IMPOSSIBILITY RESULTS AND LOWER BOUNDS FOR CONSENSUS UNDER LINK FAILURES

ULRICH SCHMID , BETTINA WEISS*, AND IDIT KEIDAR†

Abstract. We provide a suite of impossibility results and lower bounds for the required number of processes and rounds for synchronous consensus under transient link failures. Our results show that consensus can be solved even in presence of $O(n^2)$ moving omission and/or arbitrary link failures per round, provided that both the number of affected outgoing and incoming links of every process is bounded. Providing a step further towards the weakest conditions under which consensus is solvable, our findings are applicable to a variety of dynamic phenomena such as transient communication failures and end-to-end delay variations. We also prove that our model surpasses alternative link failure modeling approaches in terms of assumption coverage.

Key words. Fault-tolerant distributed algorithms, consensus, transient link failures, impossibility results, lower bounds, assumption coverage analysis.

AMS subject classifications. 68Q25, 68Q85, 68W15, 68W40

1. Introduction. Most research on fault-tolerant distributed algorithms conducted in the past rests on process failure models. Every failure occurring in a system is attributed to either the sending or receiving process here, irrespectively of whether the actual error occurs at this process or rather on the intermediate communication path. Moreover, a process that commits a single failure is often "statically" considered faulty during the whole execution, even if its failure is transient.

Although such process failure models adequately capture many important scenarios, including crash failures where a faulty process just stops operating, and Byzantine failures where a faulty process can do anything, they are not particularly suitable for modeling more dynamic phenomena. In particular, given the steadily increasing dominance of communication over computation in modern distributed systems, in conjunction with the high reliability of modern processors and robust operating system designs, transient communication failures such as lost or non-recognized packets (synchronization errors), CRC errors (data corruption), and receiver overruns (packet buffer overflow) are increasingly dominating real-world failures. Another dynamic phenomenon that is encountered frequently in practice is unpredictable variations of the end-to-end delays in multi-hop networks such as the Internet, which are caused, for example, by temporary network congestion and intermediate router failures. Since excessive end-to-end delays appear as omissions in classic (semi-)synchronous systems and other time(out)-based approaches, for example, [3–5, 7, 39, 43, 51, 53], such timing variations can also be considered as transient link failures.

The distinguishing properties of such failures are (a) that they affect the path (termed *link* in the sequel) connecting two processes, rather than the endpoints (the processes), and (b) that they are *mobile* [58], as different links may fail at different times. Hence, the ability to communicate [in a timely manner] with other processes in the system cannot be statically attributed to a process here: If the link failure rate is sufficiently high, there will never be a non-empty set of processes that appear

*Ulrich Schmid and Bettina Weiss are with Technische Universität Wien, Embedded Computing Systems Group (E182/2), Treitlstraße 1-3, A-1040 Vienna, Austria. ({s, bw}@ecs.tuwien.ac.at). Our research has been supported by the Austrian START programme Y41-MAT and the FWF project P18264 (SPAWN).

†Idit Keidar is with the Technion Haifa, Department of Electrical Engineering, Technion City, Haifa, Israel. (idish@ee.technion.ac.il).

non-faulty to each other, i.e., never send [timing] faulty messages to each other. As a consequence, classic process failure models are inappropriate for such applications.

This paper focuses on impossibility results and lower bounds for synchronous deterministic consensus in the presence of such mobile link failures. In the consensus problem (also called "Byzantine agreement" [47]), processes have to agree on a common output value, despite failures, based on some input values distributed among the processes. Consensus is widely recognized as one of the most fundamental problems in fault-tolerant distributed computing. Synchronous consensus algorithms execute in a sequence of lock-step rounds $k = 1, 2, \ldots$, which consist of sending, receiving and processing round k messages exchanged among all the processes.

Unfortunately, there is a discouraging impossibility result for deterministic synchronous consensus in the presence of general link failures (see Theorem 1 in Section 3), which goes back to Gray's 1978 paper [33] on atomic commitment in distributed databases. This result was strengthened by Santoro and Widmayer [58], who introduced the *mobile omissive process failure model*: In each round, some process can be send omissive (or omissive for short), in the sense that it can experience any number of transmission failures on its outgoing links. Even a single mobile omissive process failure was shown to render consensus unsolvable [58].

Despite those negative results, however, there are synchronous consensus algorithms for restricted link failure patterns. For example, it has been known for a long time that consensus can be solved when sufficient connectivity is always maintained [20, 37, 46].

More recently, Schmid, Weiss, and Rushby introduced a hybrid failure model for synchronous systems [66], which—in addition to classic process failures—admits up to $O(n^2)$ moving link failures per round. The link failure patterns must be such, however, that no more than f_ℓ^s outgoing links and no more than f_ℓ^r incoming links are affected at any process per round. An analysis of the assumption coverage [62] in the presence of independent, identically distributed probabilistic link failures confirmed that this model is suitable even for substantial link failure rates. Most existing consensus algorithms [10, 32, 47, 48, 68] were shown to work essentially unchanged under this hybrid failure model [11, 14, 64, 65, 71], provided that the number of processes n in the system is increased by $C_r f_\ell^r + C_s f_\ell^s$ for some small integers C_r, C_s that depend on the particular algorithm.

Naturally, the different values of C_r and C_s obtained for different consensus algorithms raised the question of lower bounds, both on the number of failures that can be tolerated, and on the number of rounds required to solve consensus. In the present paper, we provide the results of our comprehensive theoretical study of this subject:

1. In Section 2, we provide a precise definition of our system model, which involves both moving omission and moving arbitrary link failures (but no process failures).
2. In Section 3, we use a refinement of the bivalency proof techniques introduced in [58] for proving a versatile generalization of Gray's result, which reveals the importance of unimpaired *bidirectional* communication for solving consensus.
3. Using this general result, as well as a new instance of an easy impossibility proof [28], we provide a complete suite of impossibility results and lower bounds for the required number of processes (Section 4) and rounds (Section 5).
4. In Section 6, we show that our lower bounds are tight and characterize the threshold that, when exceeded, turn a correct process exhibiting omission

resp. arbitrary link failures according to our model into a classic omission resp. Byzantine faulty process.

5. In Section 7, we survey alternative approaches for modeling link failures and analyze their assumption coverage in a simple probabilistic setting. It turns out that our model is the only one with a coverage that approaches 1 (rather than 0) for large n.

Some conclusions and directions of future work in Section 8 eventually complete our paper.

2. System Model. We consider a system of n distributed *processes*, each identified by a unique id $p \in \Pi = \{1, \ldots, n\}$. The processes are fully connected by a point-to-point network made up of unidirectional *links*. Every pair of processes p and $q \neq p$ is hence connected by a pair of links (p, q), from sender process p to receiver process q, and (q, p), from sender process q to receiver process p, which are considered independent of each other. To simplify our presentation, we also assume that there is a link (p, p) from every process $p \in \Pi$ to itself. Our system hence contains n^2 unidirectional links. Links may reorder messages, that is, are not assumed to be FIFO.

2.1. Computing model. For our computing model, we employ the standard lock-step round model as used in [58]. Every process p is modeled as a deterministic state machine—acting on some local state $state_p \in State_p$—that can send and receive messages from some (possibly infinite) alphabet $\mathcal{M}$. The initial state of process p is drawn from a set of initial states $Init_p \subseteq State_p$. All processes execute, in perfect synchrony, a sequence of atomic computing steps forming a sequence of lock-step rounds $k \in K = \{1, 2, \ldots\}$: In round k, process p performs the following steps:

1. Initializes its *received messages vector* Rm_p to $\forall q \in \Pi : rm_p[q] = \emptyset$, where $\emptyset$ represents "no message", and sends at most one message $msg_p^k = Msg_p(state_p, k)$ to every process $q \in \Pi$ (including itself); $Msg_p : State_p \times K \to \mathcal{M} \cup \{\emptyset\}$ denotes the *message sending function* of the algorithm executed by p.
2. Waits for some time while receiving messages into Rm_p. This time must be sufficiently long to allow delivery of (most of) the round k messages. We assume that no messages arrive after this waiting period is over; practically, if late messages arrive, they are discarded.
3. Performs a state transition from $state_p$ to $state'_p = Trans_p(state_p, rm_p, k)$, where $Trans_p : State_p \times Rm_p \times K \to State_p$ denotes the *state transition function* of the algorithm executed by p.

Note that the round number k can be viewed as global time in this model, and is typically part of $state_p$.

The distributed algorithm executed by the processes is hence specified by the pairs of message sending function and message transition function $\{(Msg_p, Trans_p) | \, p \in \Pi\}$.

The *configuration* C^k of the system at the end of round k is the vector of states $(state_1^k, \ldots, state_n^k)$ obtained at the end of round k (after the state transition); the initial configuration is $C^0 = (init_1, \ldots, init_n)$ with $\forall p \in \Pi : init_p \in Init_p$. The system-wide $n \times n$ *received messages matrix* $\mathcal{R}^k$ for round k is the column vector $(rm_1^k, \ldots, rm_n^k)^T$ of all processes' received messages vectors in round k, i.e.,

$$\mathcal{R}^k = \begin{pmatrix} rm_1^k[1] & rm_1^k[2] & \ldots & rm_1^k[n] \\ rm_2^k[1] & rm_2^k[2] & \ldots & rm_2^k[n] \\ \vdots & \vdots & \vdots & \vdots \\ rm_n^k[1] & rm_n^k[2] & \ldots & rm_n^k[n] \end{pmatrix}, \tag{2.1}$$

with entry $rm_p^k[q]$ denoting the message that process p received from process q via its incoming link in round k, or $\emptyset$ if no such messages was received.

A *run* (also termed *execution*) of the distributed algorithm is an infinite sequence $C^0, \mathcal{R}^1, C^1, \mathcal{R}^2, \ldots$ of configurations alternating with received messages matrices, starting from some initial configuration $C^0 \in (Init_1, \ldots, Init_n)$.

2.2. Failure model. We assume that all processes are correct[1] but links may fail transiently.

Consider the system-wide $n \times n$ *sent messages matrix* $\mathcal{S}^k$ for round k, which consists of n identical rows containing the message sent by every process in round k, i.e.,

$$\mathcal{S}^k = \begin{pmatrix} msg_1^k & msg_2^k & \ldots & msg_n^k \\ msg_1^k & msg_2^k & \ldots & msg_n^k \\ \vdots & \vdots & \vdots & \vdots \\ msg_1^k & msg_2^k & \ldots & msg_n^k \end{pmatrix}, \tag{2.2}$$

with msg_p^k denoting the message process p sends to all processes via its outgoing links in round k, or $\emptyset$ if no message is sent.

Clearly, in case of no link failures in round k, $\mathcal{R}^k = \mathcal{S}^k$ since every message sent by p via its outgoing link to q is received faithfully by q via its incoming link from p. A link failure hitting the directed link (p, q) results in $rm_q^k[p] \neq msg_p^k$, however, so $\mathcal{R}^k \neq \mathcal{S}^k$ in this case. As in [58], we distinguish the following types of link failures of (p, q) in a single round k:

Correct link: $rm_q^k[p] = msg_p^k$
Lossy link: $\emptyset = rm_q^k[p] \neq msg_p^k$
Erroneous link (corruption): $\emptyset \neq rm_q^k[p] \neq msg_p^k \neq \emptyset$
Erroneous link (spurious): $\emptyset \neq rm_q^k[p] \neq msg_p^k = \emptyset$

For some round k, a lossy link is called *omission faulty*, an erroneous link (corrupted or spurious) is termed *arbitrary faulty*. A link producing either type of failure is termed *faulty*.

Our link failure model, originally introduced in [65,66], is such that, system-wide, up to $c \cdot n^2$ links, for some $c < 1$, may be faulty in any round. We cannot allow *any* set of $c \cdot n^2$ links to be hit by link failures, however, but require some restriction on the allowed link failure patterns: Let $\mathcal{F}^k$ be the $n \times n$ *failure pattern matrix* with entries

$$f_q^k[p] = \begin{cases} ok & \text{if } rm_q^k[p] = msg_p^k, \\ om & \text{if } \emptyset = rm_q^k[p] \neq msg_p^k, \\ arb(e) & \text{otherwise, where } e \text{ encodes the type of the actual error} \end{cases}$$

which is just the difference of $\mathcal{R}^k$ and $\mathcal{S}^k$ interpreted as ok, om, or $arb(e)$ on a per entry basis, depending on the corresponding link behavior. The *feasible* pattern of system-wide link failures must be such that for every process p and every round k,

(A^r) p's row $(f_p[1], \ldots, f_p[n])$ in $\mathcal{F}^k$ contains at most f_ℓ^r entries $\neq ok$, with at most $f_\ell^{r,a} \leq f_\ell^r$ of those equal to $arb(.)$. Since row p corresponds to p acting as a receiver process, we say that p may perceive at most f_ℓ^r *receive link failures* (on its incoming links), with up to $f_\ell^{r,a}$ arbitrary ones among those.

[1]Except for Section 5, where we also allow process crashes.

(A^s) p's column $(f_1[p], \ldots, f_n[p])^T$ in $\mathcal{F}^k$ contains at most f_ℓ^s entries $\neq ok$, with at most $f_\ell^{s,a} \leq f_\ell^s$ of those equal to $arb(.)$. Since column p corresponds to p acting as a sender process, we say that p may experience at most f_ℓ^s *send link failures* (on its outgoing links), with up to $f_\ell^{s,a}$ arbitrary ones among those.

Note that *every* process in the system may experience up to f_ℓ^s send link failures ($f_\ell^{s,a}$ of them arbitrary), and up to f_ℓ^r receive link failures ($f_\ell^{r,a}$ of them arbitrary) in every round. In addition, the particular links actually hit by a link failure may be different in different rounds. Of course, they may also remain the same, which makes our link failure model for example applicable to not fully-connected networks as well, cf. [67].

In the above modeling, the primary failure instance is the link failure pattern in the matrix $\mathcal{F}^k$. It determines how many link failures could be experienced by every process, both as a sender (send link failures f_ℓ^s, $f_\ell^{s,a}$) and as a receiver (receive link failures f_ℓ^r, $f_\ell^{r,a}$). Clearly, assumptions (A^r) and (A^s) imply that at most nf_ℓ^s outgoing links and at most nf_ℓ^r incoming links may be hit by a link failure. Since every outgoing link is of course some receiver's incoming link, it follows that the maximum allowed number of link failures occurs when $f_\ell^s = f_\ell^r = f_\ell$.

In our subsequent analysis, however, our assumptions on send link failures, as captured by (A^s) and f_ℓ^s, $f_\ell^{s,a}$, will be independent of the assumptions made on receive link failures, as captured by (A^r) and f_ℓ^r, $f_\ell^{r,a}$. Doing this allows us to extend the range of applicability of our model. In particular, by restricting the assumption "*every* process may commit up to f_ℓ^s send link failures" to "at most f_ℓ^r processes in some fixed subset of the processes may commit up to f_ℓ^s send link failures", we can also model *restricted process failures*: For example, a restricted omission faulty process is perceived omission faulty only by at most f_ℓ^s receiver processes per round (rather than by all receivers, as allowed in case of a standard omission faulty process), see Section 6 for details.

Note that adding classic process failures to the picture would provide a "fallback" in cases where the link failure restrictions (A^s) and/or (A^r) are violated: As long as the numbers of link failures experienced by a process p do not exceed the thresholds f_ℓ^r, $f_\ell^{r,a}$, f_ℓ^s, and $f_\ell^{s,a}$, process p can be considered correct. Otherwise, p can just be considered faulty, in which case the link failure restrictions do of course not apply. The resulting hybrid perception-based failure model has been applied successfully in the analysis of several different consensus algorithms [11, 14, 64–66, 71, 72]. In order to focus on the intrinsic costs of link failures, we will not add standard process failures to the model of this paper, however.

2.3. The consensus problem. Binary consensus is the problem of computing a common binary output value from binary input values distributed among all processes. We assume that every process p has a read-only *input value* $x_p \in \{0, 1\}$ in $state_p$, which is supplied via the initial state $init_p \in Init_p$ to p's local instance of a synchronous deterministic distributed consensus algorithm. In addition, p has a write-once *output value* $y_p \in \{0, 1\}$ in $state_p$, initially undefined. Process p irreversibly computes ("decides upon") y_p according to the following requirements:

(C0) (*Termination*): Every process decides within a finite number of rounds (that may be different for different processes, and may depend on the particular execution).

(C1) (*Agreement*): Every two processes p and q compute the same output value $y_p = y_q$.

(C2) (*Weak Validity*) [29]: If all processes start with the same input value, then

every process p computes
- $y_p = 1$ if $\forall q : x_q = 1$ and no link failure has occurred in the entire execution,
- $y_p = 0$ if $\forall q : x_q = 0$.

Note that practical consensus algorithms usually guarantee the following stronger validity property:

(C2') (*Validity*): If all processes start with the same input value $\forall q : x_q = v$, then every process p computes $y_p = v$.

In particular, $y_p = 1$ in case of $\forall q : x_q = 1$ even when link failures have occurred. We will employ the weaker form of validity in our proofs most of the time, since impossibility of consensus under (C2) obviously implies impossibility of consensus under (C2') as well.

3. Basic Results. In this section, we will provide a generic analysis of consensus in our setting, which essentially follows the approach taken in [58]:[2] Using bivalence arguments, we will show that consensus is impossible if every process p can *withhold* its information from a non-empty subset $Q = Q(p)$ of processes for an arbitrary number of rounds. Withholding is a weaker property than "adjacency-preserving" used in [58], however, so our generic results are slightly stronger than the ones of [58] and hence need a different proof. In particular, our findings reveal the importance of unimpaired *bidirectional* communication between processes for solving consensus.

In order to motivate the need for restricting failure patterns and to set the stage for our more advanced proofs, we start with Gray's well-known result [33], in the formalization of [50, Thm. 5.1]. It is devoted to the coordinated attack problem, which is just consensus with weak validity (C2) as stated in Section 2.3. This result assumes, however, that there are no constraints on the link failure pattern matrices (except that only omission failures are allowed).

