequence of Definition 6:

Lemma 1 (Difference in Perceptions) *For a pair of perception vectors $\{\mathcal{V}_p(\infty), \mathcal{V}_q(\infty)\}$ that is in accordance with the fault model of Definition 6, at most $\Delta f = f_a + 2f_{\ell a} + f_o + f_{\ell o}$ of the perceptions $V_p^s[t_p^s] \neq \emptyset$ in $\mathcal{V}_p(t)$ may be missing in $\mathcal{V}_q(t + \Delta t)$ for any $\Delta t \geq \varepsilon$.*

[8] Applicability of consistent broadcasting for initial clock synchronization purposes is hence somewhat restricted!

Proof: By Definition 5, at most $f_{\ell a}$ perceptions may have been available too early at p without being avaliable yet at q, $f_a + f_{\ell a}$ perceptions may be late at q, and $f_o + f_{\ell o}$ ones could have been omitted at q. All symmetric faulty perceptions present in $\mathcal{V}_p(t)$, however, must also be present in $\mathcal{V}_q(t + \Delta t)$, recall Remark 2 following Definition 4. $\Box$

The following major Theorem 1 proves that the algorithm of Figure 5 satisfies the properties of consistent broadcasting:

Theorem 1 (Properties Consistent Broadcast) *The consistent broadcast primitive of Figure 5 with $0 \leq U \leq f_o + f_{\ell o}$ satisfies*

P1. Correctness*: If at least $n - f_a - 2f_{\ell a} - f_o - f_{\ell o} + U$ not necessarily correct nodes broadcast* msg *by time t, then every non-faulty node delivers* msg *by time $t + t_{del}$.*

P2. Unforgeability*: If at most U non-time/value-faulty nodes broadcast* msg *by time t, then no non-faulty node delivers* msg *by t or earlier.*

P3. Relay*: If a non-faulty node delivers* msg *at time t, then every non-faulty node does so by time $t + t_\Delta$, where $t_\Delta = \tau^- + 2\varepsilon \leq t_{del}$.*

Proof: (*Correctness.*) Since at least $2(f_a + f_{\ell a} + f_s + f_{\ell s}) + f_o + f_{\ell o} + U + f_c + 1$ not necessarily correct (including crashed) nodes broadcast *init_msg* by t by assumption, the perception vector at any non-faulty node p satisfies

$$|\mathcal{I}_p(t + \tau^+)| \geq f_{\ell a} + f_a + f_s + f_{\ell s} + U + 1,$$

since at most $f = f_a + f_{\ell a} + f_o + f_{\ell o} + f_s + f_{\ell s} + f_c$ *init_msg* could be missing in $\mathcal{I}_p$ by time $t + \tau^+$ due to node and link failures; at most $f_a + f_{\ell a} + f_s + f_{\ell s}$ of those may just be late due to a time fault.

Hence, all non-faulty nodes achieve sufficient evidence in the first **if** of Figure 5 and send *echo_msg* to all nodes. This in turn implies

$$\begin{aligned} |\mathcal{E}_q(t + 2\tau^+)| &\geq n - f \\ &\geq 3f_{\ell a} + 2f_a + f_o + f_{\ell o} + f_s + f_{\ell s} + 1 \end{aligned}$$

for any non-faulty node q, which causes q to deliver by $t + 2\tau^+ = t + t_{del}$.

(*Unforgeability.*) Since at most U non-time/value-faulty nodes broadcast *msg*, at most $f_a + f_s + U$ nodes could have sent *init_msg* and hence $|\mathcal{I}_p(t)| \leq f_a + f_{\ell a} + f_s + f_{\ell s} + U$ for any non-faulty node p. Note that we assume here that symmetric and asymmetric link faults can produce time faulty messages "out of thin air". No non-faulty p can hence achieve sufficient evidence in the first **if** of Figure 5 and send *echo_msg*. Therefore, only at most $f_a + f_s$ (time/value-faulty) nodes might have sent *echo_msg*, which implies $|\mathcal{E}_p(t)| \leq f_a + f_{\ell a} + f_s + f_{\ell s}$. Consequently, no correct p can achieve sufficient evidence in $\Box$ of the first **if** of Figure 5 and hence satisfy the acceptance criterion.

