Causality and Epistemic Reasoning in Byzantine Multi-Agent Systems

Roman Kuznets*
Embedded Computing Systems
TU Wien
Vienna, Austria
roman@logic.at

Laurent Prosperi
ENS Paris-Saclay
Cachan, France
laurent.prosperi@ens-cachan.fr

Ulrich Schmid
Embedded Computing Systems
TU Wien
Vienna, Austria
s@ecs.tuwien.ac.at

Krisztina Fruzsa†
Embedded Computing Systems
TU Wien
Vienna, Austria
krisztina.fruzsa@tuwien.ac.at

Causality is an important concept both for proving impossibility results and for synthesizing efficient protocols in distributed computing. For asynchronous agents communicating over unreliable channels, causality is well studied and understood. This understanding, however, relies heavily on the assumption that agents themselves are correct and reliable. We provide the first epistemic analysis of causality in the presence of byzantine agents, i.e., agents that can deviate from their protocol and, thus, cannot be relied upon. Using our new framework for epistemic reasoning in fault-tolerant multi-agent systems, we determine the byzantine analog of the causal cone and describe a communication structure, which we call a multipede, necessary for verifying preconditions for actions in this setting.

1 Introduction

Reasoning about knowledge has been a valuable tool for analyzing distributed systems for decades [5, 9], and has provided a number of fundamental insights. As crisply formulated by Moses [17] in the form of the *Knowledge of Preconditions Principle*, a precondition for action must be known in order to be actionable. In a distributed environment, where agents only communicate by exchanging messages, an agent can only learn about events happening to other agents via messages (or sometimes the lack thereof [8]).

In *asynchronous systems*, where the absence of communication is indistinguishable from delayed communication, agents can only rely on messages they receive. Lamport's seminal definition of the *happened-before* relation [14] establishes the causal structure for asynchronous agents in the agent–time graph describing a run of a system. This structure is often referred to as a *causal cone*, whereby causal links are either time transitions from past to future for one agent or messages from one agent to another. As demonstrated by Chandy and Misra [2], the behavior of an asynchronous agent can only be affected by events from within its causal cone.

The standard way of showing that an agent does not know of an event is to modify a given run by removing the event in question in such a way that the agent cannot detect the change. By Hintikka's definition of knowledge [11], the agent thinks it possible that the event has not occurred and, hence, does not know of the event to have occurred. Chandy and Misra's result shows that in order for agent i to learn of an event happening to another agent j, there must exist a chain of successfully delivered messages

*Supported by the Austrian Science Fund (FWF) projects RiSE/SHiNE (S11405) and ADynNet (P28182).
†PhD student in the FWF doctoral program LogiCS (W1255).

leading from the moment of agent j observing the event to some past or present state of agent i. This observation remains valid in asynchronous distributed systems where messages could be lost and/or where agents may stop operating (i.e., crash) [4, 10, 19].

In *synchronous systems*, if message delays are upper-bounded, agents can also learn from the absence of communication (communication-by-time). As shown in [1], Lamport's happened-before relation must then be augmented by causal links indicating no communication within the message delay upper bound to also capture causality induced via communication-by-time, leading to the so-called *syncausality relation*. Its utility has been demonstrated using the *ordered response problem*, where agents must perform a sequence of actions in a given order: both the necessary and sufficient knowledge and a necessary and sufficient communication structure (called a *centipede*) have been determined in [1]. It is important to note, however, that syncausality works only in fault-free distributed systems with reliable communication. Although it has recently been shown in [8] that *silent choirs* are a way to extend it to distributed systems where agents may crash, the idea does not generalize to less benign faults.

Unfortunately, all the above ways of capturing causality and the resulting simplicity of determining the causal cone completely break down if agents may be *byzantine* faulty [15]. Byzantine faulty agents may behave arbitrarily, in particular, need not adhere to their protocol and may, hence, send arbitrary messages. It is common to limit the maximum number of agents that ever act byzantine in a distributed system by some number f, which is typically much smaller than the total number n of agents. Prompted by the ever growing number of faulty hardware and software having real-world negative, sometimes life-critical, consequences, capturing causality and providing ways for determining the causal cone in byzantine fault-tolerant distributed systems is both an important and scientifically challenging task. To the best of our knowledge, this challenge has not been addressed in the literature before.[1]

In a nutshell, for $f > 0$, the problem of capturing causality becomes complicated by the fact that a simple causal chain of messages is no longer sufficient: a single byzantine agent in the chain could manufacture "evidence" for anything, both false negatives and false positives. And indeed, obvious generalizations of message chains do not work. For example, it is a folklore result that, in the case of direct communication, at least $f+1$ confirmations are necessary because f of them could be false. When information is transmitted along arbitrary, possibly branching and intersecting chains of messages, the situation is even more complex and defies simplistic direct analysis. In particular, as shown by the counterexample in [16, Fig. 1], one cannot rely on Menger's Theorem [3] for separating nodes in the two-dimensional agent–time graph.

Major contributions: In this paper, we generalize the causality structure of asynchronous distributed systems described above to multi-agent systems involving byzantine faulty agents. Relying on our novel byzantine runs-and-systems framework [12] (described in full detail in [13]), we utilize some generic epistemic analysis results for determining the shape of the byzantine analog of Lamport's causal cone. Since *knowledge* of an event is too strong a precondition in the presence of byzantine agents, it has to be relaxed to something more akin to *belief* relative to correctness [18], for which we coined the term *hope*. We show that hope can only be achieved via a causal message chain that passes solely through correct agents (more precisely, through agents still correct while sending the respective messages). While the result looks natural enough, its formal proof is quite involved technically and paints an instructive picture of how byzantine agents can affect the information flow. We also establish a necessary condition for detecting an event, and a corresponding communication structure (called a *multipede*), which is severely complicated by the fact that the reliable causal cones of indistinguishable runs may be different.

Paper organization: In Sect. 2, we succinctly introduce the features of our byzantine runs-and-

[1]Despite having "Byzantine" in the title, [4, 10] only address benign faults (crashes, send/receive omissions of messages).

systems framework [13] and state some generic theorems and lemmas needed for proving the results of the paper. In Sect. 3, we describe the mechanism of run modifications, which are used to remove events an agent should not know about from a run, without the agent noticing. Our characterization of the byzantine causal cone is provided in Sect. 4, the necessary conditions for establishing hope for an occurrence of an event and the underlying multipede structure can be found in Sect. 5. Some conclusions in Sect. 6 round-off the paper.

2 Runs-and-Systems Framework for Byzantine Agents

First, we describe the modifications of the runs-and-systems framework [5] necessary to account for byzantine behavior. To prevent wasting space on multiple definition environments, we give the following series of formal definitions as ordinary text marking defined objects by italics; consult [13] for the same definitions in fully spelled-out format. As a further space-saving measure, instead of repeating every time "actions and/or events," we use *haps*[2] as a general term referring to either actions or events.

The goal of all these definitions is to formally describe a system where *asynchronous agents* $1,\dots,n$ perform actions according to their protocols, observe events, and exchange messages within an environment represented as a special agent ε. Unlike the environment, agents only have limited local information, in particular, being asynchronous, do not have access to the global clock. No assumptions apart from liveness are made about the communication. Messages can be lost, arbitrarily delayed, and/or delivered in the wrong order. This part of the system is a fairly standard asynchronous system with unreliable communication. The novelty is that the environment may additionally cause at most f agents to become faulty *in arbitrary ways*. A faulty agent can perform any of its actions irrespective of its protocol and observe events that did not happen, e.g., receive unsent or corrupted messages. It can also have false memories about actions it has performed. At the same time, much like the global clock, such malfunctions are not directly visible to an agent, especially when it mistakenly thinks it acted correctly.

We fix a finite set $\mathscr{A} = \{1,\dots,n\}$ of agents. Agent $i \in \mathscr{A}$ can perform *actions* $a \in \mathit{Actions}_i$, e.g., send *messages*, and witness *events* $e \in \mathit{Events}_i$ such as message delivery. We denote $\mathit{Haps}_i := \mathit{Actions}_i \sqcup \mathit{Events}_i$. The action of sending a copy numbered k of a message $\mu \in \mathit{Msgs}$ to an agent $j \in \mathscr{A}$ is denoted $\mathit{send}(j, \mu_k)$, whereas a receipt of such a message from $i \in \mathscr{A}$ is recorded locally as $\mathit{recv}(i, \mu)$.[3]

Agent i records actions from $\mathit{Actions}_i$ and observes events from Events_i without dividing them into correct and faulty. The environment ε, on the contrary, always knows if the agent acted correctly or was forced into byzantine behavior. Hence, the syntactic representations of each hap for agents (local view) and for the environment (global view) must differ, with the latter containing more information. In particular, the global view syntactically distinguishes correct haps from their byzantine counterparts. While there is no way for an agent to distinguish a real event from its byzantine duplicate, it can analyze its recorded actions and compare them with its protocol. Sometimes, this information might be sufficient for the agent to detect its own malfunctions.

All of $\mathit{Actions} := \bigcup_{i\in\mathscr{A}} \mathit{Actions}_i$, $\mathit{Events} := \bigcup_{i\in\mathscr{A}} \mathit{Events}_i$, and $\mathit{Haps} := \mathit{Actions} \sqcup \mathit{Events}$ represent the local view of haps. All haps taking place after a *timestamp* $t \in \mathbb{T} := \mathbb{N}$ and no later than $t+1$ are grouped into a *round* denoted $t + 1/2$ and are treated as happening simultaneously. To model *asynchronous agents*, we exclude these system timestamps from the local format of *Haps*. At the same time, the environment ε incorporates the current timestamp t into the global format of every correct action $a \in \mathit{Actions}_i$, as initi-

[2]Cf. "Till I know 'tis done, Howe'er my haps, my joys were ne'er begun." W. Shakespeare, *Hamlet*, Act IV, Scene 3.

[3]Thus, it is possible to send several copies of the same message in the same round. If one or more of such copies are received in the same round, however, the recipient does not know which copy it has received, nor that there have been multiple copies.

ated by agent i in the local format, via a one-to-one function $global(i,t,a)$. Timestamps are especially crucial for proper message processing with $global(i,t,send(j,\mu_k)) := gsend(i,j,\mu,id(i,j,\mu,k,t))$ for some one-to-one function $id\colon \mathscr{A} \times \mathscr{A} \times Msgs \times \mathbb{N} \times \mathbb{T} \to \mathbb{N}$ that assigns each sent message a unique *global message identifier* (GMI). We chose not to model agent-to-agent channels explicitly. With all messages effectively sent through one system-wide channel, these GMIs are needed to ensure the causality of message delivery, i.e., that only sent messages can be delivered correctly. The sets $\overline{GActions_i} := \{global(i,t,a) \mid t \in \mathbb{T}, a \in Actions_i\}$ of all possible correct actions for each agent in global format are pairwise disjoint due to the injectivity of $global$. We set $\overline{GActions} := \bigsqcup_{i\in\mathscr{A}} \overline{GActions_i}$.