THEOREM 1 (Gray's Impossibility [50, Thm. 5.1]). *There is no deterministic algorithm that solves the coordinated attack problem in a synchronous two-process system with arbitrary lossy links.*

Proof. Suppose that the failure-free execution $\overline{E}$ of a two-process system with omission faulty links terminates at the end of round r when starting with initial values $[1, 1]$. By validity, the common decision value must be 1 in $\overline{E}$. Since decisions are irreversible, we can safely drop all the messages some algorithm might send in rounds $> r$ without changing the decision value. The resulting "truncated" execution E shown in Fig. 3.1 is obviously feasible.

By means of induction on the number of messages k sent in E, we show that the decision value 1 does not change even when we drop all messages in E: For $k = 0$, the claim holds trivially. For the induction step, assume that $k > 0$ messages are sent in E where both processes decide by some round r and no messages are sent after round r. Let m be the last message from, w.l.o.g., $p_1 \to p_2$ in E (cp. the dotted one in Fig. 3.1). If m is dropped, the resulting execution $\overline{E}'$ is indistinguishable from E for p_1, so p_1 and, by agreement, also p_2 must eventually decide upon 1. Clearly, in $\overline{E}'$, process p_2 could decide after round r and even send out additional messages in some round $r' > r$ — we can only guarantee that the decision value is the same. However, p_1 has already decided by round r and hence cannot make use of such "late" messages.

[2]Note that we could also have employed the *layering framework* [52] by Moses and Rajsbaum, which provides generic consensus impossibility and lower bound results in a model-independent way. Although similar in its general structure, it is based on quite different lower level "tools", for example, potence instead of valence, which are—contrary to [42]—not required in our relatively simple setting.

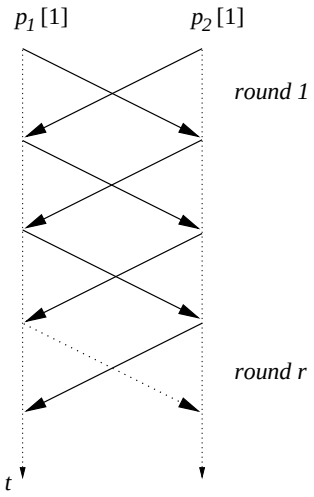

FIG. 3.1. Execution of a two-process synchronous consensus algorithm with unrestricted link omission failures, starting with initial values $[1, 1]$, truncated after round r by which both processes decide.

Consequently, we can safely drop all those late messages in $\overline{E}'$, if any, leading to an execution E' where only $k-1$ messages are sent, both processes decide upon 1 by some round r' and no messages are sent after round r'. We can hence apply the induction hypothesis to E', which completes the induction step.

Since the processes are fully isolated from each other in the resulting execution where all messages have been dropped, changing the initial values to $[1, 0]$ and then to $[0, 0]$ cannot affect the decision value either, but now the outcome of the final execution would violate validity. □

As our first "real" result, we will now show that solving consensus is even impossible when a link—viewed as a pair of unidirectional links—loses or, in case of arbitrary link failures, corrupts messages only in one direction, i.e., when either process (but not necessarily both) can withhold information for an arbitrary number of rounds from the other. Eventually bidirectional communication is hence mandatory for any deterministic consensus algorithm. Solutions exist, however, if the direction of the message loss is fixed, see the remarks following Theorem 2 below.

Unfortunately, this stronger result cannot be shown by generalizing the proof of Theorem 1: We are not allowed to simply drop all messages in later rounds to "hide" the effect of dropping/restoring round r-messages here, since this would amount to a link failure in both directions and hence an infeasible execution. It was shown in [58], however, that bivalency arguments [29] can successfully be applied in this setting.

We start with some notation: Recall that we defined an execution as an infinite sequence $C^0, \mathcal{R}^1, C^1, \mathcal{R}^2, \ldots$ of configurations alternating with received messages matrices, starting from some initial configuration $C^0 \in (\mathit{Init}_1, \ldots, \mathit{Init}_n)$. For a configuration $C^k = (c_1^k, \ldots, c_n^k)$, let $\mathit{state}_p(C^k) = c_p^k$ denote the local state of process p in C^k. Since we are dealing with deterministic algorithms, executions are uniquely determined by the initial configuration, i.e., the initial values x_p for all processes p, and the sequence of link failure pattern matrices $\mathcal{F}^k$ in rounds $k \geq 1$. Consequently, we can unambiguously specify executions as infinite sequences $C^0, \mathcal{F}^1, C^1, \mathcal{F}^2, \ldots$ of configurations alternating with failure pattern matrices. Moreover, a finite sequence of failure

pattern matrices $\mathcal{F}_x = \mathcal{F}^k, \ldots, \mathcal{F}^{k+x}$ starting from configuration C^{k-1} uniquely determines the finite *execution segment* $C^{k-1}, \mathcal{F}^k, C^k, \mathcal{F}^{k+1}, C^{k+1}, \ldots, \mathcal{F}^{k+x}, C^{k+x}$, and we write $C^{k+x} = \mathcal{F}_x(C^{k-1})$. An execution is called *feasible* if all $\mathcal{F}^k$ are feasible, i.e., adhere to the link failure pattern constraints (A^r) and (A^s) in Section 2.2. Finally, a configuration C' is called *reachable* from configuration C, if there is a feasible finite sequence of failure pattern matrices $\mathcal{F}_x$ such that $C' = \mathcal{F}_x(C)$. A configuration C is reachable if it is reachable from some initial configuration.

A configuration is called *v-decided* (*decided* for short) if all processes have decided on a common decision value $v \in \{0, 1\}$. A configuration C is *v-valent* (*univalent* if v is not known or irrelevant) if all decided configurations reachable from C are v-decided; in particular, it is impossible to reach a 1-decided configuration from a 0-valent C. On the other hand, C is *bivalent* if both 0-decided and 1-decided configurations can be reached from C.

For example, in case of $n = 2$, given any configuration $C^{k-1} = (c_1^{k-1}, c_2^{k-1})$, there are only four possible *successor configurations* C_{00}^k, C_{01}^k, C_{10}^k, and C_{11}^k in the case of message omissions: Configuration C^{k-1} is followed by the successor configuration C_{xy}^k, depending on whether the message $p_2 \to p_1$ (x) and/or $p_1 \to p_2$ (y) is lost $(x, y = 0)$ or correct $(x, y = 1)$ in round $k \geq 1$.[3] Note that C_{00}^k is feasible only in the setting of Theorem 1, where unrestricted losses are allowed, since both messages are lost there. In the context of the following Theorem 2, however, C_{00}^k cannot be reached from C^{k-1} since losing both messages is not feasible.

The situation is more complicated in case of arbitrary link failures, however, where C^{k-1} can have more than 4 successor configurations: After all, different errors e experienced, for example, by the message from $p_2 \to p_1$ due to an arbitrary link failure with $f_1^k[2] = arb(e)$ in the failure pattern matrix $\mathcal{F}^k$ might result in different states of p_1. Fortunately, we can keep the convenient assumption of just 4 successors if we replace a single successor state C_{xy}^k by the set of possible successor states. Using this extended interpretation, C_{01}^k actually consists of all configurations reachable from C^{k-1} where only the message $p_2 \to p_1$ is lost or erroneous, for example. Univalence of a set of configurations C_{xy}^k means that all individual configurations $C^k \in C_{xy}^k$ are univalent, whereas bivalence means that at least one individual configuration $C^k \in C_{xy}^k$ is bivalent. Bivalence proofs are easily generalized to this extended interpretation.

Finally, our notation can be easily generalized to $n > 2$: (Sets of) successor configurations are indexed by strings of $n(n-1)$ 0's or 1's, corresponding to every link in a system of n processes, with 0 denoting a lost or faulty message, and 1 denoting a non-faulty one.

Two successor configurations C_v^k and C_w^k are called *neighbors* if the received message matrices $\mathcal{R}_v^k$ and $\mathcal{R}_w^k$ that lead to C_v^k and C_w^k, respectively, differ in at most one entry: W.l.o.g., this entry contains the correct message in $\mathcal{R}_v^k$ but a lost/erroneous one in $\mathcal{R}_w^k$. Consequently, all configurations in C_{00}^k and C_{01}^k are neighbors in the above system, but the ones in C_{01}^k and C_{10}^k are not. The *successor graph* $\mathcal{G}_C$ of some configuration C consists of all successor configurations of C, where all neighbors are connected by an edge. We can make the following well-known observation:

LEMMA 1. *The successor graph $\mathcal{G}_C$ of any configuration C of a consensus algorithm under our system model is connected.*

Proof. Let $k \geq 1$ be the round at the end of which the transition from C to one of its successor configurations takes place. Obviously, the failure-free successor

[3]Message "self-transmission", from $p_1 \to p_1$ and $p_2 \to p_2$, is always assumed to be failure-free here. Since we are dealing with impossibility results and lower bounds, this can safely be assumed.

configuration $C_{1..1}$ where no round k messages has been lost or corrupted must be in $\mathcal{G}_C$. Let C_X be any other successor configuration caused by a feasible link failure pattern, with $\overline{\mathcal{M}}_X$ denoting the corresponding set of lost or faulty messages. Since the link failure pattern $\overline{\mathcal{M}}'_X$, obtained from $\overline{\mathcal{M}}_X$ by removing (= repairing) exactly one of the lost or faulty messages, is of course also feasible, the resulting successor configuration C'_X is a neighbor of C_X and obviously $C'_X \in \mathcal{G}_C$. Since $|\overline{\mathcal{M}}'_X| = |\overline{\mathcal{M}}_X| - 1$, this argument can be repeated until the failure-free successor configuration $C'_X = C_{1..1}$ is reached. Hence, there is a path from any C_X to $C_{1..1}$ in the successor graph $\mathcal{G}_C$. □

The result of Lemma 1 will be used primarily in conjunction with the following Lemma 2:

LEMMA 2. *Suppose that all successor configurations of some configuration C with successor graph $\mathcal{G}_C$ are univalent. If there are two arbitrary successor configurations C' and C'' among those that are 0-valent and 1-valent, respectively, then there are also two* neighboring *successor configurations $\overline{C}'$ and $\overline{C}''$ that are 0-valent and 1-valent.*

Proof. Since C' and C'' are connected in $\mathcal{G}_C$ and have different valences, there is a path of configurations connecting C' and C''. This implies that there must be neighbors $\overline{C}'$ and $\overline{C}''$ on this path where the valence changes. □

With these two lemmas, it is fairly easy to show that eventually bidirectional communication is mandatory for solving consensus in a 2-process system: The following Theorem 2 considers link omission failures only (that may change arbitrarily from round to round, however) and strengthens Gray's Theorem 1. Note that it could also be derived from the impossibility of consensus under a single moving process omission failure [58] in a system of $n = 2$ processes.

THEOREM 2 (Unidirectional 2-Process Impossibility). *There is no deterministic algorithm that solves consensus in a synchronous system with two non-faulty processes connected by a lossy link, even if communication is reliable in one direction in every round.*

Proof. Assume that there are algorithms $\mathcal{A}_{p_1}$ and $\mathcal{A}_{p_2}$ running on processes p_1 and p_2 that jointly solve consensus in a two-process system with unidirectional communication. We will show inductively that every bivalent configuration has at least one bivalent successor. This implies that it is impossible to always reach a final decision within any finite number of rounds.

For the base case $k = 0$ of our inductive construction, we have to show that there is a bivalent initial configuration. Consider the configuration $C^0(01)$ where p_1 starts with initial value 0 and p_2 starts with initial value 1. If $C^0(01)$ is bivalent, we are done. If $C^0(01)$ is 0-valent, the execution where all messages from $p_1 \to p_2$ are lost in all rounds must also lead to a 0-decided configuration. However, this execution is indistinguishable for p_2 from the equivalent execution that starts from $C^0(11)$ (where p_1 has initial value 1 instead of 0), which implies that the common decision value must also be 0 here. Since $C^0(11)$ must lead to a 1-decided configuration in case of no link failures by validity, we have shown that $C^0(11)$ is bivalent in this case. An analogous argument can be used to show that $C^0(00)$ would be bivalent when $C^0(01)$ is 1-valent.

For the induction step $k \geq 1$, we assume that we have already reached a bivalent configuration C^{k-1} at the end of round $k-1$ and show that at least one of the feasible successor configurations C^k_{01}, C^k_{10}, and C^k_{11} reached at the end of round k is bivalent. Assuming the contrary, all of those must be univalent. The bivalence of C^{k-1} implies

that at least one of C^k_{01}, C^k_{10}, and C^k_{11} must be 0-valent and one must be 1-valent (i.e., C^{k-1} must be a *fork* [69]). By Lemma 2 in conjunction with Lemma 1, either the neighbors C^k_{11} and C^k_{01}, or C^k_{11} and C^k_{10} must be v-valent and $(1-v)$-valent. W.l.o.g. assume that C^k_{11} is v-valent and C^k_{01} is $(1-v)$-valent for some $v \in \{0,1\}$. Since the only difference between those two configurations is that the message from $p_2 \to p_1$ arrives in the former but not in the latter, we only have to consider the execution C^{k*}_{11} where all messages from $p_1 \to p_2$ are lost in all rounds $> k$. However, as p_1 cannot tell p_2 whether it has received a round k message, the execution C^{k*}_{11} starting from C^k_{11} is indistinguishable for p_2 from the same execution starting from C^k_{01}. Since p_2 must eventually decide upon the same value, C^k_{11} and C^k_{01} cannot have different valences. □

Remarks:

1. If message losses can only occur in one direction, and if that direction is known to the algorithm, then there is a trivial 1-round algorithm that solves consensus in a 2-process system: The process that can communicate with its peer sends its own value and decides upon it; the other process decides upon the value received from its peer.
2. If message losses can only occur in one and the same but unknown direction, there is a simple 2-round algorithm that solves consensus in a 2-process system: Every p_i initially sets $v_i := x_i$. In the first round, p_1 sends v_1 to p_2. If p_2 receives v from p_1, it sets $v_2 := v$. In the second round, p_2 sends its value v_2 to p_1. If p_1 receives v from p_2, it sets $v_1 := v$. At the end of the second round, process p_i, $i = 1, 2$, decides upon v_i. It is easy to verify that the decision values satisfy validity and agreement.

Our next goal will be to show that consensus cannot be solved in any system of $n \geq 2$ processes if, for every process, eventually bidirectional communication with every peer cannot be guaranteed. More specifically, we will show that this is the case when every process p can *withhold* its information from some non-empty subset $\mathcal{Q} = \mathcal{Q}(p)$ of processes (but not necessarily vice versa) from any round $k+1$ on, namely, when there is a sequence of failure pattern matrices for rounds $k+1, k+2, \ldots$ such that every $q \in \mathcal{Q}$ has the same view of the resulting execution after round $k+x$, independently of the state of p in the starting configuration C^k.

Definition 1 (Withholding). *A process p, in some reachable configuration C^k, can* withhold its information from round $k+1$ on*, if there is an infinite sequence of failure pattern matrices $\mathcal{F} = \mathcal{F}^{k+1}, \mathcal{F}^{k+2}, \ldots$ and a non-empty set $\mathcal{Q}$ of processes with $p \notin \mathcal{Q}$ (where both $\mathcal{F}$ and $\mathcal{Q}$ may depend on p and C^k) such that, for any reachable configuration $\overline{C}^k$ with $\forall q \in \mathcal{Q} : state_q(C^k) = state_q(\overline{C}^k)$, it also holds that $\forall q \in \mathcal{Q} : state_q\big(\mathcal{F}_x(C^k)\big) = state_q\big(\mathcal{F}_x(\overline{C}^k)\big)$ for any finite prefix $\mathcal{F}_x = \mathcal{F}^{k+1}, \ldots, \mathcal{F}^{k+x}$ of $\mathcal{F}$.*

We say that p can withhold its information *if it can withhold its information from round $k+1$ on, for every $k \geq 0$, starting from any reachable configuration C^k.*

Definition 1 implies that if p can withhold its information, then, since $|\mathcal{Q}| \geq 1$, there is at least one process $q = q(p)$ which never gets any useful information from p at all during $\mathcal{F}$, neither directly nor indirectly via other processes.

It is important to note that withholding is a weaker property than *adjacency-preserving*, on which the generic results of [58] are based: The latter requires that *all* processes $\neq p$ have the same state in $\mathcal{F}_x(C^k)$ and $\mathcal{F}_x(\overline{C}^k)$, i.e., corresponds to p withholding its state from *every* $q \neq p$. Consequently, adjacency-preserving implies withholding, but not vice versa. In fact, the proofs of Theorems 3 and 6 will show that

withholding—but not adjacency-preservation—is possible under our failure model for certain parameter values. Basically, this is due to some "unidirectional" partitioning of the n processes in the system, which is possible when (A^s) and (A^r) hold, provided that n is small enough.

As a consequence, we cannot use the generic consensus impossibility results from [58]. The following Lemma 3 reveals, however, that the ability of every process p to withhold its information from some non-empty subset of the processes arbitrarily long already prohibits a solution to the consensus problem.

LEMMA 3 (n-Process Impossibility). *Consider a synchronous n-process system with omission and/or arbitrary link failures. There is no deterministic algorithm that solves consensus in such a system if every process p can withhold its information in any configuration.*

Proof. The proof is a generalization of the proof of Theorem 2, although more involved: We assume here that there are n algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_n$ running on the n processes $p_1, \ldots, p_n$ in the system that jointly solve consensus. We will show inductively that there is an infinite execution involving bivalent configurations only, which makes it impossible to always reach a decision within a finite number of rounds.

For the base case $k = 0$ of our inductive construction, we have to show that there is a bivalent initial configuration C^0. As in [29], we consider the initial configuration $C^0(111..1)$, where all processes start with the initial value 1. If this configuration is bivalent, we are done. Otherwise, $C^0(111..1)$ can only be 1-valent, since validity requires a decision value 1 in the failure-free case. Now consider $C^0(011..1)$, where process p_1 starts with 0 and all others with 1. If this configuration is bivalent, we are done. If not, we assume first that it is 0-valent and choose the execution starting from $C^0(011..1)$ where p_1 witholds its value from some process $q(p_1) = p_x \neq p_1$; such an execution must exist according to Definition 1 since p can withhold its information. This execution is indistinguishable for p_x from the analogous execution starting from $C^0(111..1)$, however, so p_x's (and hence the common) decision must be 1 here. This contradicts the stipulated 0-valence of $C^0(011..1)$, however, which could hence only be 1-valent.