(*Relay.*) Since some non-faulty node p delivers at time t, $|\mathcal{E}_p(t)| \geq 3f_{\ell a} + 2f_a + f_o + f_{\ell o} + f_s + f_{\ell s} + 1$ according to the acceptance criterion. Hence, the perception vector at any non-faulty node q must satisfy

$$|\mathcal{E}_q(t + \varepsilon)| \geq f_{\ell a} + f_a + f_s + f_{\ell s} + 1$$

according to Lemma 1. It follows that all non-faulty nodes achieve sufficient evidence in $\Box$ of the first **if** of Figure 5 and send *echo_msg* to all nodes. As in the proof of correctness, this implies

$$|\mathcal{E}_r(t + \varepsilon + \tau^+)| \geq 3f_{\ell a} + 2f_a + f_o + f_{\ell o} + f_s + f_{\ell s} + 1$$

for any non-faulty node r, which causes r to deliver by $t + \varepsilon + \tau^+ = t + t_\Delta$. $\Box$

Remarks:

1. From the proof of the relay property, it is apparent that Lemma 1 need not hold for arbitrary t, but only for the times when some non-faulty node accepts. This reveals that we could strengthen the definition of asymmetric faults if some information about t^s was incorporated in Definition 3, recall Remark 1 following Definition 4.
2. Note carefully that Lemma 1 and hence the proof of the relay property would be invalidated if symmetric faults were allowed to come out as omissions, i.e., if omission faults were considered weaker than symmetric faults. Similarly, if receive omissions were allowed to come out as symmetric faults, the unforgeability proof (sufficient evidence in $\Box$!) would be invalidated. Hence, symmetric faults must not be considered weaker than omission faults as well, which shows that the "fork/join"-type partial order of failure modes in Figure 4 makes sense.
3. Definition 6 and (6) reveal that link faults are less costly in terms of the fault-tolerance degree than node faults. For example, $n \geq 3f_a + 1$ nodes are required for tolerating f_a "real" asymmetric faults, but only $n \geq 4f_{\ell a} + 1$ nodes are needed to cope with $2f_{\ell a}$ faulty perceptions caused by excessive reception delays. The reason is that $2f_{\ell a}$ link faults can produce only at most $f_{\ell a}$ (and not $2f_{\ell a}$) time/value faulty perceptions at either node.
4. Our correctness property P1 is slightly more demanding than the corresponding one of [21, p.639]. Fortunately, it is not difficult to verify that the analysis of the clock synchronization algorithm in [21] actually needs the following: If all non-faulty nodes broadcast *msg* by t, then every non-faulty node delivers *msg* by $t + t_{del}$. This is implied by P1, however, since we have in fact n not necessarily correct nodes in the system, which is much more than required by our correctness condition.
5. The unforgeability property involves a threshold U, which can be used to control the algorithm's sensitivity w.r.t. omission faults. $U = 0$ just guarantees that

a non-faulty node achieves sufficient evidence after reception of at least one non-time/value-faulty message, $U = f_o$ enforces reception of a message from at least one non-faulty node, and $U = f_o + f_{\ell o}$ eventually ensures that at least one of the received messages is also seen by any other non-faulty node.

6. Our agreement property P3 is usually better than the original one of [21, p.639], since $t_\Delta < t_{del}$ in most applications.
7. For $U = 0$, it is possible to reduce the worst-case total number of messages sent by the algorithm of Figure 5 from $2n^2$ to n^2. This can be done by not sending *echo_msg* if *init_msg* has already been sent (and *vice versa*), and by using receive *init_msg* or *echo_msg* instead of receive *echo_msg* in both **if** statements. Actually, *init_msg* and *echo_msg* can be merged into a single message here.
8. By using a receive operation with programmable delay in the acceptance **if** of Figure 5, our consistent broadcast primitive could benefit from certain ε-reducing measures. More specifically, our hardware timestamping support described in detail in [16] basically allows to measure the time t_{sr} a message spends on the sending and receiving node/network interface, leaving only the pure transmission delay $\delta_{sr} \in [\hat{\tau}^-, \hat{\tau}^+]$ with uncertainty $\hat{\varepsilon} = \hat{\tau}^+ - \hat{\tau}^-$. By delaying the reception of a message by $\tau^+ - \hat{\tau}^+ - t_{sr}$ in our algorithm, an end-to-end computational + transmission delay $\delta_r^{s\,\prime} \in [\tau^+ - \hat{\varepsilon}, \tau^+]$ with uncertainty $\varepsilon' = \hat{\varepsilon}$ could be "emulated".