Unlike correct actions, correct events witnessed by agents are generated by the environment ε and, hence, can be assumed to be produced already in the global format $\overline{GEvents_i}$. We define $\overline{GEvents} := \bigsqcup_{i\in\mathscr{A}} \overline{GEvents_i}$ assuming them to be pairwise disjoint, and $\overline{GHaps} = \overline{GEvents} \sqcup \overline{GActions}$. We do not consider the possibility of the environment violating its protocol, which is meant to model the fundamental physical laws of the system. Thus, all events that can happen are considered correct. A byzantine event is, thus, a subjective notion. It is an event that was perceived by an agent despite not taking place. In other words, each correct event $E \in \overline{GEvents_i}$ has a faulty counterpart $fake(i,E)$, and agent i cannot distinguish the two. An important type of correct global events of agent j is the delivery $grecv(j,i,\mu,id) \in \overline{GEvents}_j$ of message μ with GMI $id \in \mathbb{N}$ sent by agent i. Note that the GMI, which is used in by the global format to ensure causality, must be removed before the delivery is recorded by the agent in the local format because GMIs contain the time of sending, which should not be accessible to agents. To strip this information before updating local histories, we employ a function $local\colon \overline{GHaps} \to Haps$ converting *correct* haps from the global into the local format in such a way that for actions *local* reverses *global*, i.e., $local\big(global(i,t,a)\big) := a$. For message deliveries, $local\big(grecv(j,i,\mu,id)\big) := recv(i,\mu)$, i.e., agent j only knows that it received message μ from agent i. It is, thus, possible for two distinct correct global events, e.g., $grecv(j,i,\mu,id)$ and $grecv(j,i,\mu,id')$, representing the delivery of different copies of the same message μ, possibly sent by i at different times, to be recorded by j the same way, as $recv(i,\mu)$.

Therefore, correct actions are initiated by agents in the local format and translated into the global format by the environment. Correct and byzantine events are initiated by the environment in the global format and translated into the local format before being recorded by agents.[4] We will now turn our attention to byzantine actions.

While a faulty event is purely an error of perception, actions can be faulty in another way: they can violate the protocol. The crucial question is: who should be responsible for such violations? With agents' actions governed by their protocols while everything else is up to the environment, it seems that errors, especially unintended errors, should be the environment's responsibility. A malfunctioning agent tries to follow its protocol but fails for reasons outside of its control, i.e., due to environment interference. A malicious agent tries to hide its true intentions from other agents by pretending to follow its expected protocol and, thus, can also be modeled via environment interference. Thus, we model faulty actions as byzantine events of the form $fake(i, A \mapsto A')$ where $A, A' \in \overline{GActions_i} \sqcup \{\mathbf{noop}\}$ for a special *non-action* **noop** in global format. Here A is the action (or, in case of **noop**, inaction) performed, while A' represents the action (inaction) perceived instead by the agent. More precisely, the agent either records $a' = local(A') \in Events_i$ if $A' \in \overline{GEvents_i}$ or has no record of this byzantine action if $A' = \mathbf{noop}$. The byzantine inaction $fail(i) := fake(i, \mathbf{noop} \mapsto \mathbf{noop})$ is used to make agent i faulty without performing any actions and without leaving a record in i's local history. The set of all i's byzantine events, corresponding to both faulty events and faulty actions, is denoted $BEvents_i$, with $BEvents := \bigsqcup_{i\in\mathscr{A}} BEvents_i$.

To prevent our asynchronous agents from inferring the global clock by counting rounds, we make

[4]This has already been described for correct events. A byzantine event is recorded the same way as its correct counterpart.

waking up for a round contingent on the environment issuing a special *system event* $go(i)$ for the agent in question. Agent i's local view of the system immediately after round $t + 1/2$, referred to as (*process-time* or *agent-time*) *node* $(i, t+1)$, is recorded in i's *local state* $r_i(t+1)$, also called i's *local history*. Nodes $(i, 0)$ correspond to *initial local states* $r_i(0) \in \Sigma_i$, with $\mathscr{G}(0) := \prod_{i \in \mathscr{A}} \Sigma_i$. If a round contains neither $go(i)$ nor any event to be recorded in i's local history, then the said history $r_i(t+1) = r_i(t)$ remains unchanged, denying the agent the knowledge of the round just passed. Otherwise, $r_i(t+1) = X : r_i(t)$, for $X \subseteq Haps_i$, the set of all actions and events perceived by i in round $t + 1/2$, where $:$ stands for concatenation. The exact definition will be given via the $update_i$ function, to be described shortly. Thus, the local history $r_i(t)$ is a list of all haps as perceived by i in rounds it was *active* in. The set of all local states of i is $\mathscr{L}_i$.

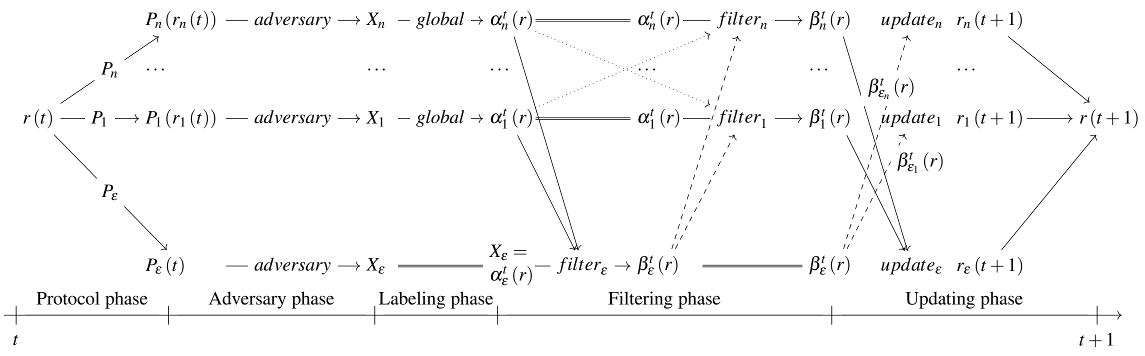
While not necessary for asynchronous agents, for future backwards compatibility, we add more system events for each agent, to serve as faulty counterparts to $go(i)$. Commands $sleep\,(i)$ and $hibernate\,(i)$ signify a failure to activate the agent's protocol and differ in that the former enforces waking up the agent (and thus recording time) notwithstanding. These commands will be used, e.g., for synchronous systems. None of the *system events* $SysEvents_i := \{go(i), sleep\,(i), hibernate\,(i)\}$ is directly detectable by agents.

To summarize, $GEvents_i := \overline{GEvents_i} \sqcup BEvents_i \sqcup SysEvents_i$ with $GEvents := \bigsqcup_{i \in \mathscr{A}} GEvents_i$ and $GHaps := GEvents \sqcup \overline{GActions}$. Throughout the paper, horizontal bars signify phenomena that are correct. Note that the absence of this bar means the absence of a claim of correctness. It does not necessarily imply a fault. Later, this would also apply to formulas, e.g., $\overline{occurred_i}(e)$ demands a correct occurrence of an event e whereas $occurred_i(e)$ is satisfied by either correct or faulty occurrence.

We now turn to the description of runs and protocols for our byzantine-prone asynchronous agents. A *run* r is a sequence of *global states* $r(t) = (r_\varepsilon(t), r_1(t), \dots, r_n(t))$ of the whole system consisting of the *state* $r_\varepsilon(t)$ *of the environment* and local states $r_i(t)$ of every agent. We already discussed the composition of local histories. Similarly, the *environment's history* $r_\varepsilon(t)$ is a list of all haps that happened, this time faithfully recorded in the global format. Accordingly, $r_\varepsilon(t+1) = X : r_\varepsilon(t)$ for the set $X \subseteq GHaps$ of all haps from round $t + 1/2$. The set of all global states is denoted $\mathscr{G}$.

What happens in each round is determined by protocols P_i of agents, protocol P_ε of the environment, and chance, the latter implemented as the *adversary* part of the environment. Agent i's *protocol* $P_i \colon \mathscr{L}_i \to \wp(\wp(Actions_i)) \setminus \{\varnothing\}$ provides a range $P_i(r_i(t))$ of sets of actions based on i's current local state $r_i(t)$, with the view of achieving some collective goal. Recall that the global timestamp t is *not* part of $r_i(t)$. The control of all events—correct, byzantine, and system—lies with the environment ε via its protocol $P_\varepsilon \colon \mathbb{T} \to \wp(\wp(GEvents)) \setminus \{\varnothing\}$, which *can* depend on a timestamp $t \in \mathbb{T}$ but *not* on the current state. The environment's protocol is thus kept impartial by denying it an agenda based on the global history so far. Other parts of the environment must, however, have access to the global history, in particular, to ensure causality. Thus, the environment's protocol provides a range $P_\varepsilon(t)$ of sets of events. Protocols P_i and P_ε are non-deterministic and always provide at least one option. The choice among the options (if more than one) is arbitrarily made by the already mentioned *adversary* part of the environment. It is also required that all events from $P_\varepsilon(t)$ be mutually compatible at time t. These *t-coherency* conditions are: (a) no more than one system event $go(i)$, $sleep\,(i)$, and $hibernate\,(i)$ per agent i at a time; (b) a correct event perceived as e by agent i is never accompanied by a byzantine event that i would also perceive as e, i.e., an agent cannot be mistaken about witnessing an event that *did* happen; (c) the GMI of a byzantine sent message is the same as if a copy of the same message were sent correctly in the same round. Note that the prohibition (b) does not extend to correct actions.

Both the global run $r \colon \mathbb{T} \to \mathscr{G}$ and its local parts $r_i \colon \mathbb{T} \to \mathscr{L}_i$ provide a sequence of snapshots of the system and local states respectively. Given the *joint protocol* $P := (P_1, \dots, P_n)$ and the environment's protocol P_ε, we focus on *$\tau_{f,P_\varepsilon,P}$-transitional runs* r that result from following these protocols and are built according to a *transition relation* $\tau_{f,P_\varepsilon,P} \subseteq \mathscr{G} \times \mathscr{G}$ for asynchronous agents at most $f \geq 0$ of which may



$P_n(r_n(t))$ — adversary → X_n – global → $\alpha_n^t(r)$
$\alpha_n^t(r)$ — $filter_n$ → $\beta_n^t(r)$
$update_n$ $r_n(t+1)$
$r(t)$ — P_1 → $P_1(r_1(t))$ — adversary → X_1 – global → $\alpha_1^t(r)$
$\alpha_1^t(r)$ — $filter_1$ → $\beta_1^t(r)$
$update_1$ $r_1(t+1)$ → $r(t+1)$
P_n
P_ε
$\beta_{\varepsilon_n}^t(r)$
$\beta_{\varepsilon_1}^t(r)$
$P_\varepsilon(t)$ — adversary → X_ε
$X_\varepsilon = \alpha_\varepsilon^t(r)$ — $filter_\varepsilon$ → $\beta_\varepsilon^t(r)$
$\beta_\varepsilon^t(r)$ $update_\varepsilon$ $r_\varepsilon(t+1)$
Protocol phase
Adversary phase
Labeling phase
Filtering phase
Updating phase
t
$t+1$

Figure 1: Details of round $t + ½$ of a $\tau_{f,P_\varepsilon,P}$-transitional run r.

become faulty in a given run. In this paper, we only deal with generic f, P_ε, and P. Hence, whenever safe, we write τ in place of $\tau_{f,P_\varepsilon,P}$. Each transitional run begins in some initial global state $r(0) \in \mathscr{G}(0)$ and progresses by ensuring that $r(t)\ \tau\ r(t+1)$, i.e., $(r(t), r(t+1)) \in \tau$, for each timestamp $t \in \mathbb{T}$. Given f, P_ε, and P, the transition relation $\tau_{f,P_\varepsilon,P}$ consisting of five consecutive phases is graphically represented in Figure 1 and described in detail below:

1. **Protocol phase.** A range $P_\varepsilon(t) \subseteq \wp(GEvents)$ of t-coherent sets of events is determined by the environment's protocol P_ε; for each $i \in \mathscr{A}$, a range $P_i(r_i(t)) \subseteq \wp(Actions_i)$ of sets of i's actions is determined by the agents' joint protocol P.
2. **Adversary phase.** The adversary non-deterministically picks a t-coherent set $X_\varepsilon \in P_\varepsilon(t)$ and a set $X_i \in P_i(r_i(t))$ for each $i \in \mathscr{A}$.
3. **Labeling phase.** Locally represented actions in X_i's are translated into the global format: $\alpha_i^t(r) := \{global(i,t,a) \mid a \in X_i\} \subseteq \overline{GActions_i}$. In particular, correct sends are supplied with GMIs.
4. **Filtering phase.** Functions $filter_\varepsilon$ and $filter_i$ for each $i \in \mathscr{A}$ remove all causally impossible attempted events from $\alpha_\varepsilon^t(r) := X_\varepsilon$ and actions from $\alpha_i^t(r)$.