The whole argument can now be repeated for p_2 in place of p_1, etc., until either a bivalent initial configuration has been found or the 1-valent initial configuration $C^0(0..001)$ has been reached; in $C^0(0..001)$, the processes $p_1, \ldots, p_{n-1}$ start with 0 and p_n starts with 1. We again consider the execution where p_n withholds its information from some process $q(p_n) = p_y \neq p_n$. For p_y, this execution is indistinguishable from the same one starting from $C^0(0..000)$, which must lead to a decision value of 0 by weak validity (C2). This contradicts the stipulated 1-valence of $C^0(0..001)$, however.

For the induction step $k \geq 1$, we assume that we have already reached a bivalent configuration C^{k-1} at the end of round $k-1$. We must show that at least one of the feasible successor configurations C^k that can be reached at the end of round k is bivalent. If this is true in the first place, we are done. If not, all successor configurations C^k must be univalent. However, the bivalence of C^{k-1} implies that at least one of those must be 0-valent and one must be 1-valent (i.e., C^{k-1} must be a *fork* [69]). By Lemma 2, there must also be 0-valent and 1-valent successor configurations C_0^k and C_1^k, respectively, that are neighbors. Assume that they differ only in the state of process r that has got some specific round k message correct in C_0^k but not or faulty in C_1^k (or vice versa). Now consider the two executions starting with C_0^k and C_1^k, where r withholds its round k-information from some process $q(r) = p_z \neq r$ in any future round $> k$. They are indistinguishable for p_z, which means that p_z and, by

agreement, all other processes must compute the same decision in both executions. This contradicts the stipulated different valences of C_0^k and C_1^k, however. □

Lemma 3 has a number of important consequences. First of all, it reveals an interesting asymmetry in the "severeness" of receive link failures (A^r) vs. send link failures (A^s). This can be seen by considering two instances of a 3-process system, where two processes A, B cannot communicate bidirectionally due to receive and/or send link failures: In the system shown in Fig. 3.2 (called of type R), processes A and B may not receive the messages from both peers due to excessive receive link failures ($f_\ell^r = 2$ and $f_\ell^s = 1$). In the system shown in Fig. 3.3 (of type S), processes A and B may fail to send to both peers due to excessive send link failures ($f_\ell^r = 1$ and $f_\ell^s = 2$).

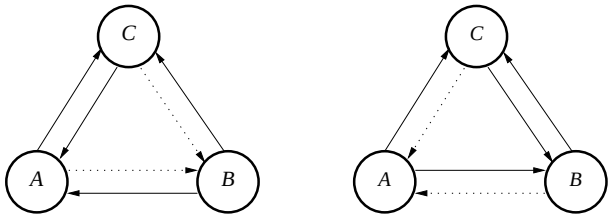

FIG. 3.2. 3-process system (type R), where processes A and B cannot communicate in one direction due to excessive receive link failures ($f_\ell^r = 2$ and $f_\ell^s = 1$). The left scenario shows the case $A \not\rightarrow B$, the right one $B \not\rightarrow A$, which may alternate arbitrarily.

It follows from Lemma 3 that no algorithm can solve consensus in a system of type R, even if process C is fixed and known to the algorithm. For, since A may fail to receive any information from any other process in the system, choosing $q(p) = A$ secures withholding by every $p \neq A$. Similarly, since B may also fail to receive the information from any peer, it provides the required $q(p)$ for withholding by process $p = A$. Hence, consensus is impossible in a 3-process system with $f_\ell^r = 2$ and $f_\ell^s = 1$; note that C is not fixed here, which makes consensus even harder to solve.

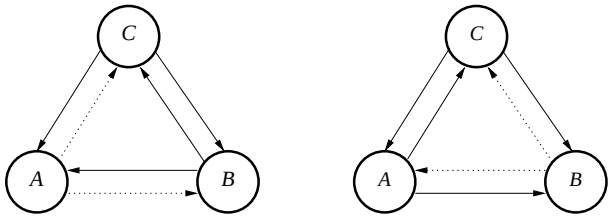

FIG. 3.3. 3-process system (type S), where processes A and B cannot communicate in one direction due to excessive send link failures ($f_\ell^r = 1$ and $f_\ell^s = 2$). The left scenario shows the case $A \not\rightarrow B$, the right one $B \not\rightarrow A$, which may alternate arbitrarily.

On the other hand, for systems of type S where C is fixed and known to the algorithm, there is a trivial solution that lets all processes decide upon the value of process C. If process C is fixed but not known, consensus can be solved by means of the algorithm described in [34]. No solution exists in a system of type S only if C is not fixed — as is the case in a 3-process system with $f_\ell^r = 1$ and $f_\ell^s = 2$.

As a final remark, we note that the above observations are in accordance with the results of [43], where it was shown that bounded-time consensus is impossible in the $\diamond$MFM ("Majority from Majority") link timing model.

4. Number of Processes. Using the results of Section 3, we will first establish a lower bound for purely omissive link failures ($f_\ell^{r,a} = f_\ell^{s,a} = 0$).

For the special case $f_\ell^s = f_\ell^r = f_\ell > 0$, such a lower bound can be inferred from Theorem 2 (even from Theorem 1), by a straightforward simulation-type proof: Assume that there is a deterministic algorithm $\mathcal{C}$ that solves consensus for $n = 2f_\ell$. Using $\mathcal{C}$, it is possible to construct a solution for consensus in a 2-process system with lossy links, which is impossible, however.

The detailed proof is as follows: Partition the n processes into two subsets P_A and P_B of size f_ℓ each. Two *super-processes* A and B are used to simulate the execution of the processes in P_A and P_B, respectively. All the links between the simulated processes in the two super-processes are routed over a single *super-link*. For a super-process' decision value, any simulated process' decision value can be taken. In order to ensure that $\mathcal{C}$ achieves consensus among all (simulated) processes, we must show that our link failure assumptions are not violated for any simulated process in any super-process in case of a super-link failure: Any simulated process must not get more than $f_\ell^r = f_\ell$ receive link failures, and must not produce more than $f_\ell^s = f_\ell$ send link failures. This is trivially satisfied since $f_\ell^s = f_\ell^r = f_\ell$, however. Hence, our solution would achieve consensus among the two super-processes, which violates Theorem 2 (even Theorem 1, since bidirectional partitioning could happen here).

For the general case of arbitrary f_ℓ^s and f_ℓ^r, the lower bound for omission link failures can immediately be derived from Lemma 3:

THEOREM 3 (Lower Bound Processes 1). *Any deterministic algorithm that solves consensus under our system model with $f_\ell^s, f_\ell^r > 0$ needs $n > f_\ell^r + f_\ell^s$.*

Proof. We first show that, for any process p, we can arbitrarily choose a set $\mathcal{P}$ of f_ℓ^r processes including p, where no process in $\mathcal{P}$ sends any messages to a process in $\mathcal{Q} = \Pi \backslash \mathcal{P}$ in case of $n = f_\ell^s + f_\ell^r$ in some feasible execution: Since there are $f_\ell^s \geq 1$ processes in $\mathcal{Q}$, every process in $\mathcal{P}$ may commit send link failures that omit all processes in $\mathcal{Q}$. Any process in $\mathcal{Q}$ thus experiences exactly f_ℓ^r receive link failures, which is also feasible with respect to our failure model. Hence, there is no information flow from processes in $\mathcal{P}$ to processes in $\mathcal{Q}$ at all, such that every process p can trivially withhold its information. According to Lemma 3, solving consensus is impossible here. □

Remarks:

1. According to Corollary 1 in Section 6, the lower bound $n > f_\ell^r + f_\ell^s$ provided by Theorem 3 is tight; it is for example matched by the authenticated algorithm ZA [65].
2. The result of Theorem 3 implies that, in order to be able to solve consensus, link failures (A^s) and (A^r) may affect at most a minority of processes only. In the setting of Gray's Theorem 1, however, there is no point in considering this case at all: There is no non-empty minority of processes for $n = 2$. Focusing on overly simple cases hence sometimes hides ways to escape from impossibility results.

In order to find a lower bound for arbitrary link failures, we will again start with the special case $f_\ell^s = f_\ell^r = f_\ell^{s,a} = f_\ell^{r,a} = f_\ell > 0$. Our derivation will be based on the following Theorem 4, which shows that no algorithm can solve consensus [with validity (C2')] in a 4-process system in the presence of a single arbitrary link failure per process ($f_\ell^{r,a} = f_\ell^{s,a} = 1$).

THEOREM 4 (4-Process Impossibility). *There is no deterministic algorithm that solves consensus with validity (C2') under our system model for a single arbitrary link*

failure in a 4-process system.

Proof. We employ a new instance of the "easy impossibility proof techniques" of [28] to show that any deterministic algorithm violates agreement if every process may see an inconsistent value from one of its neighbors. Suppose that our 4 processes execute a distributed algorithm consisting of specific programs A, B, C, D, which solve consensus under our system model with $f_\ell^{r,a} = f_\ell^{s,a} = f_\ell^s = f_\ell^r = 1$. In order to derive a contradiction, we arrange 8 processes in a cube as shown in Fig. 4.1. For example, the lower leftmost process labeled $A[0]$ executes algorithm A starting with initial value 0 (the $\boxed{0}$ on its left displays this process's decision value, as explained below). Note carefully that all processes and all links are assumed to be non-faulty here.

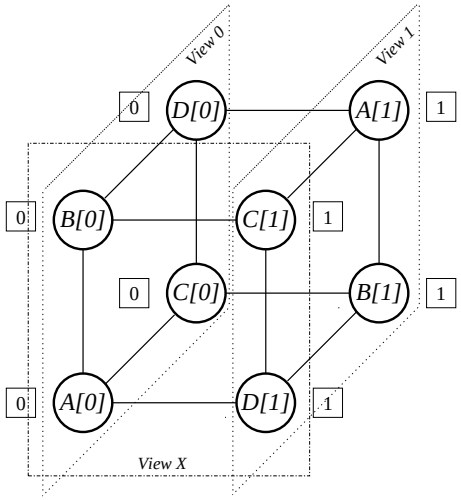

FIG. 4.1. Topology used for proving the violation of agreement in a 4-process system. 8 non-faulty processes with perfect links are arranged in a cube in a neighborhood-preserving way. The assignment of initial values ensures that all processes in view 0 (resp. view 1) decide 0 (resp. 1), but this violates agreement in view X.

Of course, dealing with a solution for a 4-process system, we cannot expect to achieve consensus in the 8-process system of Fig. 4.1. However, due to the special assignment of algorithms to processes, each process observes a neighborhood as in a 4-process system. More specifically, the 4 processes at any side of the cube (we call it a *view*) can be interpreted as an instance of a legitimate 4-process system. In fact, as can be checked easily, our assignment ensures that any process in a view is connected to exactly one process outside this view. Since we assumed that every process may see an arbitrary faulty input from at most one neighbor, the input from the process outside the view may be arbitrary—it just appears as a process the links of which deliver arbitrary faulty messages.

Now consider the processes in view 0, which all have initial value 0. By the validity property for consensus, all processes must decide 0 here (the initial value 1 of the processes outside view 0 do not matter, as the links to them are considered arbitrary faulty w.r.t. view 0). Similarly, in view 1, all processes must decide 1 since they have initial value 1. But now the processes in view X face a problem: Since e.g. the lower leftmost process $A[0]$ observes exactly the same messages in view X as in view 0 by construction, it must decide 0 as observed above. Similarly, as the

lower rightmost process $D[1]$ observes exactly the same messages in view X as in view 1, it must decide 1—but this would violate agreement in the 4-process system corresponding to view X. We hence established the required contradiction, thereby completing the proof of Theorem 4. □

Using Theorem 4, a similar simulation-type argument as in the pure omission failure case can be used to show the lower bound $n > 4f_\ell$ for $f_\ell^s = f_\ell^{s,a} = f_\ell^r = f_\ell^{r,a} = f_\ell > 0$ arbitrary link failures. Corollary 1 in Section 6 reveals that this lower bound is also tight; it is for example matched by the non-authenticated algorithm OMH [65, 66].

THEOREM 5 (Lower Bound Processes 2). *Any deterministic algorithm that solves consensus with validity (C2') under our system model with* $f_\ell^r = f_\ell^s = f_\ell^{s,a} = f_\ell^{r,a} = f_\ell > 0$ *needs* $n > 4f_\ell$.

Proof. Assume that there is a deterministic algorithm $\mathcal{C}$ that solves consensus for $n = 4f_\ell$ in our model. We use $\mathcal{C}$ to construct a solution for a 4-process system of Theorem 4, which provides the required contradiction.

We partition the set of all processes P into 4 subsets P_A, P_B, P_C, P_D of the same cardinality f_ℓ, and let each *super-process* A, B, C, D simulate all the instances of the algorithm in the respective subset. For the super-process' decision value, any simulated process' decision value can be taken. In order to ensure that $\mathcal{C}$ achieves consensus among all (simulated) processes, we must show that our link failure assumptions are not violated for any process in any super-process if at most one super-link per process may experience an arbitrary link failure. Since any super-link hosts the links to and from exactly f_ℓ processes, this is trivially fulfilled, however: In case of a super-link failure, every sender process commits at most f_ℓ send link failures (affecting the f_ℓ processes in the receiving super-process), and every receiver process experiences at most f_ℓ receive link failures (from the f_ℓ processes in the sending super process).

Therefore, we have constructed an implementation of a consensus algorithm for a 4-process system, which can withstand a single arbitrary link failure. Since this is impossible by Theorem 4, the proof of Theorem 5 is completed. □

Unfortunately, we did not find an easy way to generalize the above simulation-type argument for an arbitrary number $f_\ell^s, f_\ell^{s,a}, f_\ell^r, f_\ell^{r,a} \geq 0$ of link failures [and weak validity (C2)]. In order to derive a lower bound for n for this general case, we must hence resort to our key Lemma 3 again. What needs to be shown here, however, is that every process p can withhold its information: Lemma 5 below will prove that as many as $f_\ell^r + f_\ell^{r,a}$ processes can withhold their information from as many as $f_\ell^s + f_\ell^{s,a}$ processes in case of $n = f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$, provided that

$$\frac{f_\ell^{r,a}}{f_\ell^r} = \frac{f_\ell^{s,a}}{f_\ell^s}. \tag{4.1}$$

Hence, by Lemma 3, $n > f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$ is a lower bound for solving consensus if (4.1) holds. If (4.1) does not hold, there are cases where consensus can be solved also for $n \leq f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$. A lower bound in this case is $n > \overline{f}_\ell^r + \overline{f}_\ell^{ra} + \overline{f}_\ell^s + \overline{f}_\ell^{sa}$, however, where $\overline{f}_\ell^r \leq f_\ell^r$, $\overline{f}_\ell^{ra} \leq f_\ell^{r,a}$, $\overline{f}_\ell^s \leq f_\ell^s$, $\overline{f}_\ell^{sa} \leq f_\ell^{s,a}$ are such that (4.1) holds and n is maximal (see the comments prior to Theorem 6 for details).

Diving into the details of our lower bound proof, we start with a simple "balls and boxes" technical lemma. It shows that it is possible to drop white, orange and purple balls into a matrix such that each row and each column contains some specific numbers

of balls of each color. This result will be used subsequently to assert the existence of a certain mapping of send and receive link failures, by interpreting a white, orange, and purple ball as a correct, omission faulty, and arbitrary faulty transmission between a particular sender process (column index) and receiver process (row index).

LEMMA 4 (Balls and Boxes). *Consider a matrix with $s + s^a$ rows and $r + r^a$ columns, where $s \geq s^a > 0$ and $r \geq r^a > 0$ and*

$$r/r^a = s/s^a. \tag{4.2}$$

Then it is possible to drop white, orange, and purple balls into the matrix (one ball per entry), such that any single row contains exactly r^a white, $r - r^a$ orange, and r^a purple balls, whereas any single column contains exactly s^a white, $s - s^a$ orange, and s^a purple balls.

Proof. First, we note that summing up the number of balls of the same color by rows and columns, respectively, in any such assignment yields the same result: For example, we need $w_r = (s + s^a)r^a$ white balls when summing over rows, and $w_c = s^a(r+r^a)$ white balls when summing over columns. Since (4.2) implies $sr^a = s^a r$, it follows that $w_r = w_c$. We will now construct such an assignment explicitly.

Consider the first row in our matrix, and let

$$\pi_0, \pi_1, \ldots, \pi_{r+r^a-1}$$

with $\pi_i \in \{\mathit{white}, \mathit{orange}, \mathit{purple}\}$ be its assignment of balls to places according to the following rule: For any integer $x \geq 0$,

$$\pi_x = \begin{cases} \mathit{orange} \vee \mathit{purple} & \text{if } x = c(i) \text{ for some integer } i \geq 0, \\ \mathit{purple} & \text{if and only if } x = c\big(a(j)\big) \text{ for some integer } j \geq 0, \\ \mathit{white} & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} c(i) &= \left\lfloor \frac{r + r^a}{r} \cdot i \right\rfloor \\ a(j) &= \left\lfloor \frac{r}{r^a} \cdot j \right\rfloor \\ p(j) &= c\big(a(j)\big) = \left\lfloor \frac{r + r^a}{r} \cdot \left\lfloor \frac{r}{r^a} \cdot j \right\rfloor \right\rfloor. \end{aligned}$$

This assignment distributes colored (orange or purple), as well as purple balls alone, as regularly as possible over the $r + r^a$ available places in the first row. The following periodicity properties are immediately apparent from the above definitions: For $0 \leq i \leq r - 1$, $0 \leq j \leq r^a - 1$, and any integer $y \geq 0$,

$$0 \leq c(i) \leq r + r^a - 1 \quad \text{and} \quad c(i + r) = c(i) + r + r^a \tag{4.3}$$

$$0 \leq p(j) \leq r + r^a - 1 \quad \text{and} \quad p(j + r) = p(j) + r + r^a \tag{4.4}$$

$$\pi_{y+r+r^a} = \pi_y. \tag{4.5}$$

From the properties of $c(i)$ and $p(j)$, it follows immediately that $\pi_0, \pi_1, \ldots, \pi_{r+r^a-1}$ contains exactly r colored balls and r^a white ones. Clearly, every cyclic permutation (rotation) $\pi_y, \pi_{y+1}, \ldots, \pi_{y+r+r^a-1}$ of the original $\pi_0, \pi_1, \ldots, \pi_{r+r^a-1}$ has this property as well. Note that index addition in this cyclic permutation should actually be modulo

$r + r^a$; (4.5) reveals that this is automatically taken care of, however. Below, we will assign $\pi_y, \pi_{y+1}, \ldots, \pi_{y+r+r^a-1}$ to row y of our matrix to prove our lemma.