   Of course, those measures can only reduce the variability of the time difference between acceptance at some node and achievement of sufficient evidence at all non-faulty nodes. Hence, we can improve $t_\Delta = \tau^- + 2\varepsilon$ at most to $t'_\Delta = \tau^- + \varepsilon + \hat{\varepsilon}$. This improvement can be considerably, though: For 10 Mb/s Ethernet, our measurements [16] revealed $\tau^- \approx 300\ \mu$s and $\tau^+ \approx 30$ ms, hence $\varepsilon \approx 30$ ms without hardware timestamping. With the memory-based timestamping technique of our *Network Time Interface* M-Module, however, we obtained $\hat{\varepsilon} \approx 15\ \mu$s![9]

The consistent broadcast primitive of Figure 5 is the centrepiece of the clock synchronization algorithms of [21]. Figure 6 shows its basic version, which employs consistent broadcasting to disseminate (*round k*) messages. Broadcasting (*round k*) indicates that the node is ready to resynchronize and hence start the k-th resynchronization round, which is done every P seconds. Each round k employs a new "instance" of consistent broadcasting to ensure nearly simultaneous resynchronization of all non-faulty nodes; a suitably chosen offset α is used to ensure that a clock is never set backwards.

[9] Our novel MII-based timestamping method [15] will even push $\hat{\varepsilon}$ down to the sub-μs range.

```
cobegin
    if C^{k-1}(t) = kP /* ready to start C^k */
       → broadcast (round k);
    fi
    if delivered (round k)
       → C^k(t) := kP + α; /* resynchronize */
    fi
coend
```

Figure 6. *A non-authenticated clock synchronization algorithm for round k*

Since the original proof of correctness of this algorithm only depends upon properties P1–P3, all results of [21] remain valid under our perception-based hybrid fault model. We can, however, sharpen the achievable synchronization precision due to our improved agreement property P3. More specifically, we only have to replace $d_{\min} = t_{del}$ by $d_{\min} = t_\Delta$ in the proof of Lemma 1 in [21, p.629] and use this value in all subsequent formulas. For example, the achievable synchronization precision reads

$$D_{\max} \geq [(P - \alpha)(1 + \rho) + t_{del}]dr + d_{\min}(1 + \rho), \quad (7)$$

where ρ is the worst-case clock drift and $dr = \rho(2+\rho)/(1+\rho)$.

Finally, it is not too difficult to verify that applying our analysis to the *optimal* variant of the clock synchronization algorithm of [21] renders the appropriate results valid as well. By just modifying the initialization value of $C^k(t)$, a clock synchronization algorithm was obtained that provides optimal drift rate. As before, $d_{\min} = 2t_\Delta$ has to be used instead of $d_{\min} = 2t_{del}$ in the proof of Lemma 1 in [21, p.635] to preserve validity of the formulas.

5 Conclusions

We presented a novel hybrid fault model for clock synchronization and approximate agreement in distributed systems, where the omniscient system-wide perception of faults is replaced by the local perceptions of (any) two non-faulty participants in the system. It accommodates both node and (receiver-caused) link faults and distinguishes asymmetric, symmetric, omission, and crash faults. Our model is hence well-suited for modern wireline and, in particular, wireless networked systems. We analyzed an improved version of the consistent broadcast primitive underlying the clock synchronization algorithms of Srikanth & Toueg [21] under this fault model and showed that $n \geq 4f_{\ell a} + 3f_a + 2(f_s + f_{\ell s} + f_o + f_{\ell o}) + f_c + 1$ nodes are sufficient for tolerating f_a, f_s, f_o, f_c node faults and $f_{\ell a}$, $f_{\ell s}$, $f_{\ell o}$ link faults at any receiver.

There are to two primary directions of further research in this area: First of all, one should try to analyze other distributed algorithms under a (generalized) perception-based fault model. There are already some encouraging results, most notably an analysis of the *Hybrid Oral Message* algorithm for consensus [6] in [19] and the *Fault-Tolerant Interval* convergence function (emulating the *Fault-Tolerant Midpoint* algorithm of [7] as well) in [13]. Second, one could try to integrate our perception-based model with on-line diagnosis approaches like [23], which might prove quite effective to further increase the fault-tolerance capabilities of certain algorithms.

6 Acknowledgements

We are indebted to Neeraj Suri and Sam Toueg for their comments on earlier versions of this paper. The many suggestions of one of the anonymous reviewers of DSN'01, in particular, the ones related to [10], are also gratefully acknowledged.