   4.1. First, $filter_\varepsilon$ filters out causally impossible events based (a) on the current global state $r(t)$, which could not have been accounted for by the protocol P_ε, (b) on $\alpha_\varepsilon^t(r)$, and (c) on all $\alpha_i^t(r)$, not accessible for P_ε either. Specifically, two kinds of events are causally impossible for asynchronous agents with at most f byzantine failures and are removed by $filter_\varepsilon$ in two stages as follows (formal definitions can be found in the appendix, Definitions A.16–A.17; cf. also [13] for details):

   (1) in the 1st stage, all byzantine events are removed by $filter_\varepsilon^{\leq f}$ if they would have resulted in more than f faulty agents in total;

   (2) in the 2nd stage, correct receives without matching sends (either in the history $r(t)$ or in the current round) are removed by $filter_\varepsilon^B$.

   The resulting set of events to actually occur in round $t + ½$ is denoted

$$\beta_\varepsilon^t(r) := filter_\varepsilon\big(r(t),\quad \alpha_\varepsilon^t(r),\quad \alpha_1^t(r),\quad \dots,\quad \alpha_n^t(r)\big).$$

   4.2. After events are filtered, $filter_i$ for each agent i removes all i's actions iff $go(i) \notin \beta_\varepsilon^t(r)$. The resulting sets of actions to be actually performed by agents in round $t + ½$ are

$$\beta_i^t(r) := filter_i\big(\alpha_1^t(r),\quad \dots,\quad \alpha_n^t(r),\quad \beta_\varepsilon^t(r)\big).$$

   We have $\beta_i^t(r) \subseteq \alpha_i^t(r) \subseteq \overline{GActions_i}$ and $\beta_\varepsilon^t(r) \subseteq \alpha_\varepsilon^t(r) \subseteq GEvents$.

5. Updating phase. The resulting mutually causally consistent sets of events $\beta^t_\varepsilon(r)$ and of actions $\beta^t_i(r)$ are appended to the global history $r(t)$; for each $i \in \mathscr{A}$, all non-system events from

$$\beta^t_{\varepsilon_i}(r) := \beta^t_\varepsilon(r) \cap GEvents_i$$

as *perceived* by the agent and all correct actions $\beta^t_i(r)$ are appended *in the local form* to the local history $r_i(t)$, which may remain unchanged if no action or event triggers an update or be appended with the empty set if an update is triggered only by a system event $go(i)$ or $sleep(i)$:

$$r_\varepsilon(t+1) := update_\varepsilon\big(r_\varepsilon(t),\quad \beta^t_\varepsilon(r),\quad \beta^t_1(r),\quad \dots,\quad \beta^t_n(r)\big); \tag{1}$$

$$r_i(t+1) := update_i\big(r_i(t),\quad \beta^t_i(r),\quad \beta^t_\varepsilon(r)\big). \tag{2}$$

Formal definitions of $update_\varepsilon$ and $update_i$ are given in Def. A.19 in the appendix.

The protocols P and P_ε only affect phase 1, so we group the operations in the remaining phases 2–5 into a *transition template* τ_f that computes a transition relation $\tau_{f,P_\varepsilon,P}$ for any given P and P_ε. This transition template, primarily via the filtering functions, represents asynchronous agents with at most f faults. The template can be modified independently from the protocols to capture other distributed scenarios.

Liveness and similar properties that cannot be ensured on a round-by-round basis are enforced by restricting the allowable runs by *admissibility conditions* Ψ, which formally are subsets of the set R of all transitional runs. For example, since no goal can be achieved without allowing agents to act from time to time, it is standard to impose the *Fair Schedule* (*FS*) admissibility condition, which for byzantine agents states that an agent can only be delayed indefinitely through persistent faults:

$$FS := \{r \in R \mid (\forall i \in \mathscr{A})\,(\forall t \in \mathbb{T})\,(\exists t' \geq t)\,\beta^{t'}_\varepsilon(r) \cap SysEvents_i \neq \varnothing\}.$$

In scheduling terms, *FS* ensures that each agent be considered for using CPU time infinitely often. Denying any of these requests constitutes a failure, represented by a $sleep(i)$ or $hibernate(i)$ system event.

We now combine all these parts in the notions of context and agent-context:

Definition 1. A *context* $\gamma = (P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi)$ consists of an environment's protocol P_ε, a set of global initial states $\mathscr{G}(0)$, a transition template τ_f for $f \geq 0$, and an admissibility condition Ψ. For a joint protocol P, we call $\chi = (\gamma, P)$ an *agent-context.* A run $r\colon \mathbb{T} \to \mathscr{G}$ is called *weakly χ-consistent* if $r(0) \in \mathscr{G}(0)$ and the run is $\tau_{f,P_\varepsilon,P}$-transitional. A weakly χ-consistent run r is called *(strongly) χ-consistent* if $r \in \Psi$. The set of all χ-consistent runs is denoted $R^\chi \subseteq R$. Agent-context χ is called *non-excluding* if any finite prefix of a weakly χ-consistent run can be extended to a χ-consistent run.

We are also interested in narrower types of faults. Let $FEvents_i := BEvents_i \sqcup \{sleep(i), hibernate(i)\}$.

Definition 2. Environment's protocol P_ε makes an agent $i \in \mathscr{A}$:

1. *correctable* if $X \in P_\varepsilon(t)$ implies that $X \setminus FEvents_i \in P_\varepsilon(t)$;
2. *delayable* if $X \in P_\varepsilon(t)$ implies $X \setminus GEvents_i \in P_\varepsilon(t)$;
3. *error-prone* if $X \in P_\varepsilon(t)$ implies that, for any $Y \subseteq FEvents_i$, the set $Y \sqcup (X \setminus FEvents_i) \in P_\varepsilon(t)$ whenever it is t-coherent;
4. *gullible* if $X \in P_\varepsilon(t)$ implies that, for any $Y \subseteq FEvents_i$, the set $Y \sqcup (X \setminus GEvents_i) \in P_\varepsilon(t)$ whenever it is t-coherent;
5. *fully byzantine* if agent i is both error-prone and gullible.

In other words, correctable agents can always be made correct for the round by removing all their byzantine events; delayable agents can always be forced to skip a round completely (which does not make them faulty); error-prone (gullible) agents can exhibit any faults in addition to (in place of) correct events, thus, implying correctability (delayability); fully byzantine agents' faults are unrestricted. Common types of faults, e.g., crash or omission failures, can be obtained by restricting allowable sets Y in the definition of gullible agents.

Now that our byzantine version of the runs-and-systems framework is laid out, we define interpreted systems in this framework in the usual way, i.e., as special kinds of Kripke models for multi-agent distributed environments [5]. For an agent-context χ, we consider pairs $(r,t') \in R^\chi \times \mathbb{T}$ of a χ-consistent run r and timestamp t'. A *valuation function* $\pi\colon Prop \to \wp(R^\chi \times \mathbb{T})$ determines whether an atomic proposition from $Prop$ is true in run r at time t'. The determination is arbitrary except for a small set of *designated atomic propositions* whose truth value at (r,t') is fully determined. More specifically, for $i \in \mathscr{A}$, $o \in Haps_i$, and $t \in \mathbb{T}$ such that $t \leq t'$,

$correct_{(i,t)}$ is true at (r,t'), or *node* (i,t) *is correct in run* r, iff no faulty event happened to i by timestamp t, i.e., no event from $FEvents_i$ appears in the $r_\varepsilon(t)$ prefix of the $r_\varepsilon(t')$ part of $r(t')$;

$correct_i$ is true at (r,t') iff $correct_{(i,t')}$ is;

$fake_{(i,t)}(o)$ is true at (r,t') iff i has a *faulty* reason to believe that $o \in Haps_i$ occurred in round $t-½$, i.e., $\underline{o \in r_i(t)}$ because (at least in part) of some $O \in BEvents_i \cap \beta_\varepsilon^{t-1}(r)$;

$\overline{occurred}_{(i,t)}(o)$ is true at (r,t') iff i has a *correct* reason to believe $o \in Haps_i$ occurred in round $t-½$, i.e., $\underline{o \in r_i(t)}$ because (at least in part) of $O \in (\overline{GEvents_i} \cap \beta_\varepsilon^{t-1}(r)) \sqcup \beta_i^{t-1}(r)$;

$\overline{occurred}_i(o)$ is true at (r,t') iff at least one of $\overline{occurred}_{(i,m)}(o)$ for $1 \leq m \leq t'$ is;

$\overline{occurred}(o)$ is true at (r,t') iff at least one of $\overline{occurred}_i(o)$ for $i \in \mathscr{A}$ is;

$occurred_i(o)$ is true at (r,t') iff either $\overline{occurred}_i(o)$ is or at least one of $fake_{(i,m)}(o)$ for $1 \leq m \leq t'$ is.

An *interpreted system* is a pair $\mathscr{I} = (R^\chi,\pi)$. The epistemic language $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi$ where $p \in Prop$ and $i \in \mathscr{A}$ and derived Boolean connectives are defined in the usual way. Truth for these (*epistemic*) *formulas* is defined in the standard way, in particular, for a run $r \in R^\chi$, timestamp $t \in \mathbb{T}$, atomic proposition $p \in Prop$, agent $i \in \mathscr{A}$, and formula φ we have $(\mathscr{I},r,t) \models p$ iff $(r,t) \in \pi(p)$ and $(\mathscr{I},r,t) \models K_i\varphi$ iff $(\mathscr{I},r',t') \models \varphi$ for any $r' \in R^\chi$ and $t' \in \mathbb{T}$ such that $r_i(t) = r'_i(t')$. A formula φ is valid in $\mathscr{I}$, written $\mathscr{I} \models \varphi$, iff $(\mathscr{I},r,t) \models \varphi$ for all $r \in R^\chi$ and $t \in \mathbb{T}$.

Due to the t-coherency of all allowed protocols P_ε, an agent cannot be both right and wrong about any local event $e \in Events_i$, i.e., $\mathscr{I} \models \neg(\overline{occurred}_{(i,t)}(e) \wedge fake_{(i,t)}(e))$. Note that for actions this *can* happen.