By using the equivalences $(r + r^a)/r = (s + s^a)/s$ and $r/r^a = s/s^a$, which follow immediately from (4.2), in the definitions of $c(i)$ and $p(i)$, we obtain similar periodicity properties for $0 \leq i \leq s-1$, $0 \leq j \leq s^a - 1$ and any integer $x \geq 0$:

$$0 \leq c(i) \leq s + s^a - 1 \quad \text{and} \quad c(i+s) = c(i) + s + s^a \tag{4.6}$$

$$0 \leq p(j) \leq s + s^a - 1 \quad \text{and} \quad p(j+s) = p(j) + s + s^a \tag{4.7}$$

$$\pi_{x+s+s^a} = \pi_x. \tag{4.8}$$

As before, this implies that $\pi_0, \pi_1, \ldots, \pi_{s+s^a-1}$ contains exactly s colored balls and s^a white ones. Even more, the periodicity properties (4.6) and (4.7) imply that every cyclic permutation (rotation) $\pi_x, \pi_{x+1}, \ldots, \pi_{x+s+s^a-1}$ of $\pi_0, \pi_1, \ldots, \pi_{s+s^a-1}$ has this property as well; again, (4.8) takes care of index addition modulo $s + s^a$.

Hence, we just have to assign $\pi_y, \pi_{y+1}, \ldots, \pi_{y+r+r^a-1}$ to row y of our matrix, meaning that the entry in column 0 of row y contains the same ball as the entry in column y of row 0, for example. Our findings on the number of balls in cyclic permutations of $\pi_0, \ldots, \pi_{r+r^a-1}$ shows that this assignment respects our lemma's requirement on rows. Similarly, inspection of the resulting matrix shows that column x contains the pattern $\pi_x, \pi_{x+1}, \ldots, \pi_{x+s+s^a-1}$, which respects our requirement on the number of balls in columns as well. For example, for $r = 4, r^a = 2, s = 2, s^a = 1$, we obtain the following assignment:

$$\begin{pmatrix} p & o & w & p & o & w \\ o & w & p & o & w & p \\ w & p & o & w & p & o \end{pmatrix}$$

□

Now we are ready to prove our major Lemma 5, which shows that, in case of $n = f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$ processes satisfying (4.1), any two executions that lead to two sufficiently "similar" configurations, in the sense that $|\mathcal{Q}| = f_\ell^s + f_\ell^{s,a}$ processes have identical state in both, can be extended by one round in a way that again yields two "similar" configurations for the processes in $\mathcal{Q}$. This implies that all the remaining $f_\ell^r + f_\ell^{r,a}$ processes can withhold their information in the resulting execution. Hence, Lemma 3 can be applied again, which will finally establish our general lower bound result.

LEMMA 5 (Similarity). *Consider two configurations $C = (c_1, \ldots, c_n)$ and $C' = (c'_1, \ldots, c'_n)$ generated by executions E and E' in a system of $n = f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$ processes satisfying $f_\ell^r / f_\ell^{r,a} = f_\ell^s / f_\ell^{s,a}$, where the states $c_1 = c'_1, \ldots, c_{f_\ell^s + f_\ell^{s,a}} = c'_{f_\ell^s + f_\ell^{s,a}}$ of $f_\ell^s + f_\ell^{s,a}$ processes are the same. Then, E and E' can be feasibly extended by one round, such that the same $f_\ell^s + f_\ell^{s,a}$ processes have again the same states $d_1 = d'_1, \ldots, d_{f_\ell^s + f_\ell^{s,a}} = d'_{f_\ell^s + f_\ell^{s,a}}$ in the resulting successor configurations $D = (d_1, \ldots, d_n)$ and $D' = (d'_1, \ldots, d'_n)$.*

Proof. Let $\mathcal{Q} = \{q_0, \ldots, q_{f_\ell^s + f_\ell^{s,a} - 1}\}$ be the set of processes with equal states in C and C', and $\mathcal{P} = \{p_0, \ldots, p_{f_\ell^r + f_\ell^{r,a} - 1}\}$ be the set of the remaining processes with possibly different states in C and C'. We claim that there is a feasible link failure pattern $\mathcal{F}$ extending E by one round, yielding the execution $E \cup (\mathcal{F}, D)$, where $\cup$ denotes the concatenation operation. Therefore, every process in $\mathcal{Q}$ gets exactly $f_\ell^{r,a}$ arbitrary link failures from some processes in $\mathcal{P}$, delivering the message that would

have been sent in the failure-free extension of E' (i.e., in the absence of link failures). In addition, every process in $\mathcal{Q}$ also experiences exactly $f_\ell^r - f_\ell^{r,a}$ omission link failures from some processes in $\mathcal{P}$, whereas the messages from the remaining $f_\ell^{r,a}$ processes in $\mathcal{P}$ are received correctly. All messages from processes in $\mathcal{Q}$ to processes in $\mathcal{Q}$, as well as all messages to processes in $\mathcal{P}$ are failure-free.

Not surprisingly, the required link failure pattern has already been established in Lemma 4: We just have to map $r = f_\ell^r$, $r^a = f_\ell^{r,a}$, $s = f_\ell^s$ and $s^a = f_\ell^{s,a}$ and interpret white, orange, and purple balls as correct, omission faulty, and arbitrary faulty transmissions from the $f_\ell^r + f_\ell^{r,a}$ processes in $\mathcal{P}$ (columns) to the $f_\ell^s + f_\ell^{s,a}$ processes in $\mathcal{Q}$ (rows). The results of Lemma 4 reveal that the corresponding link failure pattern respects both the maximum number of send and receive link failures.

Knowing that such an $\mathcal{F}$ indeed exists, our lemma follows from extending E with $\mathcal{F}$ and E' with the pattern matrix $\mathcal{F}'$, which is exactly $\mathcal{F}$ except that a process that committed an arbitrary send link failure in $\mathcal{F}$ transmits correctly in $\mathcal{F}'$, whereas a process that transmitted correctly in $\mathcal{F}$ commits an arbitrary send link failure in $\mathcal{F}'$, which (erroneously) delivers the message that would have been transmitted in the failure-free extension of E: After this round, every process in $\mathcal{Q}$ has the same view of the execution both in $E \cup (\mathcal{F}, D)$ and $E' \cup (\mathcal{F}', D')$ and hence reaches the same configuration in D and D' as asserted. □

Now it is not difficult to prove our general lower bound result as given by Theorem 6. It reveals that $n > f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$ is required for solving consensus if (4.1) holds. According to Corollary 1 in Section 6, this bound is tight and is for example matched by the exponential algorithm OMH [65,66]. If (4.1) does not hold, consensus can be solved with fewer processes, namely, with $n > \overline{f}_\ell^r + \overline{f}_\ell^{ra} + \overline{f}_\ell^s + \overline{f}_\ell^{sa}$ ones. The quantities $\overline{f}_\ell^r$, $\overline{f}_\ell^{ra}$, $\overline{f}_\ell^s$ and $\overline{f}_\ell^{sa}$ are non-negative integers solving (4.1), subject to the constraints $\overline{f}_\ell^r \le f_\ell^r$, $\overline{f}_\ell^{ra} \le f_\ell^{r,a}$, $\overline{f}_\ell^s \le f_\ell^s$ and $\overline{f}_\ell^{sa} \le f_\ell^{s,a}$, such that $\overline{f}_\ell^r + \overline{f}_\ell^{ra} + \overline{f}_\ell^s + \overline{f}_\ell^{sa}$ is maximal. Note that this is in accordance with the observation that, if (4.1) does not hold, OMH solves consensus also for $n = f_\ell^r + f_\ell^{r,a} + f_\ell^s + f_\ell^{s,a}$ (if it is allowed to execute some additional rounds). Although we do not know whether the lower bound given by Theorem 6 is also tight in this case, we nevertheless conjecture that this is the case.

THEOREM 6 (Lower Bound Processes 3). *Any deterministic algorithm that solves consensus under our system model needs* $n > \overline{f}_\ell^r + \overline{f}_\ell^{ra} + \overline{f}_\ell^s + \overline{f}_\ell^{sa}$, *where* $\overline{f}_\ell^r \le f_\ell^r$, $\overline{f}_\ell^{ra} \le f_\ell^{r,a}$, $\overline{f}_\ell^s \le f_\ell^s$, $\overline{f}_\ell^{sa} \le f_\ell^{s,a}$ *are such that* $\overline{f}_\ell^r / \overline{f}_\ell^{ra} = \overline{f}_\ell^s / \overline{f}_\ell^{sa}$ *holds and the sum* $\overline{f}_\ell^r + \overline{f}_\ell^{ra} + \overline{f}_\ell^s + \overline{f}_\ell^{sa}$ *is maximal.*

Proof. Due to Theorem 3, it only remains to provide an impossibility proof for $f_\ell^{s,a}, f_\ell^{r,a} > 0$. According to Lemma 3, we just have to show that every process p can withhold its information under the conditions of our theorem: More specifically, for every $k \ge 0$, we need a failure pattern for rounds $k+1, k+2, \ldots$ such that a non-empty set of processes $\mathcal{Q} = \mathcal{Q}(p)$ has the same view of the resulting execution after round $r \ge k+1$, independent of the information p has gathered by round r.

This follows easily from inductively applying Lemma 5, however: If we assume that p is just one of the $f_\ell^r + f_\ell^{r,a}$ processes in $\mathcal{P}$ that may have a different state in two k-round executions E (resp. E') leading to configurations C (resp. C'), we are guaranteed that the remaining $|\mathcal{Q}| = f_\ell^s + f_\ell^{s,a}$ processes that had identical state in E and E' have identical state in some 1-round extension $E \cup (\mathcal{F}, D)$ and $E' \cup (\mathcal{F}', D')$ of E and E' again. Hence, no such process ever gets information from p. Since this can go on for an arbitrary number of rounds, Definition 1 reveals that every p can indeed withhold its information as required. □

5. Number of Rounds. In this section, we will show that being able to handle link failures comes at the price of additional running time. More specifically, compared to the case without link failures, solving consensus in case of $f_\ell^s, f_\ell^r > 0$ requires one additional round. Our proofs are again based on bivalency arguments and re-use some of the results developed in the previous sections.

THEOREM 7 (Lower Bound Rounds 1). *Any deterministic algorithm that solves consensus under our system model for $f_\ell^s, f_\ell^r > 0$ needs at least 2 rounds.*

Proof. Assume that there is a 1-round algorithm that solves consensus in the presence of link failures. Obviously, since any process p may suffer from a send link failure to any receiver process q, any process p can withhold its information from at least one process $q(p)$ here: The 1-round assumption does not allow other processes to learn about p's information in some later round. Therefore, Lemma 3 reveals that solving consensus is impossible here. Note that Lemma 3 is applicable here, since x-round consensus is an instance of consensus where no messages are sent in rounds $> x$. So if no consensus algorithms exists, as guaranteed by Lemma 3, x-round consensus is impossible either. □

The above result can be extended to the case where both process and link failures can occur. Using our ideas in the simple bivalency proof of the well-known $f + 1$ lower bound for the number of rounds required for solving consensus in the presence of f process crashes developed in [6], it is possible to show that $f + 2$ is a tight lower bound (matched for example by algorithm OMH of [65, 66], and also confirmed by Corollary 1 for $f = 0$).

THEOREM 8 (Lower Bound Rounds 2). *Any deterministic algorithm that solves consensus under our system model with $n \geq f + f_\ell^s + f_\ell^r$, where f denotes the maximum number of process crash failures and $f_\ell^s, f_\ell^r > 0$, needs at least $f + 2$ rounds in the worst case.*

Proof. In [6], a simple forward induction based on a bivalency argument involving message losses due to process crashes is used to show that any consensus algorithm has executions that lead to at least one bivalent configuration at the end of round $f - 1$. The executions considered in this proof are such that at most one process may crash in every round; clearly, no link failures are assumed to occur here. The impossibility of consensus within f rounds follows by contradiction: It is shown in [6, Lemma 1] that the existence of such a solution would imply that all configurations reached after $f - 1$ rounds must be univalent.

In order to prove Theorem 8, we only have to provide an analogue to [6, Lemma 1]: That the existence of a consensus algorithm that decides after $f + 1$ rounds in our failure model would imply that all configurations reached after $f - 1$ rounds are univalent. The proof is by contradiction: Assuming that not all configurations reached after $f - 1$ rounds are univalent, there must be a bivalent configuration C^{f-1} after round $f - 1$. Since at most one process may have crashed during each of the first $f - 1$ rounds, there is still one process p that may crash in round f or $f + 1$. Note that it is the crash of this process p and/or the occurrence of link failures in round f or $f + 1$ that "allows" C^{f-1} to be bivalent.

Let v be the algorithm's decision in the execution E, where no failure (i.e., neither a crash of p nor any link failure) occurs in the two rounds following C^{f-1}. Due to the bivalence of C^{f-1}, there must be a different execution $\overline{E}$ also starting from C^{f-1}, where the decision is $1 - v$. Assume first that p crashes in round f in $\overline{E}$. Then, there must be two executions E^q leading to the decision value v, and $\overline{E}^q$ leading to $1 - v$, which differ only in that the (crashing) p sends its round f message to q in E^q but

not in $\overline{E}^q$: Starting from E where p sends all its messages, we remove the messages p succeeds to send one by one until the decision value changes; this happens at latest when we arrive at the execution $\overline{E}^q$.

By construction, q is the only process that can distinguish between E^q and $\overline{E}^q$ after round f. If we allow q to produce a send link failure to some other correct process r (this process must exist since $n \geq f+2$) in the final round $f+1$, then r has the same view at the end of E^q and $\overline{E}^q$. Hence, the resulting decisions cannot be different, providing the required contradiction of some C^{f-1} being bivalent in this case.

We still have to deal with the case where p does not crash in round f in $\overline{E}$. Then, we claim that there is some bivalent configuration C^f reachable from C^{f-1} in round f. For the sake of contradiction, assume that all configurations reachable from C^{f-1} in a single round (where no process crashes) are univalent. Since C^{f-1} is bivalent, the configuration C^f reached by the link-failure-free single-round extension of C^{f-1} must be v-valent, whereas some configuration $\overline{C}^f$ reached by another single-round extension with link failures must be $(1-v)$-valent. Due to Lemma 2, there are two neighboring configurations C^f_q and $\overline{C}^f_q$ that are also v-valent and $(1-v)$-valent, respectively. Those configurations differ only in a single link failure from some sender $s \to q$ in round f, perceived by q in $\overline{C}^f_q$ but not in C^f_q.

Again, q is the only process that can distinguish between the resulting executions E^q and $\overline{E}^q$ after round f. If we allow q to produce a send link failure to some other correct process r (this process must exist since $n \geq f+2$) in the final round $f+1$, then r has the same view at the end of E^q and $\overline{E}^q$. Therefore, the resulting decisions cannot be different, contradicting the stipulated univalence of all C^f. Hence, there is indeed some reachable bivalent configuration C^f at the end of round f.

Since we still have a processor p to crash in round $f+1$ in this case, however, the original [6, Lemma 1] applies: In order to decide at the end of round $f+1$, all configurations at the end of round f must be univalent. Since C^f is bivalent, we have established the required contradiction also in this case. □

As a concluding remark, we note that the additional round required for solving consensus in the presence of link failures is not a new result. For $f=0$, it has been shown in [45] that 2 rounds are needed for solving consensus. Interestingly, the general case follows also from the general result of [25] on indulgent consensus algorithms. More specifically, link omission failures can be interpreted as false suspicions of a local failure detector module. Our algorithms tolerate such link failures, hence must be indulgent w.r.t. their "failure detectors" (= message reception). Note that our constraints (A^s) and (A^r) also ensure termination of such algorithms.

6. Other Results. In this section, we will elaborate on some consequences of our model and the results obtained so far. We start with some considerations related to connectivity in the underlying communication graph in our model, which can be used for confirming the tightness of our lower bounds $n > f^s_\ell + f^r_\ell$ and $n > f^s_\ell + f^{s,a}_\ell + f^r_\ell + f^{r,a}_\ell$.

Consider the single-round communication graph G for some round k. It consists of n vertices corresponding to the processes in Π, and contains a directed edge (p,q) iff there is no link failure on the link connecting $p \to q$ in round k. Recall from elementary graph theory that a graph G is *at least c-connected* if it remains connected when at most $c-1$ vertices and their adjacent edges are removed. Two paths connecting processes p and q are called *process-disjoint* iff they do not have common processes

except p and q.

THEOREM 9 (Connectivity). *Every single-round communication graph G of a system of $n > f_\ell^s + f_\ell^r$ processes complying to our system model is at least c-connected with $c = n - f_\ell^s - f_\ell^r > 0$. In fact, every pair of processes p, q is connected by c process-disjoint paths consisting of at most 2 non-faulty links.*

Proof. Since at least c-connectivity follows trivially if every pair of processes is connected by c process-disjoint paths, it suffices to show the latter: From (A^s), we know that p is connected to at least $n - f_\ell^s$ processes (possibly including itself) via non-faulty links. From (A^r), it follows that q is connected to a set of at least $n - f_\ell^s - f_\ell^r = c$ of these processes via non-faulty links. Let $\mathcal{I}$ with $|\mathcal{I}| \geq c$ be this set of processes. If $p \notin \mathcal{I}$ and $q \notin \mathcal{I}$ (i.e., if p and q are not adjacent), then p and q are connected by c paths consisting of 2 non-faulty links routed over the processes in $\mathcal{I}$. Otherwise, there are only $c-1$ paths of length 2 and a direct path from p to q, which are of course also process disjoint. □

Theorem 9 implies that one can build a 2-round simulation of reliable communication in our model, by using the well-known echo broadcasting scheme [15] ("crusader's agreement" [23]): p sends (msg, p) to all in the first round, and every q rebroadcasts (msg, p) in the second round. One can easily show that this broadcasting scheme allows every q to deliver (msg, p) correctly if $c = n - f_\ell^s - f_\ell^r > 0$. Interestingly, as proved in [11, 12], this simulation even works in case of arbitrary link failures if n is sufficiently large:

COROLLARY 1 (Reliable link simulation [11, Thm.2]). *There is a 2-round simulation, which implements reliable broadcasting in our link failure model if $n - f_\ell^s - f_\ell^{s,a} - f_\ell^r - f_\ell^{r,a} > 0$.*

Any synchronous consensus algorithm resilient to f classic process failures can hence be used in conjunction with this simulation for solving consensus in the hybrid failure model of [65, 66] (and therefore, trivially, in the model of Section 2). Note, however, that this simulation doubles the number of rounds and is hence sub-optimal: Theorem 8 revealed a lower bound of $f+2$ rounds for solving consensus in our model, and the algorithms provided in [65] confirm that this bound is tight.