References

[1] M. K. Aguilera, W. Chen, and S. Toueg. Failure detection and consensus in the crash-recovery model. *Distributed Computing*, 13(2):99–125, 2000.

[2] M. H. Azadmanesh and R. M. Kieckhafer. New hybrid fault models for asynchronous approximate agreement. *IEEE Transactions on Computers*, 45(4):439–449, 1996.

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

[4] M. Fischer, N. Lynch, and M. Merrit. Easy impossibility proofs for the distributed consensus problem. *Distributed Computing*, 1(1):26–39, 1986.

[5] L. Gong, P. Lincoln, and J. Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. In *Proceedings Dependable Computing for Critical Applications-5*, pages 139–157, Champaign, IL, Sept. 1995.

[6] P. Lincoln and J. Rushby. A formally verified algorithm for interactive consistency under a hybrid fault model. In *Proceedings Fault Tolerant Computing Symposium 23*, pages 402–411, Toulouse, France, June 1993.

[7] J. Lundelius-Welch and N. A. Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and Computation*, 77(1):1–36, 1988.

[8] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[9] F. J. Meyer and D. K. Pradhan. Consensus with dual failure modes. In *In Digest of Papers of the 17th International Symposium on Fault-Tolerant Computing*, pages 48–54, Pittsburgh, July 1987.

[10] D. Powell. Failure mode assumptions and assumption coverage. In *Proc. 22nd IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-22)*, pages 386–395, Boston, MA, USA, 1992. (Revised version available as LAAS-CNRS Research Report 91462, 1995).

[11] J. Rushby. A formally verified algorithm for clock sychronization under a hybrid fault model. In *Proceedings ACM Principles of Distributed Computing (PODC'94)*, pages 304–313, Los Angeles, CA, Aug. 1994.

[12] U. Schmid. Interval-based clock synchronization with optimal precision. Technical Report 183/1-78, Vienna University of Technology, Department of Automation, July 1997. (submitted to Information and Computation).

[13] U. Schmid. How to model link failures: A perception-based fault model. Technical Report 183/1-108, Vienna University of Technology, Department of Automation, Oct. 2000.

[14] U. Schmid. Orthogonal accuracy clock synchronization. *Chicago Journal of Theoretical Computer Science*, 2000(3):3–77, 2000.

[15] U. Schmid, M. Horauer, and N. Kerö. How to distribute GPS-time over COTS-based LANs. In *Proceedings of the 31th IEEE Precise Time and Time Interval Systems and Application Meeting (PTTI'99)*, pages 545–560, Dana Point, California, Dec. 1999.

[16] U. Schmid, J. Klasek, T. Mandl, H. Nachtnebel, G. R. Cadek, and N. Kerö. A Network Time Interface M-Module for distributing GPS-time over LANs. *J. Real-Time Systems*, 18(1):24–57, Jan. 2000.

[17] U. Schmid and K. Schossmaier. Interval-based clock synchronization. *J. Real-Time Systems*, 12(2):173–228, Mar. 1997.

[18] U. Schmid and K. Schossmaier. How to reconcile fault-tolerant interval intersection with the Lipschitz condition. *Distributed Computing*, 14(2), May 2001. (to appear).

[19] U. Schmid and B. Weiss. Consensus with oral/written messages: Link faults revisited. Technical Report 183/1-110, Department of Automation, TU Vienna, February 2001. (submitted to DISC'01).

[20] F. B. Schneider. A paradigm for reliable clock synchronization. In *Proceedings Advanced Seminar of Local Area Networks*, pages 85–104, Bandol, France, Apr. 1986.

[21] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, July 1987.

[22] P. M. Thambidurai and Y. K. Park. Interactive consistency with multiple failure modes. In *Proceedings 7th Reliable Distributed Systems Symposium*, Oct. 1988.

[23] C. J. Walter, P. Lincoln, and N. Suri. Formally verified on-line diagnosis. *IEEE Transactions on Software Engineering*, 23(11):684–721, Nov. 1997.

[24] C. J. Walter and N. Suri. The customizable fault/error model for dependable distributed systems. *Theoretical Computer Science*, 2000. (Special issue on Dependable Computing, to appear).

[25] C. J. Walter, N. Suri, and M. M. Hugue. Continual on-line diagnosis of hybrid faults. In *Proceedings DCCA-4*, Jan. 1994.

[26] L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. Technical Report TR2000-1828, Computer Science Department, Cornell Univesity, December 2000.