Following the concept from [6] of global events that are local for an agent, we define conditions under which formulas can be treated as such local events. A formula φ is called *localized for i within an agent-context* χ iff $r_i(t) = r'_i(t')$ implies $(\mathscr{I},r,t) \models \varphi \iff (\mathscr{I},r',t') \models \varphi$ for any $\mathscr{I} = (R^\chi,\pi)$, runs $r,r' \in R^\chi$, and timestamps $t,t' \in \mathbb{T}$. By these definitions, we immediately obtain:

Lemma 3. *The following statements are valid for any formula φ localized for an agent $i \in \mathscr{A}$ within an agent-context χ and any interpreted system $\mathscr{I} = (R^\chi,\pi)$: $\mathscr{I} \models \varphi \leftrightarrow K_i\varphi$ and $\mathscr{I} \models \neg\varphi \leftrightarrow K_i\neg\varphi$.*

The knowledge of preconditions principle [17] postulates that in order to act on a precondition an agent must be able to infer it from its local state. Thus, Lemma 3 shows that formulas localized for i can *always* be used as preconditions. Our first observation is that the agent's *perceptions* of a run are one example of such epistemically acceptable (though not necessarily reliable) preconditions:

Lemma 4. *For any agent-context χ, agent $i \in \mathscr{A}$, and local hap $o \in Haps_i$, the formula $occurred_i(o)$ is localized for i within χ.*

It can be shown that correctness of these perceptions is not localized for i and, hence, cannot be the basis for actions. In fact, Theorem 12 will reveal that no agent can establish its own correctness.

3 Run modifications

We will now introduce the pivotal technique of *run modifications*, which are used to show an agent does not know φ by creating an indistinguishable run where φ is false.

Definition 5. A function $\rho : R^\chi \to \wp(\overline{GActions_i}) \times \wp(GEvents_i)$ is called an *i-intervention for an agent-context χ and agent $i \in \mathscr{A}$*. A *joint intervention* $B = (\rho_1, \dots, \rho_n)$ consists of i-interventions ρ_i for each agent $i \in \mathscr{A}$. An *adjustment* $[B_t; \dots; B_0]$ is a sequence of joint interventions $B_0 \dots, B_t$ to be performed at rounds from ½ to $t + ½$.

We consider an i-intervention $\rho(r) = (X, X_\varepsilon)$ applied to a round $t + ½$ of a given run r to be a meta-action by the system designer, intended to modify the results of this round for i in such a way that $\beta_i^t(r') = X$ and $\beta_{\varepsilon_i}^t(r') = \beta_\varepsilon^t(r') \cap GEvents_i = X_\varepsilon$ in the artificially constructed new run r'. For $\rho(r) = (X, X_\varepsilon)$, we denote $\mathfrak{a}\rho(r) := X$ and $\mathfrak{e}\rho(r) := X_\varepsilon$. Accordingly, a joint intervention $(\rho_1, \dots, \rho_n)$ prescribes actions $\beta_i^t(r') = \mathfrak{a}\rho_i(r)$ for each agent i and events $\beta_\varepsilon^t(r') = \bigsqcup_{i \in \mathscr{A}} \mathfrak{e}\rho_i(r)$ for the round in question. Thus, an adjustment $[B_t; \dots; B_0]$ fully determines actions and events in the initial $t+1$ rounds of run r':

Definition 6. Let $adj = [B_t; \dots; B_0]$ be an adjustment where $B_m = (\rho_1^m, \dots, \rho_n^m)$ for each $0 \leq m \leq t$ and each ρ_i^m is an i-intervention for an agent-context $\chi = ((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P)$. A run r' is *obtained from $r \in R^\chi$ by adjustment adj* iff for all $t' \leq t$, all $T' > t$, and all $i \in \mathscr{A}$,

(a) $r'(0) := r(0)$,

(b) $r'_i(t'+1) := update_i\left(r'_i(t'),\ \mathfrak{a}\rho_i^{t'}(r),\ \bigsqcup_{i \in \mathscr{A}} \mathfrak{e}\rho_i^{t'}(r)\right)$,

(c) $r'_\varepsilon(t'+1) := update_\varepsilon\left(r'_\varepsilon(t'),\ \bigsqcup_{i \in \mathscr{A}} \mathfrak{e}\rho_i^{t'}(r),\ \mathfrak{a}\rho_1^{t'}(r),\ \dots,\ \mathfrak{a}\rho_n^{t'}(r)\right)$,

(d) $r'(T')\ \tau_{f,P_\varepsilon,P}\ r'(T'+1)$.

$R(\tau_{f,P_\varepsilon,P}, r, adj)$ is the set of all runs obtained from r by *adj*.

Note that adjusted runs need not be a priori transitional, i.e., obey (d), for $t' \leq t$. Of course, we intend to use adjustments in such a way that r' *is* a transitional run. But it requires a separate proof. In order to improve the readability of these proofs, we allow ourselves (and already used) a small abuse of notation. The β-sets $\beta_\varepsilon^t(r')$ and $\beta_i^t(r')$ were initially defined only for transitional runs as the result of filtering. But they also represent the sets of events and i's actions respectively happening in round $t + ½$. This alternative definition is equivalent for transitional runs and, in addition, can be used for adjusted runs r'. This is what we mean whenever we write β-sets for runs obtained by adjustments.

In order to minimize an agent's knowledge in accordance with the structure of its (soon to be defined) reliable (or byzantine) causal cone, we will use several types of i-interventions that copy round $t + ½$ of the original run to various degrees: (a) *CFreeze* denies i all actions and events, (b) $FakeEcho_i^t$ reproduces all messages sent by i but in byzantine form, (c) $X\text{-}Focus_i^t$ (for an appropriately chosen set $X \subseteq \mathscr{A} \times \mathbb{T}$) faithfully reproduces all actions and as many events as causally possible.

Definition 7. For an agent-context χ, $i \in \mathscr{A}$, and $r \in R^\chi$, we define the following i-interventions:

$$CFreeze(r) := (\varnothing, \varnothing). \tag{3}$$

$$FakeEcho_i^t(r) := \Big(\varnothing, \quad \{fail(i)\} \ \sqcup\ \{fake(i, gsend(i,j,\mu,id) \mapsto \mathbf{noop}) \quad | \quad gsend(i,j,\mu,id) \in \beta_i^t(r) \quad \vee \quad (\exists A) fake(i, gsend(i,j,\mu,id) \mapsto A) \in \beta_\varepsilon^t(r)\}\Big). \tag{4}$$

$$X\text{-}Focus_i^t(r) := \Big(\beta_i^t(r), \qquad \beta_{\varepsilon_i}^t(r) \setminus \{grecv(i,j,\mu,id(j,i,\mu,k,m)) \quad | \quad (j,m) \notin X, k \in \mathbb{N}\}\Big). \tag{5}$$

4 The Reliable Causal Cone

Before giving formal definitions and proofs, we first explain the intuition behind our byzantine analog of Lamport's causal cone, and the particular adjustments used for constructing runs with identical reliable causal cones in the main Lemma 10 of this section.

In the absence of faults [2], the only way information, say, about an event e that happens to an agent j can reach another agent i, or more precisely, its node (i,t), is via a causal chain (time progression and delivered messages) originating from j after e happened and reaching i no later than timestamp t. The set of beginnings of causal chains, together with all causal links, is called the *causal cone* of (i,t). The standard way of demonstrating the necessity of such a chain for i to learn about e, when expressed in our terminology, is by using an adjustment that removes all events and actions outside the causal cone. Once an adjusted run with no haps outside the causal cone is shown to be transitional and the local state of i at timestamp t is shown to be the same as in the given run, it follows that i considers it possible that e did not happen and, hence, does not know that e happened. This well known proof is carried out in our framework in [12] (see also [13] for an extended version).

However, one subtle aspect of our formalization is also relevant for the byzantine case. We illustrate it using a minimal example. Suppose, in the given run, j_s sent exactly one message to j_r during round $m+½$ and it was correctly received by j_r in round $l+½$. At the same round, j_r itself sent its last ever message, and sent it to i. If this message to i arrived before t, then (j_r,l) is a node within the causal cone of (i,t). On the other hand, neither $(j_r,l+1)$ nor (j_s,m) are within the causal cone. Thus, the run adjustment discussed in the previous paragraph removes the action of sending the message from j_s to j_r, which happened outside the causal cone, and, hence, makes it causally impossible for j_r to receive it despite the fact that the receipt happened within, or more precisely, on the boundary of the causal cone. On the other hand, the message sent by j_r in the same round cannot be suppressed without i noticing. Thus, suppressing all haps on the boundary of the causal cone is not an option. These considerations necessitate the use of $X\text{-}Focus^l_j$ to remove such "ghosts" of messages instead of the exact copy of round $l+½$ of the given run. To obtain Chandy–Misra's result, one needs to set X to be the entire causal cone.[5]

We now explain the complications created by the presence of byzantine faults. Because byzantine agents can lie, the existence of a causal chain is no more sufficient for reliable delivery of information. Causal chains can now be *reliable*, i.e., involve only correct agents, or *unreliable*, whereby a byzantine agent can corrupt the transmitted information or even initiate the whole communication while pretending to be part of a longer chain. If several causal chains link a node (j,m) witnessing an event with (i,t), where the decision based on this event is to be made, then, intuitively, the information about the event can only be relied upon if at least one of these causal chains is reliable. In effect, all correct nodes, i.e., nodes (j,m) such that $(\mathscr{I},r,t) \models correct_{(j,m)}$, are divided into three categories: those without any causal chains to (i,t), i.e., nodes outside Lamport's causal cone, those with causal chains but only unreliable ones, and those with at least one reliable causal chain. There is, of course, the fourth category consisting of byzantine nodes, i.e., nodes (j,m) such that $(\mathscr{I},r,t) \not\models correct_{(j,m)}$. Since there is no way for nodes without reliable causal chains to make themselves heard, we call these nodes *silent masses* and apply to them the *CFreeze* intervention: since they cannot have an impact, they need not act. The nodes with at least one reliable causal chain to (i,t), which must be correct themselves, form the *reliable causal cone* and treated the same way as Lamport's causal cone in the fault-free case, except that the removal of "ghost" messages is more involved in this case. Finally, the remaining nodes are byzantine and form a *fault buffer* on the

[5]This treatment of the cone's boundary could be perceived as overly pedantic. But in our view this is preferable to being insufficiently precise.

way of reliable information. Their role is to pretend the run is the same independently of what the silent masses do. We will show that $\mathit{FakeEcho}^m_j$ suffices since only messages sent from the fault buffer matter.

Before stating our main Lemma 10, which constructs an adjusted run that leaves agent i at t in the same position while removing as many haps as possible, it should be noted that our analysis relies on *knowing which agents are byzantine* in the given run, which may easily change without affecting local histories. This assumption will be dropped in the following section.

First we define simple causal links among nodes as binary relations on $\mathscr{A} \times \mathbb{T}$ in infix notation:

Definition 8. For all $i \in \mathscr{A}$ and $t \in \mathbb{T}$, we have $(i,t) \to_l (i,t+1)$. Additionally, for a run r, we have $(i,m) \to^r_c (j,l)$ iff there are $\mu \in \mathit{Msgs}$ and $id \in \mathbb{N}$ such that $\mathit{grecv}(j,i,\mu,id) \in \beta^{l-1}_\varepsilon(r)$ and either $\mathit{gsend}(i,j,\mu,id) \in \beta^m_i(r)$ or $\mathit{fake}(i, \mathit{gsend}(i,j,\mu,id) \mapsto A) \in \beta^m_\varepsilon(r)$ for some $A \in \{\mathbf{noop}\} \sqcup \overline{\mathit{GActions}_i}$. *Causal r-links* $\to^r := \to_l \cup \to^r_c$ are either local or communication related. A *causal r-path* for a run r is a sequence $\xi = \langle \theta_0, \theta_1, \dots, \theta_k \rangle$, $k \geq 0$, of nodes connected by causal r-links, i.e., such that $\theta_l \to^r \theta_{l+1}$ for each $0 \leq l < k$. This causal r-path is called *reliable* iff node $(j_l, t_l + 1)$ is correct in r for each $\theta_l = (j_l, t_l)$ with $0 \leq l < k$ and, additionally, node $\theta_k = (j_k, t_k)$ is correct in r. We also write $\theta_0 \rightsquigarrow^r_\xi \theta_k$ to denote the fact that path ξ connects node θ_0 to θ_k in run r, or simply $\theta_0 \rightsquigarrow^r \theta_k$ to state that such a causal r-path exists.