Certain consequences of the results of the previous sections also shed some light on classic process failures. After all, the effect of an omission (resp. arbitrary) faulty process on its peers is principally the same as the effect of omission (resp. arbitrary) send link failures (A^s) committed by a non-faulty process: Some receiving processes inconsistently get no (resp. erroneous) messages instead of the correct ones. So the question arises why f omission faulty processes require at least $f+1$ rounds of execution (Theorem 8), whereas any number of processes committing send link failures can be handled in just 2 rounds according to Corollary 1, cp. the exponential algorithm OMH [65, Thm. 5.4], for example.

From our link failure model, it is apparent that arbitrary send link failures (A^s) can also be viewed as the consequence of a *restricted process failure* with inconsistency limited to f_ℓ^s. Since $f_\ell^s < n$, however, the inconsistency caused by send link failures is strictly less than that of an arbitrary faulty process, since the latter is not restricted in the number of recipients that may get an erroneous message. If at most $f_\ell^r = f_\ell^{r,a}$ processes suffer from restricted process failures with inconsistency limited to $f_\ell^s = f_\ell^{s,a}$, both (A^s) and (A^r) are satisfied, which implies that this alternative interpretation leads to feasible link failure patterns in our model. Note carefully that (A^s) and (A^r) admit more general failure patterns than just restricted process failures, however: A

receive failure may hit any incoming link in the former, but is restricted to one of the links from the f_ℓ^r restricted faulty processes in the latter.

Anyway, using the alternative interpretation of arbitrary link failures as restricted arbitrary process failures, the result of Theorem 6 allows us to characterize what makes a process failure a truly arbitrary (Byzantine) one: Choosing $f_\ell^r = f_\ell^{r,a} > 0$ arbitrary and fixing $n > 2f_\ell^{r,a}$, an optimal consensus algorithm such as OMH solves consensus with $n \geq 2f_\ell^{s,a} + 2f_\ell^{r,a} + 1$ in only two rounds. It hence tolerates $f_\ell^{r,a}$ restricted arbitrary process failures with inconsistency limited to

$$f_\ell^{s,a} \leq \lfloor (n-1)/2 \rfloor - f_\ell^{r,a}. \tag{6.1}$$

For example, $n = 9$ processes are required for tolerating two restricted arbitrary failures with inconsistency $f_\ell^{s,a} = 2$ in just two (instead of three) rounds. Due to (6.1), at least $\lceil (n-1)/2 \rceil + f_\ell^{r,a}$ processes, i.e., a majority[4] of the non-faulty processes, get the correct message even in the broadcast of a restricted arbitrary faulty process. Provided that n is chosen appropriately, *any* number $f_\ell^{r,a}$ of process failures with inconsistency limited to $f_\ell^{s,a}$ can hence be tolerated in just two rounds here, i.e., in a number of rounds that does not depend on the number of failures $f_\ell^{r,a}$. If, on the other hand, more than $f_\ell^{s,a}$, i.e., more than a minority of the non-faulty processes, can get a faulty message in the broadcast of a process, then the sender must be considered arbitrary faulty and thus increases the number of rounds required for solving consensus.

A similar observation can be made for omission failures. Choosing $f_\ell^{s,a} = f_\ell^{r,a} = 0$, $f_\ell^r > 0$ arbitrary and fixing $n > f_\ell^r$, an optimal consensus algorithm such as ZA [65] solves consensus for $n \geq f_\ell^r + f_\ell^s + 1$ in only two rounds. It hence tolerates f_ℓ^r restricted omission process failures with inconsistency

$$f_\ell^s \leq n - 1 - f_\ell^r \tag{6.2}$$

It follows from (6.2) that at least $f_\ell^r + 1$ processes, i.e., at least one non-faulty process, must get the correct message even in the broadcast of a restricted omission faulty process. If this is warranted, any number f_ℓ^r of such restricted process failures can be tolerated within 2 rounds.

It hence follows that a process that disseminates inconsistent information cannot do much harm—in the sense of requiring additional rounds for solving consensus—if at most a certain number of recipients can get inconsistent information:

- A process must be considered arbitrary faulty if it can supply erroneous information to a majority of the non-faulty processes.
- A process must be considered omission faulty if it can fail to provide information to any and all non-faulty processes.

For sub-optimal algorithms, the thresholds (numbers of affected receivers) are of course smaller.

Note finally that our observations do not contradict the lower bound $f + 1$ for the number of rounds required for consensus in the presence of f faulty processes, recall Theorem 8 or [8, Sec. 5.1.4], since this result relies heavily on the fact that a faulty process can disseminate inconsistent information to as many processes as desired.

[4]For a sub-optimal algorithm, even more than a majority of the non-faulty processes must get the correct message here.

7. Relation to other Models. In this section, we will relate our model to alternative link failure modeling approaches. Particular emphasis will be put on the issue of assumption coverage, which will be analyzed and compared in detail in a simple probabilistic setting.

7.1. Overview of related approaches. It has long been known that consensus is impossible with arbitrarily lossy links [33]. Therefore, every useful failure model must restrict link failures in some way. Some previous work has considered links that eventually become reliable "for sufficiently long", at least among a majority of the processes, (e.g., [24,46]), or "stubborn links" [36] that eventually deliver every message provided the message is sent sufficiently many times. These models in essence provide safety despite link failures, but require communication to eventually become reliable in order to ensure liveness. A similar approach is employed in the crash-recovery model [2], which also deals with transient failures, albeit at the level of whole processes and on a much larger time-scale. There are only a few failure models for synchronous systems in the literature that deal with transient link failures that continue to occur indefinitely.

One straightforward way to deal with link failures is to map link failures to sender process failures [32] or, preferably, to general send/receive-omission failures [54]. Unfortunately, this approach suffers from poor model coverage (see Section 7.2), and is also quite restrictive in the sense that only specific processes—the faulty ones—may experience link failures.

Another class of models [55,61,67] considers a small number of link failures explicitly: Those papers assume that at most $O(n)$ links may be faulty system-wide during the entire execution. Obviously, such models can only be applied when link failures are rare. Hadzilacos [37] presents a theoretical study of connectivity requirements for solving consensus in case of arbitrary networks with stopping and omission failures.

The most severe problem of the models surveyed so far is their inability to deal with the "moving" nature of transient link failures: In a real network, there is a positive probability for message loss (or delay) on every link. In the aforementioned models, once a single message is lost, either the link or the process is deemed faulty. Since failures are considered persistent during an execution in the above models, the "exhaustion" of non-faulty processes and links progresses rapidly with every round. This makes any attempt to solve consensus in models such as those presented in [32, 37, 54] void in case of significant link failure rates (see Section 7.2 for a detailed analysis). A more adequate approach to capture message loss is to allow for transient failures that hit different processes or links in different communication rounds.

Santoro and Widmayer were the first to introduce this assumption: In [58, 60], they showed that consensus cannot be solved in the presence of $n-1$ (resp. $\lfloor n/2 \rfloor$) omission (resp. Byzantine) link failures per round, in particular, if those link failures hit the same sender process. As a consequence, consensus cannot be solved in the presence of just a single *mobile* omission or Byzantine faulty process, i.e., a single process—which may be different in different rounds—that suffers from omission or even Byzantine failures. This result has been proven in [58] by means of a similar approach as employed in Section 3, and re-proven in the layering framework by Moses and Rajsbaum [52].

On the other hand, if the number of moving link failures is further restricted to less than $n-1$ (resp. $\lfloor n/2 \rfloor$) per round in case of omission (resp. Byzantine) link failures, consensus can be solved in a constant number of rounds [49, 59]. Other distributed computing problems [21] and special system architectures [22] have also been studied

under this model.

The failure model introduced in [56] can be seen as a first step in the direction of increasing the link failure resilience from $O(n)$ to $O(n^2)$: For a system with $n \geq 20f+1$ processes, at most f of which may be Byzantine faulty per round, a consensus algorithm was given that tolerates a small number $l = n/20$ of link failures at every node. A different (but related) model has been proposed in [31], which considers at most f Byzantine process failures per round that may move from one process to another with a certain maximum speed.

Cristian et al. [20] provide a suite of synchronous atomic broadcast protocols with much better link failure resilience (which is comparable to results we presented in an earlier paper [65]). Although atomic broadcast is usually investigated in a more communications-oriented context, it can be used to solve consensus as well, see e.g. [38] for an overview. The three algorithms of Cristian et al. [20] tolerate an arbitrary number of processes with omission, timing, or Byzantine failures (if authentication is available) and work on general communication graphs subject to link failures. Instead of making the number of link failures explicit, however, it is just assumed that any two processes in the system are always connected via a path of non-faulty links.

Unlike in the deterministic setting, link failures are easily tolerated by *randomized* consensus algorithms such as the one of [70]. Such algorithms circumvent Gray's impossibility result (cf. Theorem 1) by adding non-determinism (coin tossing) to the computations. Still, due to the inherent non-zero probability of failure/non-termination within a fixed number of rounds, randomized algorithms are unsuitable for some applications. Moreover, there is a lower bound $1/(R+1)$ for the probability of disagreement after R rounds with arbitrary loss [50, Thm. 5.5]. It is interesting to note, however, that our link failure modeling approach also circumvents this lower bound: For a well-known randomized algorithm, Schmid and Fetzer established a probability of disagreement of only $(1/2)^R$ [64].

Though our paper primarily deals with message omissions in the synchronous timing model, our model can also be used to reason about *timeliness* of some links in otherwise asynchronous round-based systems (where late messages are discarded). In this context, our threshold f_ℓ^r (resp. f_ℓ^s) can be seen as a restriction on the number of late (or untimely) messages a process receives (resp. sends) in a round. Restrictions of this type have received much attention recently: A suite of papers [3–5, 7, 39, 43, 44, 51, 53] provided weaker and weaker models that are still sufficiently strong for implementing failure detectors and/or solving consensus in the presence of at most f process crash failures and dynamic timing variations.

In particular, Keidar and Shraer introduced a model called *All From Majority* ($\diamond$AFM) in their round-by-round GIRAF framework [43], which is closely related to the moving link failure model introduced in Section 2.2: It allows $O(n^2)$ links per round to be non-timely, provided that every process has at least $m+1$ timely outgoing links and $n-m$ timely incoming links at any time, for some suitable m. The set of timely links may be moving. $\diamond$AFM was shown in [44] to be the only model (out of those defined thus far in this context) that scales with n, in the sense that consensus can be guaranteed to terminate in a constant expected number of rounds in the independent identically distributed probabilistic link failure model even for $n \to \infty$.

Our model is also related to the *Heard-of Model* (HO Model) developed by Charron-Bost and Schiper [18]. The HO Model is a round-based distributed computing model, which unifies synchrony and (benign) failures of both processes and links.

It has recently been extended to Byzantine failures [13] as well. Our link failure restrictions (A^s) and (A^r) can be elegantly expressed as simple communication predicates in the HO model, namely, $\forall p \in \Pi, k \geq 1 : |HO(p,k)| \geq n - f_\ell^r \wedge |TT(p,k)| \geq n - f_\ell^s$ in case of omission link failures, where $HO(p,k)$ is the set of processes p hears from in round k and $TT(p,k)$ is the set of processes p talks to in round k. Moreover, the reliable link simulation that led to Corollary 1 can be seen as a 2-round translation that simulates a global kernel (of size n, i.e., a failure-free system) in our model. It is also interesting to compare the upper bound results of [13] with our lower bounds: Theorem 5 reveals that we can allow at most $n/4$ arbitrary link failures per round. The algorithm $\mathcal{A}_{T,E}$ in [13] admits up to $n/4$ arbitrary receive link failures per round, without posing a restriction to send link failures, however. There is no contradiction here, due to the fact that the analysis of $\mathcal{A}_{T,E}$ separates safety and liveness: The algorithm actually needs some rounds with much less than $n/4$ link failures for guaranteed termination. By contrast, our lower bounds guarantee both safety and liveness simultaneously.

Finally, we already noted that, in the context of round-by-round fault detectors [30], false suspicions of a local failure detector [16] can also be interpreted as transient link failures. Our results, such as the lower bound of $f+2$ rounds for solving consensus, are hence also applicable to stable periods [26] and stable runs [25,41] of indulgent [35] consensus algorithms.

An alternative way to cope with transient link failures are reliable link simulation protocols based on retransmissions [1, 2, 9]. Asynchronous algorithms can then be used atop of such protocols for solving consensus. However, since retransmission protocols can obviously mask omissions only (but not timing failures and/or erroneous messages), they are no panacea. Moreover, using time redundancy for tolerating link failures necessarily increases the end-to-end delay in case of a failure, which eventually affects the consensus algorithms' termination time. And last but not least, since it is impossible to solve consensus in asynchronous systems with even a single crash failure [29], one has to add some synchrony to the system anyway [3–5, 7, 39, 43, 51, 53]. This makes our synchronous lower bound results applicable again, at least to asynchronous algorithms designed for round-by-round-based frameworks such as [19,43,64]. A detailed survey of link failures in partially synchronous and asynchronous systems can be found in [63].

Note that using reliable link protocols in conjunction with *synchronous* consensus algorithms is not particularly useful, since the duration of the rounds must be fixed a priori. As a consequence, only a certain number of retransmissions can be accomodated in a round, which is not sufficient for simulating reliable links in the presence of high link failure rates. In sharp contrast, our approach towards handling link failures uses additional processes (i.e., some larger value of n) instead of retransmissions and therefore does not suffer from this problem: The duration of a round just needs to encompass a single end-to-end delay.

7.2. Model coverage. In this section, we will prove that our link failure model also surpasses alternative approaches in terms of assumption coverage.

If link failures are just mapped to sender process failures, as in [32], even a single link failure per process and round ($f_\ell = 1$ in our terminology) would end up with $f = n$ faulty processes in some runs. Fig. 7.1 shows an example for $n = 4$ and $f_\ell = 1$.

In the more elaborate send/receive-omission failure model of [54], an omission can be attributed to either the sender or the receiver process. Still, in the example of Fig. 7.1, only at most two processes can be considered correct. Ending up with

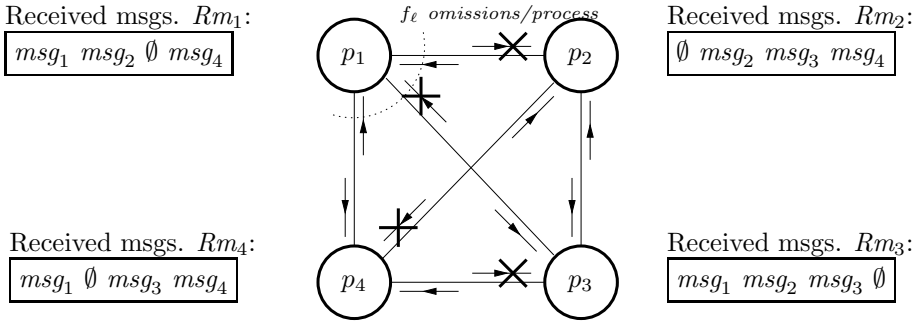

FIG. 7.1. Example of a 4-process system with $f_\ell = 1$ send and receive omission failures per process in each round, where all processes must be considered faulty in traditional process failure models.

less than a majority of correct processes renders uniform consensus unsolveable [17], however. Hence, by attributing link failures to processes, we may miss the opportunity to solve consensus in scenarios which can be handled in our model.

We now turn to examine whether the additional scenarios captured by our model are significant. After all, one could argue that failure patterns as depicted above almost never occur in practice, such that more refined models have only marginal added value. We counter this argument by quantifying the *coverage*[5] of various models, using a simple probabilistic "benchmarking scenario":

DEFINITION 2. *Consider a synchronous system of n processes, where each of the $n(n-1)$ unidirectional links fails with some independent probability $0 \leq p < 1$, independently in every round. The* coverage *$Cov(M)$ of a model M is the probability that the model assumptions hold true during the execution of an m-round algorithm, for some arbitrary $m \geq 1$.*

Combining ideas from [62] and [44], we analyze the coverage of the following failure models:

- f general omission-faulty processes (GO_f) [54],
- at least one process with f non-moving timely links (TL_f) [5],
- at most $n-2$ moving link failures per round (ML_{n-2}) [58],
- f_ℓ moving link failures per process and round (MLO_{f_ℓ}), the model of Section 2.2.

It will turn out that the link failure model introduced in Section 2.2 surpasses all other modeling approaches above in terms of coverage. In particular, it is the only model that scales with n, in the sense that for example $\mathrm{Cov}(MLO_{n/2}) \to 1$ for $n \to \infty$. By contrast, the coverage of all the other models even goes to 0 for $n \to \infty$ in comparable settings.

We should mention, though, that our coverage analysis does not aim at replacing a direct analysis of a distributed algorithm in our probabilistic "benchmarking scenario". A particular algorithm $\mathcal{A}$ may perform much better than our coverage analysis predicts, since $\mathcal{A}$ may also work well in executions where the deterministic model M is violated. Hence, $\mathrm{Cov}(M)$ is just a lower bound on the achievable performance of $\mathcal{A}$, but is of course a meaningful measure for assessing the quality of a deterministic

[5]We note that the term coverage suffers from overloading in the literature; throughout this paper, "coverage" must be read as "model coverage in synchronous systems with independent link failure probability p".

model independently of a particular protocol.

Analysis of f general omission-faulty processes $\mathbf{GO_f}$. Under this failure model, there must be a set K of at least $k = n - f$ processes that never commit a send nor a receive omission. This requirement is mapped to our probabilistic link failure setting as follows: All the links among the processes in K must be correct in all rounds, and the links connecting processes in $\Pi\backslash K$ with processes in $\Pi\backslash K$ may be either correct or faulty. The links to/from processes in K from/to processes in $\Pi\backslash K$ may also be either correct or faulty, since we can attribute a link failure to the adjacent process in $\Pi\backslash K$.