Note that neither receives nor sends of messages forming a reliable causal r-path can be byzantine. The latter is guaranteed by the immediate future of nodes on the path being correct.

Definition 9. The *reliable causal cone* $\blacktriangleright^r_\theta$ *of node* θ *in run* r consists of all nodes $\zeta \in \mathscr{A} \times \mathbb{T}$ such that $\zeta \rightsquigarrow^r_\xi \theta$ for some reliable causal r-path ξ. The *fault buffer* $\rangle\!\rangle^r_\theta$ *of node* θ *in run* r consists of all nodes (j,m) with $m < t$ such that $(j,m) \rightsquigarrow^r \theta$ and $(j,m+1)$ is not correct. Abbreviating $\rangle\!\rangle\!\blacktriangleright^r_\theta := \blacktriangleright^r_\theta \sqcup \rangle\!\rangle^r_\theta$, the *silent masses of node* θ *in run* r are all the remaining nodes $\cdot\rangle^r_\theta := (\mathscr{A} \times \mathbb{T}) \setminus \rangle\!\rangle\!\blacktriangleright^r_\theta$.

Here the filling of the cone $\blacktriangleright$ signifies reliable communication, $\rangle\!\rangle$ represents a barrier for correct information, whereas $\cdot\rangle$ depicts correct information isolated from its destination. We can now state the main result of this section:

Lemma 10 (Cone-equivalent run construction). *For $f \in \mathbb{N}$, for a non-excluding agent-context $\chi = \big((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P\big)$ such that all agents are gullible, correctable, and delayable, for any $\tau_{f,P_\varepsilon,P}$-transitional run r, for a node $\theta = (i,t) \in \mathscr{A} \times \mathbb{T}$ correct in r, let adjustment $adj = [B_{t-1}; \dots; B_0]$ where $B_m = (\rho^m_1, \dots, \rho^m_n)$ for each $0 \leq m \leq t-1$ such that*

$$\rho^m_j := \begin{cases} \rangle\!\rangle\!\blacktriangleright^r_\theta\text{-}\mathit{Focus}^m_j & \text{if } (j,m) \in \blacktriangleright^r_\theta\,, \\ \mathit{FakeEcho}^m_j & \text{if } (j,m) \in \rangle\!\rangle^r_\theta, \\ \mathit{CFreeze} & \text{if } (j,m) \in \cdot\rangle^r_\theta. \end{cases} \tag{6}$$

Then each $r' \in R(\tau_{f,P_\varepsilon,P}, r, adj)$ satisfies the following properties:

(A) $(\forall (j,m) \in \blacktriangleright^r_\theta) \quad r'_j(m) = r_j(m)$*;*

(B) $(\forall m \leq t) \quad r'_i(m) = r_i(m)$*;*

(C) for any $m \leq t$, we have that $\beta^{m-1}_\varepsilon(r') \cap \mathit{FEvents}_j \neq \varnothing$ iff both $(j,m-1) \rightsquigarrow^r \theta$ and (j,m) is not correct in r;

(D) for any $m \leq t$, any node (j,m) correct in r is also correct in r';

(E) the number of agents byzantine by any $m \leq t$ in run r' is not greater than that in run r and is $\leq f$;

(F) r' is $\tau_{f,P_\varepsilon,P}$-transitional.

Proof sketch. The following properties follow from the definitions:

$$\blacktriangleright^r_\theta \cap \rangle\!\rangle^r_\theta = \varnothing, \qquad \theta \in \blacktriangleright^r_\theta, \tag{7}$$

$$(j,m) \in \blacktriangleright^r_\theta \quad \& \quad (k,m') \to^r (j,m) \quad \Longrightarrow \quad (k,m') \in \rangle\!\rangle\!\blacktriangleright^r_\theta. \tag{8}$$

Note that for $\beta_k = (j_k, m_k)$ with $k = 1,2$, we have $\beta_1 \to^r \beta_2$ implies $m_1 < m_2$ and $\beta_1 \rightsquigarrow^r \beta_2$ implies $m_1 \le m_2$. Thus, all parts of the lemma except for Statement (F) only concern $m \le t$, and even this last statement for $m > t$ is a trivial corollary of Def. 6(d). Thus, we focus on $m \le t$.

Statement (A) can be proved by induction on m using the following auxiliary lemma for the given transitional run r and the adjusted run r', which is also constructed using the standard update functions.

Lemma 11. *If* $r_j(m) = r'_j(m)$*, and* $\beta^m_j(r) = \mathfrak{a}\rho^m_j(r)$*, and* $\beta^m_{\varepsilon_j}(r) = \mathfrak{e}\rho^m_j(r)$*, then* $r_j(m+1) = r'_j(m+1)$.

Proof. This statement follows from (2) for the transitional run r, Def. 6(b) for the adjusted run r', and the fact that $update_j$ only depends on events of agent j, in particular, on the presence of $go(j)$ or $sleep(j)$ (see Def. A.19 in the appendix for details). □

The third condition of Lemma 11 is satisfied for $\rho^m_j(r) = \rangle\!\rangle\!\blacktriangleright^r_\theta\text{-}Focus^m_j$ within $\blacktriangleright^r_\theta$ by (8). Further, if $(j,m) \in \blacktriangleright^r_\theta$, then so are all (j,m') with $m' \le m$. In particular, $(i,m') \in \blacktriangleright^r_\theta$ for any $m' \le t$. Thus, Statement (B) follows from Statement (A) we have already proved.

Statement (C) is due to the fact that (a) $\rangle\!\rangle\!\blacktriangleright^r_\theta\text{-}Focus^m_j$ does not produce any new byzantine events relative to $\beta^m_{\varepsilon_j}(r)$, which contains none for $(j,m) \in \blacktriangleright^r_\theta$, (b) *CFreeze* never produces byzantine events, whereas (c) $FakeEcho^m_j$ always contains at least $fail(j) \in BEvents_j$. Statements (D) and (E) are direct corollaries of Statement (C).

The bulk of the proof concerns Statement (F), or, more precisely the transitionality up to timestamp t. For each $m < t$, we need sets $\alpha^m_\varepsilon(r') \in P_\varepsilon(m)$ and $\alpha^m_j(r') = \{global(j,m,a) \mid a \in X_j\}$ for some $X_j \in P_j(r'_j(m))$ for each $j \in \mathscr{A}$ such that for $\beta^m_\varepsilon(r') = \bigsqcup_{j \in \mathscr{A}} \mathfrak{e}\rho^m_j(r)$ and $\beta^m_j(r') = \mathfrak{a}\rho^m_j(r)$ for all $j \in \mathscr{A}$,

$$\beta^m_\varepsilon(r') = filter_\varepsilon\left(r'(m), \alpha^m_\varepsilon(r'), \alpha^m_1(r'), \dots, \alpha^m_n(r')\right), \tag{9}$$
$$\beta^m_j(r') = filter_j\left(\alpha^m_1(r'), \dots, \alpha^m_n(r'), \beta^m_\varepsilon(r')\right). \tag{10}$$

The construction of such α-sets and the proof of (9)–(10) for them is by induction on m. Note that $r'_\varepsilon(0) = r_\varepsilon(0)$ and $r'_j(0) = r_j(0)$ for all $j \in \mathscr{A}$ by Def. 6(a). We will show that it suffices to choose

$$\alpha^m_j(r') := \begin{cases} \alpha^m_j(r) & \text{if } (j,m) \in \blacktriangleright^r_\theta, \\ \{global(j,m,a) \mid a \in X_j\} \text{ for some } X_j \in P_j(r'_j(m)) & \text{otherwise,} \end{cases} \tag{11}$$

with the choice in the latter case possible by $P_j(r'_j(m)) \neq \varnothing$, and

$$\begin{aligned} \alpha^m_\varepsilon(r') := \Big(filter^{\le f}_\varepsilon\left(r(m), \alpha^m_\varepsilon(r), \alpha^m_1(r), \dots, \alpha^m_n(r)\right) \setminus \bigsqcup_{(l,m) \in \cdot\rangle^r_\theta \sqcup \rangle\!\rangle^r_\theta} GEvents_l\Big) \sqcup \{fail(l) \mid (l,m) \in \rangle\!\rangle^r_\theta\} \sqcup \\ \Big\{fake\left(l, gsend(l,j,\mu,id) \mapsto \mathbf{noop}\right) \;\Big|\; (l,m) \in \rangle\!\rangle^r_\theta \quad \& \\ \Big(gsend(l,j,\mu,id) \in \beta^m_l(r) \quad \vee \quad (\exists A) fake\left(l, gsend(l,j,\mu,id) \mapsto A\right) \in \beta^m_\varepsilon(r)\Big)\Big\}. \end{aligned} \tag{12}$$

Informally, according to (11), in r', we just repeat the choices made in r within the reliable causal cone and make arbitrary choices elsewhere. According to (12), events are chosen in a more complex way. First, mimicking the 1st-stage filtering in the given run r, the originally chosen $\alpha^m_\varepsilon(r) \in P_\varepsilon(m)$ is preventively purged of all byzantine events whenever they would have caused more than f agents to become faulty in r. Note that, in our transitional simulation of the adjusted run r', this is done prior to filtering (9) by exploiting the correctability of all agents. Secondly, for all agents l outside the reliable causal cone at the current timestamp m, i.e., with $(l,m) \in \cdot\rangle^r_\theta \sqcup \rangle\!\rangle^r_\theta$, all events are removed, to comply with the total freeze among the silent masses $\cdot\rangle^r_\theta$ and to make room for byzantine communication in the fault buffer $\rangle\rangle^r_\theta$. The resulting set complies with P_ε because all agents are delayable. For the silent masses, this is the desired result. For the fault buffer, on the other hand, byzantine sends are added for every correct or byzantine send in r, thus, ensuring that the incoming information in the reliable causal cone in r' is the same as in r. For the case when a faulty buffer node (l,m) sent no messages in the original run, $fail(l)$ is added to make the immediate future $(l,m+1)$ byzantine despite its silence, which is crucial for fulfilling Statement (C) and simplifying bookkeeping for byzantine agents.

The proof of (9)–(10) is by induction on $m = 0,\dots,t-1$. To avoid overlong formulas, we abbreviate the right-hand side of (9) by Υ^m_ε and the right-hand sides of (10) for each $j \in \mathscr{A}$ by Ξ^m_j for the specific $\alpha^m_j(r')$ and $\alpha^m_\varepsilon(r')$ defined in (11) and (12) respectively. Thus, it only remains to show that $\beta^m_\varepsilon(r') = \Upsilon^m_\varepsilon$ and $(\forall j \in \mathscr{A})\,\beta^m_j(r') = \Xi^m_j$, or equivalently, further abbreviating $\Upsilon^m_j := \Upsilon^m_\varepsilon \cap GEvents_j$, that

$$\beta^m_{\varepsilon_j}(r') = \Upsilon^m_j \qquad \text{and} \qquad \beta^m_j(r') = \Xi^m_j$$

for all $j \in \mathscr{A}$, by simultaneous induction on m.

Induction step for the silent masses $(j,m) \in \cdot\rangle^r_\theta$. By (12), $\alpha^m_{\varepsilon_j}(r') := \alpha^m_\varepsilon(r') \cap GEvents_j = \varnothing$, and filtering it yields $\Upsilon^m_j = \varnothing = \beta^m_{\varepsilon_j}(r')$ as prescribed by *CFreeze*. In particular, $go(j) \notin \beta^m_{\varepsilon_j}(r')$, thus, ensuring that filtering $\alpha^m_j(r')$, whatever it is, yields $\Xi^m_j = \varnothing = \beta^m_j(r')$, once again in compliance with *CFreeze* applied within $\cdot\rangle^r_\theta$.

Before proceeding with the induction step for the remaining nodes, observe that events in $\alpha^m_\varepsilon(r')$, if added to $r'(m)$, do not cross the byzantine-agent threshold f, meaning that the 1st-stage filtering does not affect $\alpha^m_\varepsilon(r')$:

$$filter^{\leq f}_\varepsilon\left(r'(m), \alpha^m_\varepsilon(r'), \alpha^m_1(r'), \dots, \alpha^m_n(r')\right) = \alpha^m_\varepsilon(r'). \tag{13}$$

Indeed there are two sources of byzantine events in $\alpha^m_\varepsilon(r')$: byzantine events from $\alpha^m_\varepsilon(r)$ that survived $filter^{\leq f}_\varepsilon$ in (12) and those pertaining to nodes in the fault buffer $\rangle\rangle^r_\theta$. The former were also present in $\beta^m_\varepsilon(r)$ in the original run because the 2nd-stage filter $filter^B_\varepsilon$ only removes correct (receive) events. At the same time, for any $(l,m) \in \rangle\rangle^r_\theta$, the immediate future $(l,m+1)$ was a faulty node in r by the definition of $\rangle\rangle^r_\theta$. In either case, any agent faulty in r' based on $\alpha^m_\varepsilon(r')$ was also faulty by timestamp $m+1$ in r. Additionally, any agent already faulty in $r'(m)$ was also faulty in $r(m)$ by Statement (D). Since the number of agents faulty by $m+1$ in the original transitional run r could not exceed f, adding $\alpha^m_\varepsilon(r')$ to $r'(m)$ does not exceed this threshold either. It follows from (13) that

$$\Upsilon^m_j = filter^B_\varepsilon(r'(m), \alpha^m_\varepsilon(r'), \alpha^m_1(r'), \dots, \alpha^m_n(r')) \cap GEvents_j. \tag{14}$$

Induction step for the fault buffer $(j,m) \in \rangle\rangle^r_\theta$. For these nodes, the $\alpha^m_{\varepsilon_j}(r')$ part of $\alpha^m_\varepsilon(r')$ contains no correct events, hence, $filter^B_\varepsilon$, which only removes correct receives, has no effect. In other words,

$$\begin{aligned}\Upsilon^m_j \;=\; \alpha^m_\varepsilon(r') \cap GEvents_j \;=\; \{fail(j)\} \;\sqcup\; \Big\{ & fake\,(j, gsend(j,h,\mu,id) \mapsto \mathbf{noop}) \;\Big|\\ & gsend(j,h,\mu,id) \in \beta^m_j(r) \vee (\exists A) fake\,(j, gsend(j,h,\mu,id) \mapsto A) \in \beta^m_\varepsilon(r) \Big\} \;=\; \beta^m_{\varepsilon_j}(r')\end{aligned}$$

as prescribed by $FakeEcho^m_j$. As in the case of the silent masses, $go(j) \notin \beta^m_\varepsilon(r')$ guarantees that the $\Xi^m_j = \varnothing = \beta^m_j(r')$ requirement is fulfilled within $\rangle\rangle^r_\theta$.

Induction step for the reliable causal cone $(j,m) \in \blacktriangleright^r_\theta$. The case of the nodes with a reliable causal path to θ, whose immediate future remains correct in r, is the final and also most complex induction step. Recall that $\alpha^m_j(r') = \alpha^m_j(r) \in P_j(r(m)) = P_j(r'(m))$ because within $\blacktriangleright^r_\theta$ by Statement (A) $r'(m) = r(m)$.

Thus, our choice of $\alpha^m_j(r')$ in (11) is in compliance with transitionality. Since $(j, m+1)$ is correct, the $\alpha^m_{\varepsilon_j}(r) := \alpha^m_\varepsilon(r) \cap GEvents_j$ part of $\alpha^m_\varepsilon(r)$ contained no byzantine events and, hence, is unchanged by (12). For the same reason it is not affected by 1st-stage filtering in either run. Thus, the same set of j's events undergoes the 2nd-stage filtering in both the original run r and in our transitional simulation of the adjusted run r'. Let us call this set of j's events Ω_j.

Since both $filter^B_\varepsilon$ and $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^m_j$ can only remove receive events, it immediately follows that $\beta^m_{\varepsilon_j}(r') \subseteq \beta^m_{\varepsilon_j}(r)$ and Υ^m_j agree on all non-receive events. Importantly, this includes $go(j)$ events, thus ensuring that $\Xi^m_j = \alpha^m_j(r)$.

A receive event $U = grecv(j,k,\mu,id) \in \Omega_j$ is retained in either run iff it is causally grounded by a matching send, correct or byzantine. Due to the uniqueness of GMI *id*, as ensured by the injectivity of both *id* and *global* functions, as well as Condition (c) of the t-coherency of sets produced by P_ε, there is at most one agent k's node where such a matching send can originate from. If *id* is not well-formed and no such send can exist, U is filtered out from both $\beta^m_\varepsilon(r)$ and Υ^m_ε, the former ensuring $U \notin \beta^m_\varepsilon(r')$. The reasoning in the case such a node $\eta = (k,z)$ exists depends on where timestamp z is relative to m and where η falls in our partition of nodes. Generally, to retain U in $\beta^m_{\varepsilon_j}(r)$ and Υ^m_j, one must find either a correct send $V := gsend(k,j,\mu,id)$ or a faulty send $W_A := fake\,(k, V \mapsto A)$ for some $A \in \overline{GActions_k} \sqcup \{\mathbf{noop}\}$.

- If $z > m$ is in the future of m, then U is filtered out from both $\beta^m_\varepsilon(r)$ and Υ^m_ε, hence, $U \notin \beta^m_\varepsilon(r')$.
- If $z \leq m$ and $\eta \in \cdot\rangle^r_\theta$, then, independently of filtering in r, hap $U \notin \mathfrak{e}\big(\rangle\!\rangle\!\blacktriangleright^r_\theta\text{-}Focus^m_j(r)\big) = \beta^m_{\varepsilon_j}(r')$ because the message's origin is outside the focus area. At the same time, no actions or events are scheduled at η in r' (for $z = m$ it follows from the already proven induction step for silent masses). Without either V or W_A, event U is filtered out from Υ^m_ε.
- If $z < m$ and $\eta \in \rangle\!\rangle^r_\theta$, then, by (4) in the definiton of $FakeEcho^z_k$, only $W_{\mathbf{noop}}$ can save U in Υ^m_j and $W_{\mathbf{noop}} \in \beta^z_{\varepsilon_k}(r')$ iff either $V \in \beta^z_k(r)$ or $W_A \in \beta^z_{\varepsilon_k}(r)$ for some A. Thus, filtering U yields the same result in both runs, and $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^m_j$ does not affect U because $\eta \in \rangle\!\rangle\!\blacktriangleright^r_\theta$.
- If $z = m$ and $\eta \in \rangle\!\rangle^r_\theta$, again only $W_{\mathbf{noop}}$ can save U in Υ^m_j, this time by construction (12) of $\alpha^m_\varepsilon(r') \not\ni go(k)$. Here $W_{\mathbf{noop}} \in \alpha^m_{\varepsilon_k}(r')$ iff either $V \in \beta^z_k(r)$ or $W_A \in \beta^z_{\varepsilon_k}(r)$ for some A. Thus, filtering U yields the same result in both runs, and $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^m_j$ does not affect U because $\eta \in \rangle\!\rangle\!\blacktriangleright^r_\theta$.
- If $z < m$ and $\eta \in\blacktriangleright^r_\theta$, then $(k, z+1)$ is still correct in both r and r', hence, no byzantine events such as W_A are present in either $r(m)$ or $r'(m)$. Accordingly, only V can save U in this case. Since $\beta^z_k(r) = \beta^z_k(r')$ by construction (5) of $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^z_k$, filtering U yields the same result in both runs, and $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^m_j$ does not affect U because $\eta \in \rangle\!\rangle\!\blacktriangleright^r_\theta$.
- If $z = m$ and $\eta \in\blacktriangleright^r_\theta$, again $(k, m+1)$ is correct in r meaning this time that no W_A are present in Ω_k. Again, only V can save U from filtering. Since $\alpha^m_k(r) = \alpha^m_k(r')$ by construction (11) and the sets of events being filtered agree on $go(k) \in \Omega_k$, here too filtering U yields the same result in both runs, and $\rangle\!\rangle\!\blacktriangleright^r_\theta$-$Focus^m_j$ does not affect U because $\eta \in \rangle\!\rangle\!\blacktriangleright^r_\theta$.

This case analysis completes the induction step for the reliable causal cone, the induction proof, proof of Statement (F), and the proof of the whole Lemma 10. □

5 Preconditions for Actions: Multipedes

Arguably the most important application of Lemma 10, and, hence, of causal cones, is to derive preconditions for agents' actions, cp. [1]. While relatively simple in traditional settings, where events can be preconditions according to the knowledge of preconditions principle [17] and where Lamport's causal cone suffices, this is no longer true in byzantine settings. As Theorem 12 reveals, if $f > 0$, an asynchronous agent can learn neither that it is (still) correct nor that a particular event[6] really occurred.

Theorem 12 ([12])**.** *If $f \geq 1$, then for any $o \in Events_i$, for any interpreted system $\mathscr{I} = (R^\chi, \pi)$ with any non-excluding agent-context $\chi = ((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P)$ where i is gullible and every $j \neq i$ is delayable,*

$$\mathscr{I} \models \neg K_i \overline{occurred}(o) \qquad \text{and} \qquad \mathscr{I} \models \neg K_i correct_i. \tag{15}$$

These validities can be shown by modeling the infamous *brain in a vat* scenario (see [12] for details).

Theorem 12 obviously implies that *knowledge* of simple preconditions, e.g., events, is never achievable if byzantine agents are present. Settling for the next best thing, one could investigate whether i knows o has happened relative to its own correctness, i.e., whether $K_i(correct_i \to \overline{occurred}(o))$ holds (cf. [18]), a kind of non-factive *belief* in o. This means that i can be mistaken about o due to its own faults (in which case it cannot rely on any information anyway), not due to being misinformed by other agents. It is, however, sometimes overly restrictive to assume that $K_i(correct_i \to \overline{occurred}(o))$ holds in situations when i is, in fact, faulty: typical specifications, e.g., for distributed agreement [15], do not restrict the behavior of faulty agents, and agents might sometimes learn that they are faulty. We therefore introduced the *hope* modality

$$H_i\varphi := correct_i \to K_i(correct_i \to \varphi),$$

which was shown in [7] to be axiomatized by adding to K45 the axioms $correct_i \to (H_i\varphi \to \varphi)$, and $\neg correct_i \to H_i\varphi$, and $H_i correct_i$.

The following Theorem 13 shows that hope is also closely connected to reliable causal cones, in the sense that events an agent can hope for must lie within the reliable causal cone.