So let K be a subset of $k = n - f \geq 1$ distinct processes where all $k(k-1)$ links among processes in K never experience any omission failure during m rounds. Since there are $\binom{n}{k}$ different sets K of k processes out of n processes, the probability $P_n(k)$ that there is at least one such set in a run satisfies

$$P_n(k) \leq \binom{n}{k}(1-p)^{k(k-1)m}. \tag{7.1}$$

Note that $P_n(k)$ is not equal to $\binom{n}{k}(1-p)^{k(k-1)m}$, since there may be multiple sets K in a given run [recall that the links outside K can also be correct].

Hence, the model coverage of the standard general omission failure model with at most f faulty processes GO_f satisfies

$$\mathrm{Cov}(GO_f) = P_n(n-f) \leq \binom{n}{n-f}(1-p)^{(n-f)(n-f-1)m}. \tag{7.2}$$

In case of $f = \lambda n$ for any $0 < \lambda < 1$, it is not difficult to prove that $\mathrm{Cov}(GO_{\lambda n}) \to 0$ for $n \to \infty$. We will use asymptotic analysis for this purpose[6], with Stirling's formula

$$n! \sim \sqrt{2\pi n} \cdot \Big(\frac{n}{e}\Big)^n \qquad \text{for } n \to \infty \tag{7.3}$$

as our major ingredient.

LEMMA 6 (Asymptotic expansion $\binom{n}{n\lambda}$). *For any $0 < \lambda < 1$ and $n \to \infty$,*

$$\binom{n}{\lambda n} \sim \frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}} \cdot \Big((1-\lambda)^{-(1-\lambda)} \cdot \lambda^{-\lambda}\Big)^n. \tag{7.4}$$

Proof. Using Stirling's formula in $\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$, we find

$$\begin{aligned}
\binom{n}{\lambda n} &\sim \frac{\big(\frac{n}{e}\big)^n}{\big(\frac{n(1-\lambda)}{e}\big)^{n(1-\lambda)}\big(\frac{\lambda n}{e}\big)^{\lambda n}} \cdot \frac{\sqrt{2\pi n}}{\sqrt{2\pi n(1-\lambda)} \cdot \sqrt{2\pi\lambda n}} \\
&\sim \Big(\frac{n}{n(1-\lambda)}\Big)^n \cdot \Big(\frac{n(1-\lambda)}{\lambda n}\Big)^{\lambda n} \cdot \frac{1}{\sqrt{1-\lambda} \cdot \sqrt{2\pi\lambda n}} \\
&\sim (1-\lambda)^{-n} \cdot \Big(\frac{1-\lambda}{\lambda}\Big)^{\lambda n} \cdot \frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}},
\end{aligned}$$

from which (7.4) follows immediately. □

[6]We use the notation $f(n) \sim g(n) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n) = 1$, and $f(n) \propto g(n) \Leftrightarrow 0 \leq \lim_{n\to\infty} f(n)/g(n) \leq 1$.

Using the result of Lemma 6 in (7.2), the coverage of $GO_{n\lambda}$ evaluates to

$$\text{Cov}(GO_{n\lambda}) \propto \frac{\left((1-\lambda)^{-(1-\lambda)} \cdot \lambda^{-\lambda}\right)^n}{\sqrt{2\pi n\lambda(1-\lambda)}} \cdot (1-p)^{\left(n(1-\lambda)\right)\left(n(1-\lambda)-1\right)m}. \tag{7.5}$$

Since the exponent of $1-p<1$ is quadratic in n, it is obvious that $\text{Cov}(GO_{n\lambda}) \to 0$ for $n \to \infty$, for any $p>0$, $0<\lambda<1$, and $m \geq 1$. In particular, for $f = n/2$, we obtain

$$\text{Cov}(GO_{n/2}) \propto \frac{2^n}{\sqrt{\pi n/2}}(1-p)^{(n/2)(n/2-1)m}. \tag{7.6}$$

$\text{Cov}(GO_{n/2})$ hence very quickly goes to 0 for $n \to \infty$ for any $0<p<1$ and any $m \geq 1$. The following Figure 7.2 gives some numerical values, which reveal that the coverage of the general omission process failure model in our benchmarking scenario is indeed very poor. For example, for $p = 0.01$ and $m = 2$ (first plot in the left figure), the coverage is only about 10^{-25} in a system of $n = 40$ processes (log-scale)!

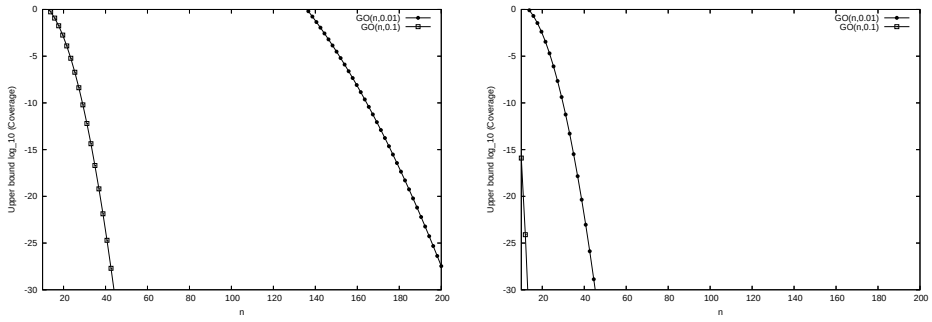

FIG. 7.2. Coverage general process omission failure model: Upper bound on $\log_{10}\left(\text{Cov}(GO_{n/2})\right)$ over n with $p = 0.01$ and $p = 0.1$, for $m = 2$ (left) and $m = 20$ (right).

Analysis of at least one process with f non-moving timely links ($\mathbf{TL_f}$). We now turn our attention to the "non-moving" link timing models [3–5] used for solving Ω and consensus in an (almost) asynchronous system of n processes with up to f process crash failures and eventually reliable links. The weakest model among those (denoted TL_f here) assumes that (eventually) there is at least one process p with at least f timely outgoing links in each of its broadcasts. Note that those f links are fixed throughout the (suffix of the) execution.

We note that the model TL_f is actually too weak for solving consensus within *bounded* time: As shown in [43, 44], considerably more timely links are required to solve consensus within a bounded number m of (timely) rounds. We incorporate the analysis of TL_f here, however, since it provides sort of an "upper bound" w.r.t. coverage: Any model that, in addition to TL_f, requires additional timely links must have an even lower coverage in our benchmarking scenario. Bear in mind, however, that the number of rounds m should be considered large here.

With p representing the probability that a link is non-timely in a round here, the probability $P_{n-1}(f) = Cov(TL_f)$ that some process has at least f timely links during m rounds is at most $n\binom{n-1}{f}(1-p)^{fm}$: We have n processes, and there are $\binom{n-1}{f}$ different subsets of f processes among the $n-1$ neighbors of a process; $(1-p)^{fm}$

gives the probability that the links to a fixed set of f neighbors are correct during all m rounds. Hence,

$$\mathrm{Cov}(TL_f) \le n \cdot \binom{n-1}{f}(1-p)^{fm}. \tag{7.7}$$

As in the analysis of GO_f, this is only a [quite conservative] upper bound for $P_{n-1}(f)$, however, since the involved events are not independent.

For $f = \lambda(n-1) = \lambda n'$, $0 < \lambda < 1$, where we employed the abbreviation

$$n' = n - 1,$$

used throughout this section, we obtain

$$\begin{aligned}\mathrm{Cov}(TL_{\lambda n'}) &\le (n'+1) \cdot \binom{n'}{\lambda n'}(1-p)^{m\lambda n'} \\ &\propto \sqrt{\frac{n'}{2\pi\lambda(1-\lambda)}} \cdot \left((1-\lambda)^{-(1-\lambda)} \cdot \lambda^{-\lambda}\right)^{n'} \cdot (1-p)^{m\lambda n'}. \end{aligned} \tag{7.8}$$

Choosing $\lambda = 1/2$, i.e., $f = (n-1)/2 = n'/2$ to facilitate comparison with our other results, we obtain

$$\mathrm{Cov}(TL_{n'/2}) \propto \sqrt{\frac{2n'}{\pi}}\left(4(1-p)^m\right)^{n'/2}. \tag{7.9}$$

The above bound goes to 0 for $n' \to \infty$ if $4(1-p)^m < 1$, i.e., when the number of rounds satisfies

$$m > -\frac{\log 4}{\log(1-p)} > \frac{\log 4}{p}, \tag{7.10}$$

according to the series expansion $\log(1-x) = -\sum_{k\ge1} x^k/k$, valid for $|x| < 1$. For smaller values of m, (7.9) increases exponentially. Since $\mathrm{Cov}(TL_{n'/2}) = P_{n'}(n'/2)$ is a probability and hence ≤ 1, however, the question arises whether the range of m where $\mathrm{Cov}(TL_{n'/2}) \to 0$ could be extended by a refined analysis.

Some advanced results on the distribution of the maximum degree of nodes in a geometric random graph [57] can be used for this purpose: The sought probability $P_{n-1}(f)$ is just the probability that the maximum degree Δ of the nodes in a random graph with n nodes (where an edge exists, independently of the other edges, with some fixed probability $0 < q < 1$) satisfies $\Delta \ge f$. More specifically, we have to consider the random graph with n nodes, corresponding to our processes, where an edge (x, y) exists if there is no link failure on the link $x \to y$ during m rounds. Clearly, the probability of the latter event is $q = (1-p)^m$. Note that [57] actually deals with undirected graphs. Considering the undirected random graph RG corresponding to an execution, instead of its directed counterpart $\overline{RG}$, provides an upper bound: Since every directed edge in $\overline{RG}$ is also present in RG, it follows that $\mathbf{P}\{\Delta_{RG} \ge f\} \ge \mathbf{P}\{\Delta_{\overline{RG}} \ge f\}$.

THEOREM 10 (Maximum degree in geometric random graphs [57]). *Given a geometric random graph with n nodes and edge probability q, the maximum degree Δ is strongly concentrated, in the sense that almost always*

$$\left|\Delta - qn - \sqrt{2q(1-q)n\log n} + \log\log n\sqrt{\frac{q(1-q)n}{8\log n}}\right| \le \log\log\sqrt{\frac{n}{\log n}}. \tag{7.11}$$

Moreover, the tail satisfies $\mathbf{P}\{\Delta < qn + b\sqrt{nq(1-q)}\} = \big(c(b) + o(1)\big)^n$ *for* $n \to \infty$, *where* $c(b) < 1$ *is the root of a certain equation;* $c(0) = 0.6102$ *and* $c(b)$ *is independent of* q.

According to our exposition above, we must set $q := (1-p)^m$, $n := n'+1$ and $f = n'/2$. Now, if $n'/2 > nq = (n'+1)(1-p)^m$ with $n'/2-(n'-1)(1-p)^m \geq C \cdot n'$ for some constant $C > 0$, then the area $\Delta \geq n'/2$ is to the right of the area of concentration $\big[nq - O(\sqrt{n \log n}), nq + O(\sqrt{n \log n})\big]$ of Δ given in (7.11) if n is sufficiently large. As a consequence, $\mathrm{Cov}(TL_{n'/2}) = \mathbf{P}\{\Delta_{\overline{RG}} \geq n'/2\} \leq \mathbf{P}\{\Delta_{RG} \geq n'/2\}$ actually goes to 0 for $n \to \infty$ if $(1-p)^m < 1/2$, i.e., when $m > \frac{\log 2}{p} > \frac{\log 2}{p}$, cp. (7.10). For smaller values of m, $\mathrm{Cov}(TL_{n'/2}) \to 1$ for $n \to \infty$ (with a fast transition phase in between, as usual in random graphs).

The following Figure 7.3 provides some numerical values in a system with $p = 0.1$ for $m = 20$ rounds. It reveals that the coverage of $n'/2$ fixed timely links in our benchmarking scenario is poor if m is above its critical value (7.10) w.r.t. p (which is the case for $m = 20$ and $p = 0.1$).

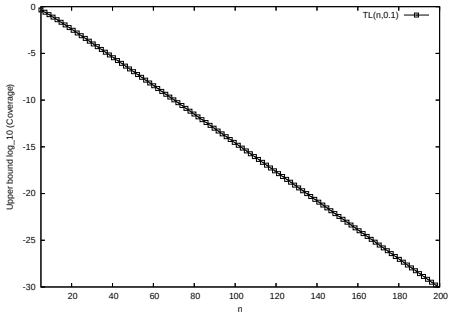

FIG. 7.3. Coverage of non-moving timely links: Upper bound on $\log_{10}\big(\mathrm{Cov}(TL_{n'/2})\big)$ over n with $p = 0.1$ for $m = 20$. The number of rounds $m = 20$ is above its critical value (7.10) here.

Analysis of n − 2 moving link failures per round ($\mathbf{ML_{n-2}}$). Classic moving link failure models, in particular [58], admit only $O(n)$ link failures per round. Let $ML_{\lambda n}$ be the model that admits at most λn link failures per round, for some real constant $\lambda > 0$. In [58], it was shown that consensus possibility demands at most $\lambda n = n - 2$ link failures per round, hence $\lambda = (n-2)/n$ here.

We start our derivations with a simple bound on the tail of the binomial distribution taken from Feller's book [27][7], which will also be required in the analysis of the coverage of our model MLO_{f_ℓ}. Consider the binomial distribution $B(\overline{n}, \overline{p})$, where $\overline{p}$ is the "success" probability, and let

$$p_{\overline{n}}(f_\ell) = \sum_{l=0}^{f_\ell} \binom{\overline{n}}{l} \overline{p}^l (1-\overline{p})^{\overline{n}-l} \tag{7.12}$$

be the probability of at most f_ℓ "successes", and $q_{\overline{n}}(f_\ell) = 1 - p_{\overline{n}}(f_\ell)$ be the probability

[7]This method gives a better bound than Chernoff's, i.e., $q_n(f_\ell) \leq \min_{z \geq 1} B(z; n, p)/z^{f_\ell}$, where $B(z; n, p) = (pz + 1 - p)^n$ is the generating function of the binomial distribution.

of more than f_ℓ "successes". Clearly,

$$q_{\overline{n}}(f_\ell) = \sum_{l=f_\ell+1}^{\overline{n}} \binom{\overline{n}}{l} \overline{p}^l (1-\overline{p})^{\overline{n}-l}. \tag{7.13}$$

The following Lemma 7 gives an upper bound for this quantity.

LEMMA 7 (Upper bound for binomial tail [27]). *For any $0 \le \overline{p} \le 1$, $\overline{n} \ge 1$ and $f_\ell + 1 > \overline{np}$, we have*

$$q_{\overline{n}}(f_\ell) \le \frac{(1-\overline{p})(f_\ell+1)}{f_\ell+1-\overline{np}} \cdot \binom{\overline{n}}{f_\ell+1} \overline{p}^{f_\ell+1} (1-\overline{p})^{\overline{n}-f_\ell-1}. \tag{7.14}$$

Proof. Following the argument [27, p. 151, eq. (3.4)], let $b(k;n,p) = \binom{n}{k} p^k (1-p)^{n-k}$ and note that $q_{\overline{n}}(f_\ell) = \sum_{k=0}^{\infty} b(k+f_\ell+1; \overline{n}, \overline{p})$. Using the straightforward identity $\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$, one obtains for any $k \ge 1$

$$\begin{aligned}
b(k+f_\ell+1;n,p) &= \frac{(n-k-f_\ell-1+1)p}{(k+f_\ell+1)(1-p)} \cdot b(k-1+f_\ell+1;n,p) \\
&= \Big(1 - \frac{k+f_\ell+1-(n+1)p}{(k+f_\ell+1)(1-p)}\Big) \cdot b(k-1+f_\ell+1;n,p) \\
&= \prod_{j=1}^{k} \Big(1 - \frac{j+f_\ell+1-(n+1)p}{(j+f_\ell+1)(1-p)}\Big) \cdot b(f_\ell+1;n,p) \\
&= \prod_{j=1}^{k} \Big(1 - \frac{1-\frac{(n+1)p}{j+f_\ell+1}}{1-p}\Big) \cdot b(f_\ell+1;n,p).
\end{aligned}$$

Since it is easily checked that, for any $j \ge 1$,

$$1 - \frac{1-\frac{(n+1)p}{j+f_\ell+1}}{1-p} \le 1 - \frac{1-\frac{np}{f_\ell+1}}{1-p}$$

it follows that

$$b(k+f_\ell+1;n,p) \le \Big(1 - \frac{1-\frac{np}{f_\ell+1}}{1-p}\Big)^k \cdot b(f_\ell+1;n,p),$$

which holds even for $k \ge 0$. Consequently,

$$\begin{aligned}
q_{\overline{n}}(f_\ell) &= \sum_{k=0}^{\infty} b(k+f_\ell+1;\overline{n},\overline{p}) \le b(f_\ell+1;\overline{n},\overline{p}) \sum_{k=0}^{\infty} \Big(1 - \frac{1-\frac{\overline{np}}{f_\ell+1}}{1-\overline{p}}\Big)^k \\
&\le \frac{1-\overline{p}}{1-\frac{\overline{np}}{f_\ell+1}} \cdot \binom{\overline{n}}{f_\ell+1} \overline{p}^{f_\ell+1} (1-\overline{p})^{\overline{n}-f_\ell-1}
\end{aligned}$$

as asserted in (7.14). ∎

In case of $ML_{\lambda n}$, we set $\overline{n} := n(n-1)$, $\overline{p} := 1-p$ and $f_\ell := n(n-1) - \lambda n - 1 = n(n-1-\lambda) - 1$ in Lemma 7, such that $q_{n(n-1)}\big(n(n-1-\lambda)-1\big)$ is the probability that at least $n(n-1-\lambda)$ non-faulty links (and hence at most λn faulty links) occur per round. It evaluates to

$$q_{n(n-1)}\big(n(n-1-\lambda)-1\big) \le \frac{pn(n-1-\lambda)}{n(n-1-\lambda) - n(n-1)(1-p)} \cdot \binom{n(n-1)}{n(n-1-\lambda)} (1-p)^{n(n-1-\lambda)} p^{\lambda n}.$$