Theorem 13. *For a non-excluding agent-context $\chi = ((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P)$ such that all agents are gullible, correctable, and delayable, for a correct node $\theta = (i,t)$, and for an event $o \in Events$, if all occurrences of $O \in \overline{GEvents}$ such that $local(O) = o$ happen outside the reliable causal cone $\blacktriangleright^r_\theta$ of a run $r \in R^\chi$, i.e., if $O \in \beta^m_\varepsilon(r) \cap \overline{GEvents}_j$ & $local(O) = o$ implies $(j,m) \notin \blacktriangleright^r_\theta$, then for any $\mathscr{I} = (R^\chi, \pi)$,*

$$(\mathscr{I}, r, t) \not\models H_i \overline{occurred}(o).$$

Proof. Constructing the first t rounds according to the adjustment from Lemma 10 and extending this prefix to an infinite run $r' \in R^\chi$ using the non-exclusiveness of χ, we obtain a run with no correct events recorded as o. Indeed, in r', there are no events originating from $\cdot\rangle^r_\theta$, no correct events from $\rangle\rangle^r_\theta$, and all events originating from $\blacktriangleright^r_\theta$, though correct, were also present in r and, hence, do not produce o in local histories. At the same time, $r_i(t) = r'_i(t)$ by Lemma 10(B), making $(\mathscr{I}, r', t)$ indistinguishable for i, and $(\mathscr{I}, r', t) \models correct_i$ by Lemma 10(D). $\square$

It is interesting to compare the results and proofs of Theorems 12 and 13. Essentially, in the run r' modeling the brain in a vat in the former, i is a faulty agent that perceives events while none really happen. Therefore, $K_i\overline{occurred}(o)$ can *never* be attained. In the run r' constructed by Lemma 10 in Theorem 13, on the other hand, i remains correct. The reason that $H_i\overline{occurred}(o)$ fails here is that o does not occur within the reliable causal cone.

Theorem 13 shows that, in order to act based on the hope that an event occurred, it is necessary that the event originates from the reliable causal cone. Unfortunately, this is not sufficient. Consider the case of a run r where no agent exhibits a fault: every causal message chain is reliable and the ordinary

[6]Actually, the reasoning in this section also extends to actions, i.e., arbitrary haps.

and reliable causal cones coincide. However, since up to f agents *could be* byzantine, it is trivial to modify r by seeding $\mathit{fail}(j)$ events in round ½ for several agents j in a way that is indistinguishable for agent i trying to hope for the occurrence of o. This would enlarge the fault buffer and shrink the reliable causal cone in the so-constructed adjusted run $\hat{r}$. Obviously, by making different sets of agents byzantine (without violating f, of course), one can fabricate multiple adjusted runs where $\hat{r}_i(t) = r_i(t)$ is exactly the same but fault buffers and reliable causal cones vary in size and shape. Any single one of those $\hat{r}$ satisfying the conditions of Theorem 13, in the sense that all occurrences of o happen outside its reliable causal cone, dash the *hope* of i for o in r.

Thus, in order for i to have hope at (i,t) in run r that o really occurred, it is necessary that some correct global version O of o (not necessarily the same one) is present somewhere (not necessarily at the same node) in the reliable causal cone of *every* run $\hat{r}$ that ensures $r_i(t) = \hat{r}_i(t)$. This gives rise to the definition of a *multipede*, which ensures $(\mathscr{I}, r, t) \models H_i\overline{occurred}(o)$ according to Theorem 13:

Definition 14 (Multipede)**.** We say that a run r in a non-excluding agent context $\chi = ((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P)$ contains a multipede$^o_\theta$ for event $o \in Events$ at some node $\theta = (i,t)$ iff, for all runs $\hat{r} \in R^\chi$ with $r_i(t) = \hat{r}_i(t)$, it holds that o happens inside its reliable causal cone, i.e., that

$$(\exists (j,m) \in \blacktriangleright^{\hat{r}}_\theta)\, (\exists O \in \overline{GEvents_j})\big(O \in \beta^m_\varepsilon(\hat{r})\ \&\ local(O) = o\big).$$

We obtain the following necessary condition for the existence of a multipede:

Theorem 15 (Necessary condition for a multipede)**.** *Given an arbitrary non-excluding agent-context $\chi = \big((P_\varepsilon, \mathscr{G}(0), \tau_f, \Psi), P\big)$ such that all agents are gullible, correctable, and delayable and for any run $r \in R^\chi$ in any interpreted system $\mathscr{I} = (R^\chi, \pi)$, if $(\mathscr{I}, r, t) \models H_i\overline{occurred}(o)$ for a correct node $\theta = (i,t)$, i.e., if there is a multipede$^o_\theta$ in r, then the following must hold: Let $Byz^r_\theta := \{j \in \mathscr{A} \mid (\exists m)(j,m) \in \rangle\!\rangle^r_\theta\}$. For any $S \subseteq \mathscr{A} \setminus (\{i\} \sqcup Byz^r_\theta)$ such that $|S| = f - |Byz^r_\theta|$, there must exist a witness $w_S \in \mathscr{A}$ of some correct event $O_S \in \beta^{m_S}_\varepsilon(r) \cap \overline{GEvents}_{w_S}$ such that $local(O_S) = o$ and such that there is causal path $(w_S, m_S) \rightsquigarrow^r_{\xi_S} \theta$ that does not involve agents from $S \sqcup Byz^r_\theta$.*

Proof. Since, by Lemma 10, the adjusted run $r' \in R^\chi$ and since the only faults up to t occur in r' in the fault buffer $\rangle\!\rangle^r_\theta$, i.e., pertain to agents from Byz^r_θ, for any S described above, one can construct first t rounds by setting $\beta^0_\varepsilon(r^S) := \beta^0_\varepsilon(r') \sqcup \{\mathit{fail}(j) \mid j \in S \sqcup Byz^r_\theta\}$ and keeping the rest of r' intact. These first t rounds can be extended to complete infinite runs $r^S \in R^\chi$ indistinguishable for i at θ from either r' or r because the addition of $\mathit{fail}(j)$ is imperceptible for agents and does not affect protocols. The only potentially affected element could have been $filter_\varepsilon$ in the part ensuring byzantine agents do not exceed f in number, but it also behaves the same way as in r' because $|S| + |Byz^r_\theta| = f$. Since $r_i(t) = r'_i(t) = r^S_i(t)$, we have $(\mathscr{I}, r^S, t) \models H_i\overline{occurred}(o)$. Node θ remains correct in these runs because $i \notin S$. Thus, by Theorem 13, each run r^S must have a requisite correct event $O_S \in \beta^{m_S}_{\varepsilon_{w_S}}(r^S) \cap \blacktriangleright^{r^S}_\theta$. It remains to note that any such correct event from r_S must be present in r' and in r and any causal path in r^S exists already in r' and r, according to the construction from Lemma 10. Thus, there must exist a causal path ξ in r from (w_S, m_S) to θ such that ξ is reliable in r^S. Finally, since all f byzantine agents in r^S, namely $S \sqcup Byz^r_\theta$, are made faulty from round ½, path ξ_S being reliable in r^S means not involving these agents. ☐

From the perspective of protocol design, arguably, of more interest are sufficient conditions for the existence of a multipede in a given run. Whereas a sufficient condition could be obtained directly from Def. 14, of course, identifying all the transitional runs $\hat{r}$ with $r_i(t) = \hat{r}_i(t)$ is far from being computable in general. Actually, we conjecture that sufficient conditions cannot be formulated in a protocol-independent way at all. Unfortunately, however, protocol-dependence cannot be expected to be simple either. For instance, even just varying the number and location of faults in r for suppressing $\overline{occurred}(o)$ in a modified run $\hat{r}$ could be non-trivial. If k agents are already faulty in run r, at least $f-k$ ones can freely be used for this purpose. However, some of the k byzantine faults in r may also be re-located in $\hat{r}$, as agents that only become faulty after timestamp t cannot be part of any fault buffer. Rather than making them faulty, it would suffice to just freeze them.

For instance, for the following communication structure with $f = 2$ and agents 1 and 2 being byzantine (we omit the time dimension for simplicity's sake),

$$\begin{tikzcd}
\boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{3} & \boxed{4} \arrow[l]
\end{tikzcd}$$

they would both participate in the fault buffer, whereas, already 2 alone would suffice because even were 1 correct, the observed communication does not give it a chance to pass by 2. Depending on 1's protocol, it might be possible to reassign 1 to the silent masses, thereby allowing to consider 4 as the second faulty agent and, thus, showing the impossibility for 3 to act in this situation. An opposite outcome is possible in the following scenario:

$$\begin{tikzcd}
 & \boxed{A1.1} \arrow[dr] & & \boxed{A2.1} \arrow[dl] & \\
\boxed{I1} \arrow[ur] \arrow[r] \arrow[dr] & \boxed{A1.2} \arrow[r] & \boxed{C} & \boxed{A2.2} \arrow[l] & \boxed{I2} \arrow[ul] \arrow[l] \arrow[dl] \\
 & \boxed{A1.3} \arrow[ur] & & \boxed{A2.3} \arrow[ul] &
\end{tikzcd}$$

Let $f = 2$ and the faulty agents be $A2.1$ and $A1.1$. While the sufficient condition forces C to consider the case of both $I1$ and $I2$ being compromised and information originating from them unreliable, our necessary condition does not rule out C's ability to make a decision. Indeed, suppose $I1$ and $I2$ are investigators sending in their reports via three aides each. Having received 4 identical reports that are correct from $A1.2$, $A1.3$, $A2.2$, and $A2.3$ and only 2 fake reports from $A1.1$ and $A2.1$, agent C would have been able to choose the correct version if the possibility of both investigators being compromised were off the table. Our method of adjusting the run does not allow us to move the faulty agent from $A1.1$ to $I1$ because it is not clear how $A1.1$ would have behaved were it correct and had it received a fake report from $I1$. By designing a protocol in such a way that $A1.1$'s correct behavior in such a hypothetical situation is different, we can eliminate the possibility of investigators being compromised and, thus, resolve the situation for C.

6 Conclusions

The main contribution of this paper is the characterization of the analog of Lamport's causal cone in asynchronous multi-agent systems with byzantine faulty agents. Relying on our novel byzantine runs-and-systems framework, we provided an accurate epistemic characterization of causality and the induced reliable causal cone in the presence of asynchronous byzantine agents. Despite the quite natural final shape of a reliable causal cone, it does not lead to simple conditions for ascertaining preconditions: the detection of what we called a multipede is considerably more complex than the verification of the existence of one causal path in the fault-free case. Since the agents' actions depend on the shape of multiple alternative reliable causal cones in byzantine fault-tolerant protocols like [20], however, there is no alternative but to detect multipedes.

Developing practical sufficient conditions for the existence of a multipede poses exciting challenges, which are currently being addressed in the context of the epistemic analysis of some real byzantine fault-tolerant protocols. This context-dependency is unavoidable, since the agent that tries to detect a multipede in a run lacks global information such as the actual members of the fault buffer. On the other hand, the gap between the necessary and sufficient conditions can potentially be minimized by designing protocols based on the insights into the causality structure we have uncovered. For instance, while we treated all error-creating nodes as part of the fault buffer in our necessary conditions for a multipede, it is sometimes possible to relegate redundant parts of it into the silent masses. As this would allow to re-locate byzantine faults for intercepting more causal paths, one may design protocols in a way that does not allow this.

A larger and more long-term goal is to extend our study to syncausality and the reliable syncausal cone in the context of synchronous byzantine fault-tolerant multi-agent systems, and to possibly incorporate protocols explicitly into the logic.

Acknowledgments. We are grateful to Yoram Moses, Hans van Ditmarsch, and Moshe Vardi for fruitful discussions and valuable suggestions that have helped shape the final version of this paper. We also thank the anonymous reviewers for their comments and suggestions.

References

[1] Ido Ben-Zvi & Yoram Moses (2014): *Beyond Lamport's* Happened-before*: On Time Bounds and the Ordering of Events in Distributed Systems*. Journal of the ACM 61(2:13), doi:10.1145/2542181.

[2] K. M. Chandy & Jayadev Misra (1986): *How processes learn*. Distributed Computing 1(1), pp. 40–52, doi:10.1007/BF01843569.

[3] Reinhard Diestel (2017): *Graph Theory*, Fifth edition. Springer, doi:10.1007/978-3-662-53622-3.

[4] Cynthia Dwork & Yoram Moses (1990): *Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures*. Information and Computation 88(2), pp. 156–186, doi:10.1016/0890-5401(90)90014-9.

[5] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1995): *Reasoning About Knowledge*. MIT Press.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1999): *Common knowledge revisited*. Annals of Pure and Applied Logic 96(1–3), pp. 89–105, doi:10.1016/S0168-0072(98)00033-5.

[7] Krisztina Fruzsa (2019): *Hope for Epistemic Reasoning with Faulty Agents!* In: Proceedings of ESSLLI 2019 Student Session. (To appear).

[8] Guy Goren & Yoram Moses (2018): *Silence*. In: PODC '18, Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, ACM, pp. 285–294, doi:10.1145/3212734.3212768.

[9] Joseph Y. Halpern & Yoram Moses (1990): *Knowledge and Common Knowledge in a Distributed Environment*. Journal of the ACM 37(3), pp. 549–587, doi:10.1145/79147.79161.

[10] Joseph Y. Halpern, Yoram Moses & Orli Waarts (2001): *A characterization of eventual Byzantine agreement*. SIAM Journal on Computing 31(3), pp. 838–865, doi:10.1137/S0097539798340217.

[11] Jaakko Hintikka (1962): *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press.

[12] Roman Kuznets, Laurent Prosperi, Ulrich Schmid & Krisztina Fruzsa (2019): *Epistemic Reasoning with Byzantine-Faulty Agents*. In: Proceedings of FroCoS 2019. (To appear).

[13] Roman Kuznets, Laurent Prosperi, Ulrich Schmid, Krisztina Fruzsa & Lucas Gréaux (2019): *Knowledge in Byzantine Message-Passing Systems I: Framework and the Causal Cone*. Technical Report TUW-260549, TU Wien. Available at https://publik.tuwien.ac.at/files/publik_260549.pdf.

[14] Leslie Lamport (1978): *Time, Clocks, and the Ordering of Events in a Distributed System*. Communications of the ACM 21(7), pp. 558–565, doi:10.1145/359545.359563.

[15] Leslie Lamport, Robert Shostak & Marshall Pease (1982): *The Byzantine Generals Problem*. ACM Transactions on Programming Languages and Systems 4(3), pp. 382–401, doi:10.1145/357172.357176.

[16] Alexandre Maurer, Sébastien Tixeuil & Xavier Defago (2015): *Reliable Communication in a Dynamic Network in the Presence of Byzantine Faults*. eprint 1402.0121, arXiv. Available at https://arxiv.org/abs/1402.0121.

[17] Yoram Moses (2015): *Relating Knowledge and Coordinated Action: The Knowledge of Preconditions Principle*. In R. Ramanujam, editor: Proceedings of TARK 2015, pp. 231–245, doi:10.4204/EPTCS.215.17.

[18] Yoram Moses & Yoav Shoham (1993): *Belief as defeasible knowledge*. Artificial Intelligence 64(2), pp. 299–321, doi:10.1016/0004-3702(93)90107-M.

[19] Yoram Moses & Mark R. Tuttle (1988): *Programming Simultaneous Actions Using Common Knowledge*. Algorithmica 3, pp. 121–169, doi:10.1007/BF01762112.

[20] T. K. Srikanth & Sam Toueg (1987): *Optimal Clock Synchronization*. Journal of the ACM 34(3), pp. 626–645, doi:10.1145/28869.28876.

Appendix

Filter functions

Definition A.16. The filtering function $filter_\varepsilon$ for asynchronous agents with at most $f \geq 0$ byzantine faults is defined as follows.

First, we define a subfilter $filter^B_\varepsilon \colon \mathscr{G} \times \wp(GEvents) \times \prod_{i=1}^n \wp(\overline{GActions_i}) \to \wp(GEvents)$ that removes impossible receives: for a global state $h \in \mathscr{G}$, set $X_\varepsilon \subseteq GEvents$, and sets $X_i \subseteq \overline{GActions_i}$,

$$\begin{aligned} filter^B_\varepsilon(h, X_\varepsilon, X_1, \dots, X_n) := X_\varepsilon \setminus \Big\{ grecv(j,i,\mu,id) \;\Big|\; & gsend(i,j,\mu,id) \notin h_\varepsilon \quad \wedge \\ (\forall A \in \{\mathbf{noop}\} \sqcup \overline{GActions_i}) fake\,(i, gsend(i,j,\mu,id) \mapsto A) \notin h_\varepsilon \wedge & (gsend(i,j,\mu,id) \notin X_i \vee go(i) \notin X_\varepsilon) \wedge \\ (\forall A \in \{\mathbf{noop}\} \sqcup \overline{GActions_i}) & fake\,(i, gsend(i,j,\mu,id) \mapsto A) \notin X_\varepsilon \Big\}, \end{aligned}$$

where h_ε is the environment's record of all haps in the global state h and $O \in h_\varepsilon$ ($O \notin h_\varepsilon$) states that the hap $O \in GHaps$ is (isn't) present in this record of all past rounds, X_ε represents all events attempted by the environment and X_i's represent all actions attempted by agents i in the current round.

Second, using $X^B_{\varepsilon_i} := X_\varepsilon \cap \big(BEvents_i \sqcup \{sleep\,(i), hibernate\,(i)\}\big)$ and defining $\mathscr{A}(Failed\,(h))$ to be the set of agents who have already exhibited faulty behavior in the global state h, we define a subfilter $filter^{\leq f}_\varepsilon \colon \mathscr{G} \times \wp(GEvents) \times \prod_{i=1}^n \wp(\overline{GActions_i}) \to \wp(GEvents)$ that removes all byzantine events in the situation when having them would have exceeded the f threshold:

$$filter^{\leq f}_\varepsilon(h, X_\varepsilon, X_1, \dots, X_n) := \begin{cases} X_\varepsilon & \text{if } \left|\mathscr{A}(Failed\,(h)) \cup \left\{ i \mid X^B_{\varepsilon_i} \neq \varnothing \right\}\right| \leq f, \\ X_\varepsilon \setminus \bigsqcup_{i \in \mathscr{A}} X^B_{\varepsilon_i} & \text{otherwise.} \end{cases}$$

The filter $filter_\varepsilon \colon \mathscr{G} \times \wp(GEvents) \times \prod_{i=1}^n \wp(\overline{GActions_i}) \to \wp(GEvents)$ is obtained by composing these two subfilters, with the $\leq f$ subfilter applied first:

$$filter_\varepsilon(h, X_\varepsilon, X_1, \dots, X_n) := filter^B_\varepsilon\left(h, filter^{\leq f}_\varepsilon(h, X_\varepsilon, X_1, \dots, X_n), X_1, \dots, X_n\right).$$

The composition in the opposite order could violate causality if a message receipt is preserved by $filter^B_\varepsilon$ based on a byzantine send in the same round, which is later removed by $filter^{\leq f}_\varepsilon$.

Definition A.17. The filters $filter_i \colon \prod_{j=1}^n \wp(\overline{GActions}_j) \times \wp(GEvents) \to \wp(\overline{GActions}_i)$ for agents' actions are defined as follows: for X_ε representing all environment's events and X_i representing all actions attempted by agent i in the current round,

$$filter_i(X_1, \dots, X_n, X_\varepsilon) := \begin{cases} X_i & \text{if } go(i) \in X_\varepsilon, \\ \varnothing & \text{otherwise.} \end{cases}$$

Update functions

Before defining the update functions, we need several auxiliary functions:

Definition A.18. We use a function $local \colon \overline{GHaps} \to Haps$ converting *correct* haps from the global format into the local formats for the respective agents in such a way that, for any $i, j \in \mathscr{A}$, any $t \in \mathbb{T}$, any $a \in Actions_i$, any $\mu \in Msgs$, and any $M \in \mathbb{N}$:

1. $local(\overline{GActions}_i) = Actions_i$;
2. $local(\overline{GEvents}_i) = Events_i$;
3. $local\big(global(i,t,a)\big) = a$;
4. $local\big(grecv(i, j, \mu, M)\big) = recv(j, \mu)$.

For all other haps, the localization cannot be done on a hap-by-hap basis because system events and byzantine events $fake\,(i, A \mapsto \textbf{noop})$ do not create a local record. Accordingly, we define a *localization function* $\sigma \colon \wp(GHaps) \to \wp(Haps)$ as follows: for each $X \subseteq GHaps$,

$$\sigma(X) := local\Big((X \cap \overline{GHaps}) \quad \cup \\ \{E \in \overline{GEvents} \mid (\exists i) fake\,(i, E) \in X\} \cup \{A' \in \overline{GActions} \mid (\exists i)(\exists A) fake\,\big(i, A \mapsto A'\big) \in X\}\Big).$$

Definition A.19. We abbreviate $X_{\varepsilon_i} := X_\varepsilon \cap GEvents_i$ for performed events $X_\varepsilon \subseteq GEvents$ and actions $X_i \subseteq \overline{GActions}_i$ for each $i \in \mathscr{A}$. Given a global state $r(t) = \big(r_\varepsilon(t), r_1(t), \dots, r_n(t)\big) \in \mathscr{G}$, we define agent i's $update_i \colon \mathscr{L}_i \times \wp(\overline{GActions}_i) \times \wp(GEvents) \to \mathscr{L}_i$ that outputs a new local state from $\mathscr{L}_i$ based on i's actions X_i and events X_ε:

$$update_i(r_i(t), X_i, X_\varepsilon) := \begin{cases} r_i(t) & \text{if } \sigma(X_{\varepsilon_i}) = \varnothing \text{ and } X_{\varepsilon_i} \cap \{go(i), sleep(i)\} = \varnothing, \\ \big[\sigma\big(X_{\varepsilon_i} \sqcup X_i\big)\big] : r_i(t) & \text{otherwise} \end{cases}$$

(note that in transitional runs, $update_i$ is always used after the action $filter_i$, thus, in the absence of $go(i)$, it is always the case that $X_i = \varnothing$).

Similarly, the *environment's state update*$_\varepsilon \colon \mathscr{L}_\varepsilon \times \wp(GEvents) \times \prod_{i=1}^n \wp(\overline{GActions}_i) \to \mathscr{L}_\varepsilon$ outputs a new state of the environment based on events X_ε and all actions X_i:

$$update_\varepsilon(r_\varepsilon(t), X_\varepsilon, X_1, \dots, X_n) := (X_\varepsilon \sqcup X_1 \sqcup \dots \sqcup X_n) \colon r_\varepsilon(t).$$

Accordingly, the global update function $update \colon \mathscr{G} \times \wp(GEvents) \times \prod_{i=1}^n \wp(\overline{GActions}_j) \to \mathscr{G}$ modifies the global state as follows:

$$update(r(t), X_\varepsilon, X_1, \dots, X_n) := \Big(update_\varepsilon(r_\varepsilon(t), X_\varepsilon, X_1, \dots, X_n), \\ update_1(r_1(t), X_1, X_\varepsilon), \dots, update_n(r_n(t), X_n, X_\varepsilon)\Big).$$