By independence, the probability that at most λn link failures occur during m rounds is $\big[q_{n(n-1)}\big(n(n-1-\lambda)-1\big)\big]^m$ and thus

$$\text{(7.15)}\quad \mathrm{Cov}(ML_{\lambda n}) \le \left[\frac{p\cdot(n-1-\lambda)}{p\cdot(n-1)-\lambda}\cdot\binom{n(n-1)}{n(n-1-\lambda)}(1-p)^{n(n-1-\lambda)}p^{\lambda n}\right]^m$$

by Lemma 7. In order to determine the asymptotic value of $\mathrm{Cov}(ML_{\lambda n})$, we will need

$$\begin{aligned}
\Big(1-\frac{x}{n}\Big)^n &\sim e^{-x} \qquad \text{for any fixed } x \text{ and } n\to\infty,\\
\Big(1-\frac{x}{n-1}\Big)^{n(n-1)} &= e^{n(n-1)\log\left(1-\frac{x}{n-1}\right)}\\
&= e^{n(n-1)\cdot\left(-\frac{x}{n-1}-\frac{x^2}{2(n-1)^2}+O(x^3/n^3)\right)}\\
&= e^{-nx-\frac{nx^2}{2(n-1)}+O(x^3/n)}\\
&\sim e^{-nx-\frac{x^2}{2}} \qquad \text{for } |x|<1,
\end{aligned}$$

where we used the series expansion $\log(1-x) = -\sum_{k\ge1} x^k/k$. Applying Stirling's formula (7.3) again yields

$$\begin{aligned}
\binom{n(n-1)}{n(n-1-\lambda)} &\sim \frac{\left(\frac{n(n-1)}{e}\right)^{n(n-1)}}{\left(\frac{n(n-1-\lambda)}{e}\right)^{n(n-1-\lambda)}\left(\frac{\lambda n}{e}\right)^{\lambda n}}\cdot\frac{\sqrt{2\pi n(n-1)}}{\sqrt{2\pi n(n-1-\lambda)}\cdot\sqrt{2\pi\lambda n}}\\
&\sim \Big(\frac{n-1}{n-1-\lambda}\Big)^{n(n-1)}\cdot\Big(\frac{n-1-\lambda}{\lambda}\Big)^{\lambda n}\cdot\frac{1}{\sqrt{1-\frac{\lambda}{n-1}}\cdot\sqrt{2\pi\lambda n}}\\
&\sim \frac{1}{\left(1-\frac{\lambda}{n-1}\right)^{n(n-1)}}\cdot\Big(\frac{n}{\lambda}\Big)^{\lambda n}\cdot\Big(1-\frac{\lambda\cdot(1+\lambda)}{\lambda n}\Big)^{\lambda n}\cdot\frac{1}{\sqrt{2\pi\lambda n}}\\
&\sim e^{\lambda n+\lambda^2/2}\cdot\Big(\frac{n}{\lambda}\Big)^{\lambda n}\cdot e^{-\lambda(1+\lambda)}\cdot\frac{1}{\sqrt{2\pi\lambda n}}\\
&\sim \frac{e^{-\lambda(1+\lambda/2)}}{\sqrt{2\pi\lambda n}}\cdot\Big(\frac{en}{\lambda}\Big)^{\lambda n} \qquad \text{(7.16)}\\
&\sim e^{\lambda n\log n+\lambda n(1-\log\lambda)-\frac12\cdot\log n-\lambda(1+\lambda/2)-\frac12\cdot\log(2\pi\lambda)}. \qquad \text{(7.17)}
\end{aligned}$$

The dominant term in the exponent in (7.17) is clearly $\lambda n\log n > 0$. On the other hand,

$$(1-p)^{n(n-1-\lambda)}p^{\lambda n} = e^{n^2\log(1-p)-(1+\lambda)n\log(1-p)+(\lambda n)\log p},$$

which rapidly goes to 0 for $n\to\infty$ since the dominant term in the exponent is $\log(1-p)n^2 < 0$. Multiplying this with (7.16) according to (7.15) hence yields

$$\text{(7.18)}\qquad \mathrm{Cov}(ML_{\lambda n}) \propto \left[\frac{e^{-\lambda(1+\lambda/2)}}{\sqrt{2\pi\lambda n}}\cdot\Big(\frac{enp}{\lambda}\Big)^{\lambda n}(1-p)^{n(n-1-\lambda)}\right]^m,$$

which quickly goes to 0 for $n\to\infty$, for any λ and m.

For the special case $\lambda = (n-2)/n$, which implies at most $n-2$ link failures per round, we obtain for $n\to\infty$

$$\lambda \sim 1$$

$$\frac{e^{-\lambda(1+\lambda/2)}}{\sqrt{2\pi\lambda n}} \sim \frac{e^{-3/2}}{\sqrt{2\pi n}}$$
$$\Big(\frac{enp}{\lambda}\Big)^{\lambda n} = \Big(\frac{enp}{\frac{n-2}{n}}\Big)^{n-2} = \frac{(1-2/n)^2 \cdot (enp)^{n-2}}{(1-2/n)^n} \sim \frac{(enp)^{n-2}}{e^{-2}}$$
$$(1-p)^{n(n-1-\lambda)} = (1-p)^{n(n-1)-n+2} = \Big(\frac{1}{(1-p)^2}\Big)^{n-2}(1-p)^{n^2-2}.$$

Using this in expression (7.18) hence yields

$$\begin{aligned}\mathrm{Cov}(ML_{n-2}) &\sim \Big(\frac{e^{-3/2}}{e^{-2}\sqrt{2\pi n}}\Big)^m \Big(\frac{enp}{(1-p)^2}\Big)^{(n-2)m}(1-p)^{(n^2-2)m} \\ &\sim \Big(\sqrt{\frac{e}{2\pi n}}\Big)^m \Big(\frac{enp}{(1-p)^2}\Big)^{(n-2)m}(1-p)^{(n^2-2)m},\end{aligned}$$

which very quickly goes to 0 for $n \to \infty$. The following Figure 7.4 gives some numerical values, which reveal that the coverage of this model in our benchmarking scenario is very poor even for relatively small system sizes n.

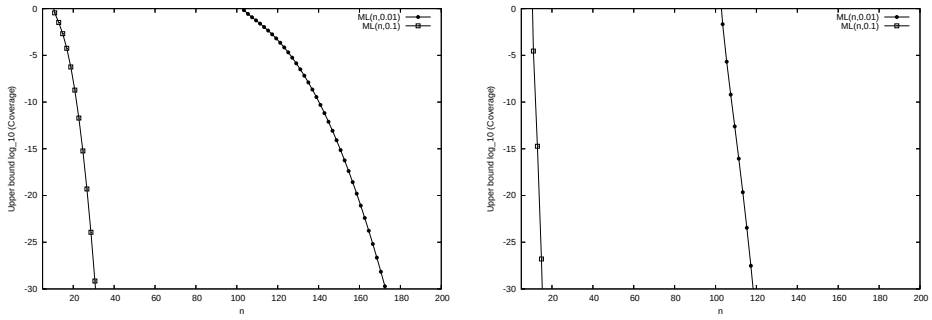

FIG. 7.4. Coverage of $O(n)$ moving link failures per round: Upper bound on $\log_{10}\big(\mathrm{Cov}(ML_{n-2})\big)$ over n with $p = 0.01$ and $p = 0.1$, for $m = 2$ (left) and $m = 20$ (right).

Analysis of $\mathbf{f_\ell}$ moving link failures per process and round ($\mathbf{MLO_{f_\ell}}$). Finally, we will show that the moving link omission failure model MLO_{f_ℓ} with $f_\ell^r = f_\ell^s = f_\ell$ introduced in Section 2.2 does not suffer from poor coverage: On the contrary, in accordance with [44], we will show that $\mathrm{Cov}(MLO_{n\lambda}) \to 1$ for $n \to \infty$, for any $p < 1/2$ and $\lambda > p$.

Recalling (7.12), we only have to set $\overline{n} := n-1$ and choose $\overline{p}$ to be our link failure probability p in this equation in order to obtain the probability $p_{n-1}(f_\ell)$ that at most f_ℓ outgoing links are faulty in the broadcast of a single process to its $n-1$ receivers in a round. Similarly, the probability $q_{n-1}(f_\ell) = 1 - p_{n-1}(f_\ell)$ that more than f_ℓ links are faulty in this event is given by (7.13), and Lemma 7 provides an upper bound for $q_{n-1}(f_\ell)$ for $f_\ell + 1 > (n-1)p$. Since we will eventually choose $f_\ell = (n-1)/2 - 1$ and $p < 1/2$, the latter condition is indeed satisfied. Note that there is a reasonably small upper bound for $q_{n-1}(f_\ell)$ also in case of small values $f_\ell + 1 \le (n-1)p$, see [62].

The probability that none of the n processes in the system experiences more than f_ℓ link failures on its outgoing links in a single round is $P_s = p_{n-1}(f_\ell)^n$, since the failures on the outgoing links of different processes are independent.

Obviously, Equation (7.12) for $p_{n-1}(f_\ell)$ also provides the probability that a single receiver process experiences at most f_ℓ link failures on its incoming links. As before, the probability that none of the n processes in the system experiences more than f_ℓ link failures on its incoming links in a round is $P_r = p_{n-1}(f_\ell)^n$.

The probability P_{sr} that none of the n processes in the system experiences more than f_ℓ link failures on its outgoing links and no more than f_ℓ link failures on its incoming links is not just the product of P_s and P_r, however, since they are not independent. However, $P_{sr} = P_{r|s}P_s$, where $P_{r|s}$ denotes the conditional probability that no process perceives more than f_ℓ link failures on its incoming links, conditioned on the fact that no process experiences more than f_ℓ link failures on its outgoing links. Since trivially $P_{r|s} \geq P_r$, we obtain $P_{sr} \geq P_s P_r = p_{n-1}(f_\ell)^{2n}$ and hence

$$\mathrm{Cov}(MLO_{f_\ell}) = P_{sr}^m \geq (1 - q_{n-1}(f_\ell))^{2nm}.$$

By the Bernoulli inequality $(1+\alpha)^n \geq 1 + n\alpha$ for any $\alpha > -1$, we obtain

$$(1 - q_{n-1}(f_\ell))^{2nm} \geq 1 - 2nmq_{n-1}(f_\ell), \tag{7.19}$$

which is valid for $q_{n-1}(f_\ell) < 1$; since the latter is a probability < 1, this condition is of course satisfied.

Consequently, using Lemma 7, $1 - \mathrm{Cov}(MLO_{f_\ell})$ can be upper bounded by

$$1 - \mathrm{Cov}(MLO_{f_\ell}) \leq \frac{2nm(f_\ell+1)(1-p)}{f_\ell + 1 - (n-1)p}\binom{n-1}{f_\ell+1}p^{f_\ell+1}(1-p)^{n-1-f_\ell-1}. \tag{7.20}$$

A very similar analysis as for $ML_{\lambda n}$ proves that $\mathrm{Cov}(MLO_{f_\ell})$ quickly approaches 1 as $n \to \infty$ for any $f_\ell + 1 = \lambda(n-1)$ with $\lambda > p$ and $p < 1/2$: Recalling Lemma 6, we immediately obtain

$$\binom{n}{\lambda n}p^{\lambda n}(1-p)^{n(1-\lambda)} \sim \Big(\frac{1-p}{1-\lambda}\Big)^{n(1-\lambda)}\Big(\frac{p}{\lambda}\Big)^{\lambda n}\frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}}.$$

Plugging in $n' := n-1$ and $f_\ell + 1 = (n-1)\lambda = n'\lambda$ according to (7.20) in the above equation provides

$$1 - \mathrm{Cov}(MLO_{n'\lambda-1}) \propto \frac{m(1-p)}{\lambda - p}\cdot\sqrt{\frac{2n'\lambda}{\pi(1-\lambda)}}\left[\Big(\frac{1-p}{1-\lambda}\Big)^{1-\lambda}\Big(\frac{p}{\lambda}\Big)^{\lambda}\right]^{n'}.$$

Since $x^\lambda \leq x$ for any $0 \leq \lambda \leq 1$, and $p < \lambda$,

$$\Big(\frac{1-p}{1-\lambda}\Big)^{1-\lambda}\Big(\frac{p}{\lambda}\Big)^{\lambda} = \frac{1-p}{1-\lambda}\Big(\frac{p(1-\lambda)}{\lambda(1-p)}\Big)^{\lambda} \leq \frac{1-p}{1-\lambda}\cdot\frac{p(1-\lambda)}{\lambda(1-p)} = \frac{p}{\lambda} < 1,$$

so $\mathrm{Cov}(MLO_{(n-1)\lambda-1})$ indeed quickly approaches 1 as $n \to \infty$. In the special case $\lambda = 1/2 > p$, where $f_\ell + 1 = (n-1)/2 = n'/2$ with $n = n'+1 \sim n'$ for $n \to \infty$, we obtain

$$1 - \mathrm{Cov}(MLO_{n'/2-1}) \propto \frac{m(1-p)}{1/2-p}\cdot\sqrt{\frac{2n'}{\pi}}\cdot\big(4p(1-p)\big)^{n'/2} \tag{7.21}$$

which rapidly goes to 0 for $n \to \infty$, for any $p < 1/2$ and any $m = O(n^k)$, k arbitrary but fixed. Figure 7.5 confirms that our link failure model[8] indeed scales well with

[8] As well as the moving timely link model for Ω and consensus of [39,40], which can be analyzed in a similar way.

n and gives excellent coverage for any reasonable choice of the parameters. Note carefully that those figures, in sharp contrast to all previous ones, show an upper bound on $1 - \mathrm{Cov}(MLO_{n'/2})$, i.e., the difference to ideal coverage.

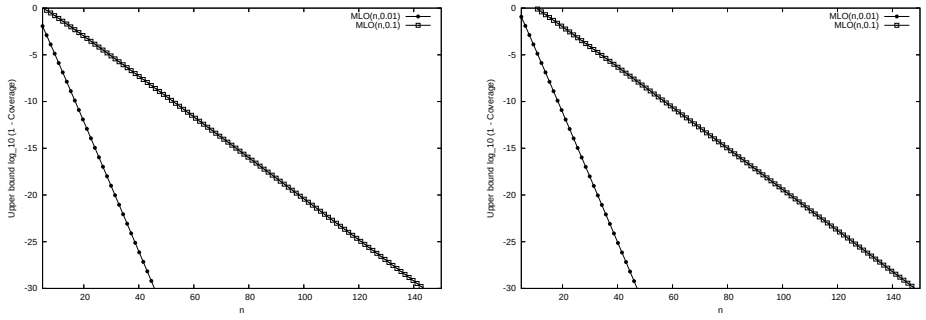

FIG. 7.5. Coverage of f_ℓ moving link failures per round per process: Upper bound on $\log_{10}\big(1 - \mathrm{Cov}(MLO_{n'/2})\big)$ over n with $p = 0.01$ and $p = 0.1$, for $m = 2$ (left) and $m = 20$ (right).

Finally, Figure 7.6 compares our upper bounds on $\mathrm{Cov}(GO_{n/2})$, $\mathrm{Cov}(TL_{n'/2})$ and $\mathrm{Cov}(ML_{n-2})$ in a system of $n = 30$ processes, for $m = 2$ and $m = 20$ rounds, under varying link failure rates p; Figure 7.7 does the same for $n = 60$. The numerical results reveal that all those models provide poor coverage in our benchmarking scenario, in particular under substantial link failure rates. Note that this is also true for $TL_{n'/2}$, unless the number of rounds m is not below the critical value (7.10) w.r.t. p.

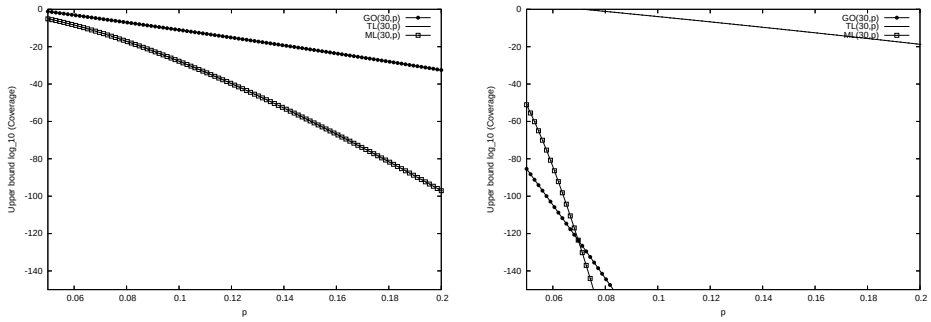

FIG. 7.6. Upper bound on $\mathrm{Cov}(GO_{n/2})$, $\mathrm{Cov}(TL_{n'/2})$ and $\mathrm{Cov}(ML_{n-1})$ over p for a system of $n = 30$ processes, for $m = 2$ (left) and $m = 20$ (right) rounds.

By contrast, Figures 7.8 and 7.9 provide numerical results for our upper bound on $1 - \mathrm{Cov}(MLO_{n'/2})$, under the same parameter values for n and m as in Figures 7.6 and 7.7. They reveal a high coverage also under substantial link failure rates, as well as a remarkably low dependence on the number m of rounds. They finally justify our claim that MLO is the only model that performs well in our benchmarking scenario.

8. Conclusions. We provided a complete theoretical treatment of the impossibility of deterministic synchronous consensus under a novel link failure model, which grants every process a certain maximum number of send and receive link failures per round. Link failures may both be omissive and arbitrary and can hit messages to/from different processes in every round. Using novel instances of "easy impossibility" and

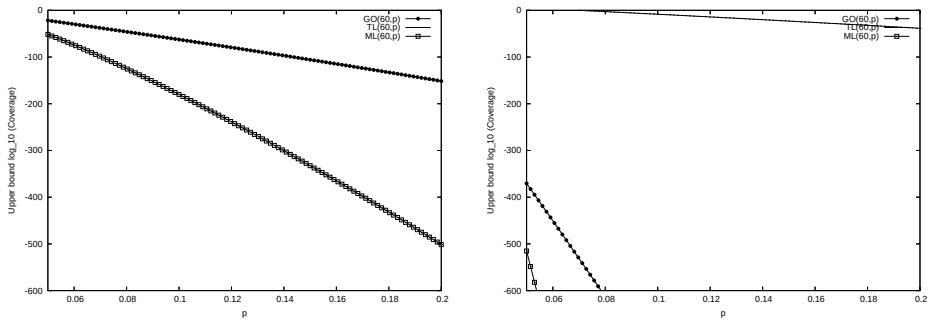

FIG. 7.7. Upper bound on $\mathrm{Cov}(GO_{n/2})$, $\mathrm{Cov}(TL_{n'/2})$ and $\mathrm{Cov}(ML_{n-1})$ over p for a system of $n = 60$ processes, for $m = 2$ (left) and $m = 20$ (right) rounds.

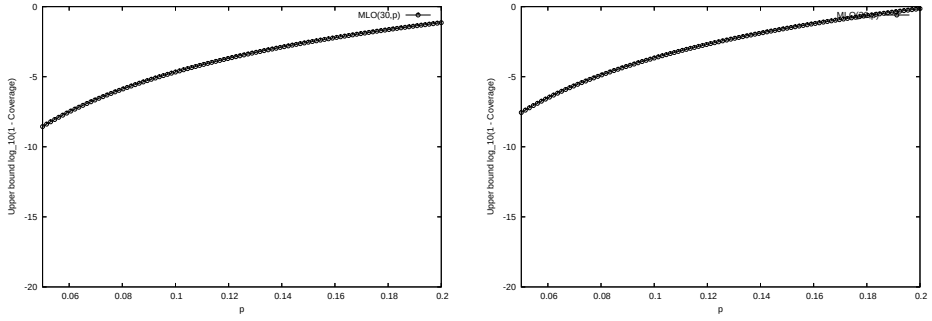

FIG. 7.8. Upper bound on $\log_{10}\big(1 - \mathrm{Cov}(MLO_{n'/2})\big)$ over p for a system of $n = 30$ processes, for $m = 2$ (left) and $m = 20$ (right) rounds.

bivalency proofs, we provided related lower bounds for the number of processes and rounds as well. Most of them are matched by existing consensus algorithms and hence tight. An analysis of the assumption coverage in a simple probabilistic setting revealed that our model is the only one with a coverage that approaches 1 (rather than 0) for large n.

Part of our current/future theoretical research in this area is devoted to consensus lower bounds under our fully-fledged hybrid failure model, which captures both process and link failures simultaneously. We analyzed several algorithms under this model and found that the respective numbers of processes required just add up. This suggested that tolerating link failures and process failures is more or less orthogonal. In [11], however, it was shown that this is not true in general. Generalized lower bounds are hence required for reasoning about optimal algorithms here.

Acknowledgments. We are grateful to Michael Drmota for providing us with Theorem 10 from [57], and to the anonymous reviewers for their thorough and very helpful comments.

REFERENCES

[1] YEHUDA AFEK, HAGIT ATTIYA, ALAN FEKETE, MICHAEL FISCHER, NANCY LYNCH, YISHAY MANSOUR, DAI-WEI WANG, AND LENORE ZUCK, *Reliable communication over unreliable channels*, Journal of the ACM (JACM), 41 (1994), pp. 1267–1297.

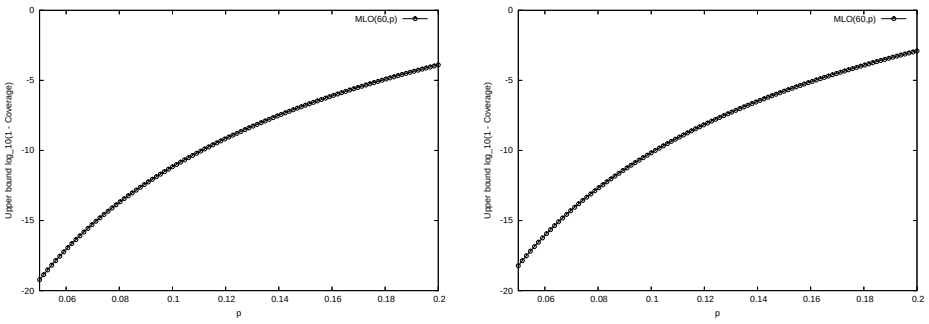

FIG. 7.9. Upper bound on $\log_{10}\big(1 - \mathrm{Cov}(MLO_{n'/2})\big)$ over p for a system of $n = 60$ processes, for $m = 2$ (left) and $m = 20$ (right) rounds.

[2] MARCOS KAWAZOE AGUILERA, WEI CHEN, AND SAM TOUEG, *Failure detection and consensus in the crash-recovery model*, Distributed Computing, 13 (2000), pp. 99–125.

[3] MARCOS KAWAZOE AGUILERA, CAROLE DELPORTE-GALLET, HUGUES FAUCONNIER, AND SAM TOUEG, *Stable leader election*, in DISC '01: Proceedings of the 15th International Conference on Distributed Computing, Springer-Verlag, 2001, pp. 108–122.

[4] MARCOS K. AGUILERA, CAROLE DELPORTE-GALLET, HUGUES FAUCONNIER, AND SAM TOUEG, *On implementing Omega with weak reliability and synchrony assumptions*, in Proceeding of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC'03), New York, NY, USA, 2003, ACM Press, pp. 306–314.

[5] MARCOS KAWAZOE AGUILERA, CAROLE DELPORTE-GALLET, HUGUES FAUCONNIER, AND SAM TOUEG, *Communication-efficient leader election and consensus with limited link synchrony*, in Proceedings of the 23th ACM Symposium on Principles of Distributed Computing (PODC'04), St. John's, Newfoundland, Canada, 2004, ACM Press, pp. 328–337.

[6] MARCOS KAWAZOE AGUILERA AND SAM TOUEG, *A simple bivalency proof that t-resilient consensus requires t+1 rounds*, Inf. Process. Lett., 71 (1999), pp. 155–158.

[7] EMMANUELLE ANCEAUME, ANTONIO FERNÁNDEZ, ACHOUR MOSTÉFAOUI, GIL NEIGER, AND MICHEL RAYNAL, *A necessary and sufficient condition for transforming limited accuracy failure detectors*, J. Comput. Syst. Sci., 68 (2004), pp. 123–133.

[8] HAGIT ATTIYA AND JENNIFER WELCH, *Distributed Computing*, McGraw-Hill, 1998.

[9] ANINDYA BASU, BERNADETTE CHARRON-BOST, AND SAM TOUEG, *Crash failures vs. crash + link failures*, in Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, United States, 1996, ACM Press, p. 246.

[10] PIOTR BERMAN, JUAN A. GARAY, AND KENNETH J. PERRY, *Asymptotically optimal distributed consensus*. http://www.bell-labs.com/user/garay/#distributed-pub, 1992. (A combination of results from ICALP'89, FOCS'89, and WDAG'91).

[11] MARTIN BIELY, *An optimal Byzantine agreement algorithm with arbitrary node and link failures*, in Proc. 15th Annual IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'03), Marina Del Rey, California, USA, Nov. 3–5, 2003, pp. 146–151.

[12] ———, *Towards an optimal algorithm for hybrid Byzantine agreement*, Tech. Report 183/1-130, Department of Automation, Technische Universität Wien, April 2003. (Extended version of [11]).

[13] MARTIN BIELY, BERNADETTE CHARRON-BOST, ANTOINE GAILLARD, MARTIN HUTLE, ANDRÉ SCHIPER, AND JOSEF WIDDER, *Tolerating corrupted communication*, in Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC'07), Portland, OR, USA, Aug. 2007, pp. 244–253.

[14] MARTIN BIELY AND ULRICH SCHMID, *Message-efficient consensus in presence of hybrid node and link faults*, Tech. Report 183/1-116, Department of Automation, Technische Universität Wien, August 2001.

[15] GABRIEL BRACHA AND SAM TOUEG, *Resilient consensus protocols*, in Proceedings of the 2nd Symposium on the Principles of Distributed Computing (PODC'83), Montreal, Canada, 1983, pp. 12–26.

[16] TUSHAR DEEPAK CHANDRA AND SAM TOUEG, *Unreliable failure detectors for reliable distributed systems*, Journal of the ACM, 43 (1996), pp. 225–267.

[17] Bernadette Charron-Bost and André Schiper, *Uniform consensus is harder than consensus.*, J. Algorithms, 51 (2004), pp. 15–37.

[18] Bernadette Charron-Bost and André Schiper, *The heard-of model: Unifying all benign failures*, Tech. Report LSR-REPORT-2006-004, EPFL, 2006.

[19] ———, *Improving fast paxos: being optimistic with no overhead*, in 12th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2006), IEEE Computer Society, 2006, pp. 287–295.

[20] Flaviu Cristian, Houtan Aghili, Ray Strong, and Danny Dolev, *Atomic broadcast: From simple message diffusion to byzantine agreement*, in Proceedings 15th Int. Conf. on Fault-Tolerant Computing (FTCS-15), Ann Arbor, Michigan, USA, 1985, pp. 200–206.

[21] Stefan Dobrev, *Computing input multiplicity in anonymous synchronous networks with dynamic faults*, in Proceedings of the 26th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'00), London, UK, 2000, Springer-Verlag, pp. 137–148.

[22] Stefan Dobrev and Imrich Vrto, *Optimal broadcasting in hypercubes with dynamic faults.*, Inf. Process. Lett., 71 (1999), pp. 81–85.

[23] Danny Dolev, *The Byzantine generals strike again*, Journal of Algorithms, 3 (1982), pp. 14–30.

[24] Danny Dolev, Roy Friedman, Idit Keidar, and Dahlia Malkhi, *Failure detectors in omission failure environments*, in Proc. 16th ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, 1997, p. 286.

[25] Partha Dutta and Rachid Guerraoui, *The inherent price of indulgence*, in Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing, Monterey, California, 2002, ACM Press, pp. 88–97.

[26] Partha Dutta, Rachid Guerraoui, and Idit Keidar, *The overhead of consensus failure recovery*, Distributed Computing, 19 (2007), pp. 373–386.

[27] William Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, John Wiley & Sons, New York, 3rd ed., 1968.

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

[29] Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson, *Impossibility of distributed consensus with one faulty process*, Journal of the ACM, 32 (1985), pp. 374–382.

[30] Eli Gafni, *Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony*, in Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, Puerto Vallarta, Mexico, 1998, ACM Press, pp. 143–152.

[31] Juan A. Garay, *Reaching (and maintaining) agreement in the presence of mobile faults*, in 8th International Workshop on Distributed Algorithms, vol. 857 of Lecture Notes in Computer Science, Springer, Sept. 1994, pp. 253–264.

[32] Li Gong, Patrick Lincoln, and John Rushby, *Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults*, in Proceedings Dependable Computing for Critical Applications-5, Champaign, IL, Sept. 1995, pp. 139–157.

[33] Jim N. Gray, *Notes on data base operating systems*, in Operating Systems: An Advanced Course, G. Seegmüller R. Bayer, R.M. Graham, ed., vol. 60 of Lecture Notes in Computer Science, Springer, New York, 1978, ch. 3.F, p. 465.

[34] Günther Gridling, *An algorithm for three-process consensus under restricted link failures*, Tech. Report 183/1-123, Department of Automation, Technische Universität Wien, Oct. 2002.

[35] Rachid Guerraoui, *Indulgent algorithms (preliminary version)*, in Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, Portland, Oregon, United States, 2000, ACM Press, pp. 289–297.

[36] Rachid Guerraoui, Rui Oliveira, and Andre Schiper, *Stubborn communication channels*, Tech. Report 98-278, Dèpartment d'Informatique, Ècole Polytechnique Fédérale de Lausanne, 1998.

[37] Vassos Hadzilacos, *Connectivity requirements for Byzantine agreement under restricted types of failures*, Distributed Computing, 2 (1987), pp. 95–103.

[38] Vassos Hadzilacos and Sam Toueg, *Fault-tolerant broadcasts and related problems*, in Distributed Systems, Sape Mullender, ed., Addison-Wesley, 2nd ed., 1993, ch. 5, pp. 97–145.

[39] Martin Hutle, Dahlia Malkhi, Ulrich Schmid, and Lidong Zhou, *Brief announcement: Chasing the weakest system model for implementing omega and consensus*, in Proceedings Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems (formerly Symposium on Self-stabilizing Systems) (SSS 2006), LNCS, Dallas, USA, Nov. 2006, Springer Verlag, pp. 576–577.

[40] ———, *Chasing the weakest system model for implementing omega and consensus*, IEEE Trans-

actions on Dependable and Secure Computing, (2008). (to appear).

[41] Keidar and Rajsbaum, *On the cost of fault-tolerant consensus when there are no faults (preliminary version)*, SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory), 32 (2001).

[42] Idit Keidar and Sergio Rajsbaum, *A simple proof of the uniform consensus synchronous lower bound*, Information Processing Letters, 85 (2003), pp. 47–52.

[43] Idit Keidar and Alex Shraer, *Timeliness, failure detectors, and consensus performance*, in Proceedings of the twenty-fifth annual ACM SIGACT-SIGOPS symposium on Principles of Distributed Computing (PODC'06), New York, NY, USA, 2006, ACM Press.

[44] Idit Keidar and Alexander Shraer, *How to choose a timing model*, in Proceedings 37th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), June 2007, pp. 389–398.

[45] Leslie Lamport, *Lower bounds on consensus.* (unpublished manuscript, available via *http://lamport.org*, 2000).

[46] ———, *The part-time parliament*, ACM Transactions on Computer Systems, 16 (1998), pp. 133–169.

[47] Leslie Lamport, Robert Shostak, and Marshall Pease, *The Byzantine generals problem*, ACM Transactions on Programming Languages and Systems, 4 (1982), pp. 382–401.

[48] Patrick Lincoln and John Rushby, *A formally verified algorithm for interactive consistency under a hybrid fault model*, in Proceedings Fault Tolerant Computing Symposium 23, Toulouse, France, June 1993, pp. 402–411.

[49] Zsuzsanna Lipták and Arfst Nickelsen, *Broadcasting in complete networks with dynamic edge faults*, in 4th International Conference on Principles of Distributed Systems (OPODIS'00), Dec. 20–22, 2000.

[50] Nancy Lynch, *Distributed Algorithms*, Morgan Kaufman Publishers, Inc., San Francisco, USA, 1996.

[51] Dahlia Malkhi, Florin Oprea, and Lidong Zhou, Ω *meets paxos: Leader election and stability without eventual timely links*, in Proceedings of the 19th Symposium on Distributed Computing (DISC'05), vol. 3724 of LNCS, Cracow, Poland, 2005, Springer Verlag, pp. 199–213.

[52] Yoram Moses and Sergio Rajsbaum, *A layered analysis of consensus*, SIAM J. Comput., 31 (2002), pp. 989–1021.

[53] Achour Mostéfaoui and Michel Raynal, *Solving consensus using chandra-toueg's unreliable failure detectors: A general quorum-based approach*, in Distributed Computing: 13th International Symposium (DISC'99), P. Jayanti, ed., vol. 1693 of Lecture Notes in Computer Science, Bratislava, Slovak Republic, Sept. 1999, Springer-Verlag GmbH, pp. 49–63.

[54] Kenneth J. Perry and Sam Toueg, *Distributed agreement in the presence of processor and communication faults*, IEEE Transactions on Software Engineering, SE-12 (1986), pp. 477–482.

[55] Shlomit S. Pinter and Ilka Shinahr, *Distributed agreement in the presence of communication and process failures*, in Proceedings of the 14th IEEE Convention of Electrical & Electronics Engineers in Israel, IEEE, Mar. 1985.

[56] Rüdiger Reischuk, *A new solution for the Byzantine generals problem*, Information and Control, 64 (1985), pp. 23–42.

[57] Oliver Riordan and Alex Selby, *The maximum degree of a random graph*, Comb. Probab. Comput., 9 (2000), pp. 549–572.

[58] Nicola Santoro and Peter Widmayer, *Time is not a healer*, in Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89), LNCS 349, Paderborn, Germany, Feb. 1989, Springer-Verlag, pp. 304–313.

[59] ———, *Distributed function evaluation in the presence of transmission faults.*, in SIGAL International Symposium on Algorithms, 1990, pp. 358–367.

[60] ———, *Agreement in synchronous networks with ubiquitous faults*, Theor. Comput. Sci., 384 (2007), pp. 232–249.

[61] Hasan M. Sayeed, Marwan Abu-Amara, and Hosame Abu-Amara, *Optimal asynchronous agreement and leader election algorithm for complete networks with Byzantine faulty links*, Distributed Computing, 9 (1995), pp. 147–156.

[62] Ulrich Schmid, *Failure model coverage under transient link failures*, Research Report 2/2004, Technische Universität Wien, Institut für Technische Informatik, Treitlstraße 3, A-1040 Vienna, Austria, 2004. (submitted).

[63] Ulrich Schmid and Christof Fetzer, *Randomized asynchronous consensus with imperfect communications*, Tech. Report 183/1-120, Department of Automation, Technische Universität Wien, January 2002. (Extended version of [64]).

[64] ———, *Randomized asynchronous consensus with imperfect communications*, in 22nd Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, Oct. 6–8, 2003, pp. 361–370.

[65] Ulrich Schmid and Bettina Weiss, *Synchronous Byzantine agreement under hybrid process and link failures*, Tech. Report 183/1-124, Department of Automation, Technische Universität Wien, Nov. 2002. (replaces TR 183/1-110).

[66] Ulrich Schmid, Bettina Weiss, and John Rushby, *Formally verified byzantine agreement in presence of link faults*, in 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2-5, 2002, pp. 608–616.

[67] Hin-Sing Siu, Yeh-Hao Chin, and Wei-Pang Yang, *Byzantine agreement in the presence of mixed faults on processors and links*, IEEE Transactions on Parallel and Distributed Systems, 9 (1998), pp. 335–345.

[68] T.K. Srikanth and Sam Toueg, *Simulating authenticated broadcasts to derive simple fault-tolerant algorithms*, Distributed Computing, 2 (1987), pp. 80–94.

[69] Gerard Tel, *Introduction to Distributed Algorithms*, Cambridge University Press, 1994.

[70] George Varghese and Nancy A. Lynch, *A tradeoff between safety and liveness for randomized coordinated attack protocols*, in Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 1992, pp. 241–250.

[71] Bettina Weiss and Ulrich Schmid, *Consensus with written messages under link faults*, in 20th Symposium on Reliable Distributed Systems (SRDS'01), New Orleans, LA, USA, Oct. 28–31, 2001, pp. 194–197.

[72] Josef Widder and Ulrich Schmid, *Booting clock synchronization in partially synchronous systems with hybrid process and link failures*, Distributed Computing, 20 (2007), pp. 115–140.