Common Knowledge and Consistent Simultaneous Coordination

Gil Neiger*
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
gil@cc.gatech.edu

Mark R. Tuttle
DEC Cambridge Research Lab
One Kendall Square, Building 700
Cambridge, Massachusetts 02139
tuttle@crl.dec.com

Abstract

Traditional problems in distributed systems include the *Reliable Broadcast*, *Distributed Consensus*, and *Distributed Firing Squad* problems. These problems require coordination only among the processors that do not fail. In systems with benign processor failures, however, it is reasonable to require that a faulty processor's actions are consistent with those of nonfaulty processors, assuming that it performs any action at all. We consider problems requiring consistent, simultaneous coordination and analyze these problems in terms of common knowledge. (Others have performed similar analyses of traditional coordination problems [1,9].) In several failure models, we use our analysis to give round-optimal solutions. In one benign failure model, however, we show that such problems cannot be solved, even in failure-free executions.

1 Introduction

Concurrent message-passing systems provide many advantages over other kinds of systems, including efficiency and fault-tolerance. Most interesting computation in such systems requires processors to coordinate their actions in some way, but this coordination becomes especially difficult when processors can fail. Traditionally, researchers have considered problems that require only the correct processors to coordinate their actions, placing no requirements on the behavior of faulty processors. For example, in one formulation of the *Byzantine Agreement* problem (cf. [12]), each processor begins with an input bit and chooses an output bit, subject to the condition that all nonfaulty processors choose the same value as their output bit, and that this value is the value of some processor's input bit. This problem was originally proposed in the *Byzantine* failure model where faulty processors can behave in completely arbitrary ways. Consequently, only the nonfaulty processors are required to agree on their output bits, since it is impossible to force the faulty processors to do anything. Subsequently, however, a substantial body of the literature has considered this problem in systems where processors fail in relatively benign ways, such as intermittently failing to send or receive certain messages.

In the context of such benign failures, however, it seems reasonable to require that, if a faulty processors performs any action at all, then this action is *consistent* with the actions performed by the nonfaulty processors. For example, Neiger and Toueg [11] define *Uniform Agreement*, a variant of *Reliable Broadcast* in which faulty processors are forbidden to choose inconsistent values; and Gopal and Toueg [2] consider a family of related problems. It is often not only reasonable, but also critical, to consistently coordinate the behavior of *all* processors. In distributed database systems, for example, it is undesirable for sites to acquire inconsistent views of the database simply because of intermittent

*This author was supported in part by the National Science Foundation under grant CCR-8909663.

communication failures. Not surprisingly, however, achieving consistent coordination is more difficult than general coordination, where faulty processors are not forbidden to act inconsistently.

Problems requiring consistently coordinated actions promise to become increasingly important in the study of distributed computing. In this paper, we define the class of *consistent simultaneous choice* problems, problems in which all processors (faulty or nonfaulty) that perform an action perform the same action at the same time. We then analyze the state of knowledge that processors must attain to solve these problems and we use the results of this analysis to derive round-optimal solutions to these problems in a number of benign failure models.

The idea of analyzing the state of knowledge attained by a processor was first introduced by Halpern and Moses [5], who present a method for formally ascribing knowledge to processors and show how knowledge can be used to analyze distributed computation. In their work, the highest state of knowledge a group of processors can attain is the state of *common knowledge*: intuitively, a fact is common knowledge if everyone knows it, everyone knows that everyone knows it, and so on.

Since then, many researchers have analyzed problems in terms of the state of knowledge that processors in a system must attain to solve them. In particular, several papers have demonstrated a close relationship between common knowledge and problems requiring simultaneous coordination by the nonfaulty processors. For example, Dwork and Moses [1] consider the problem of *Simultaneous Byzantine Agreement* (a version of *Byzantine Agreement* or *Distributed Consensus* in which all correct processors choose their output bit simultaneously) in the crash failure model—a model in which processors fail by simply halting—and show that processors can reach agreement only when they have achieved common knowledge of certain facts. Using this, they derive a solution that is *optimal in all runs*: in any given context—a fixed pattern of failures and processor inputs—the round in which this solution causes processors reach agreement is at least as early as the round in which any other solution would do so. (Optimality is therefore measured in terms of rounds and not computational complexity.) Moses and Tuttle [9] extend this work to general simultaneous actions and consider systems with more general failures. They show that the coordination of any simultaneous choice requires the correct processors to achieve common knowledge. Using this observation, they too derive optimal protocols in the models they consider. However, in one of these models, the general omissions model—a model in which faulty processors may omit to send or receive messages—their optimal protocols require exponential time. In fact, using the close relationship between common knowledge and simultaneity, they are able to prove that optimally achieving such coordination in this model is inherently intractable (specifically, the problem is shown to be NP-hard).

In this paper, we perform a similar analysis of *consistent* simultaneous choice problems, but our analysis requires a generally stronger definition of common knowledge. In systems with crash and send omission failures, however, the two forms of common knowledge are equivalent (when considered with respect to the group of nonfaulty processors); the optimal protocols derived earlier [1,9] for general simultaneous choice problems also solve the corresponding consistent problems and do so optimally. In systems with general omission failures, we show that the two forms are again equivalent if fewer than half the processors in the system may fail; a minor modification to the previous protocols optimally solves consistent simultaneous choice problems in such systems. Surprisingly, in all these cases, optimal protocols for the consistent version of a problem halt as soon as optimal protocols for the original version: consistency does not delay action.

The same is not true in the general omissions model when more than half the processors may fail. In such systems, we show that the two forms of common knowledge are *not* equivalent and, furthermore, that consistent simultaneous choice problems cannot be solved. This remains true even if the definition of such problems is weakened so that processors are required to act only in failure-free executions.

The remainder of this paper is organized as follows. Section 2 defines our model of computation, and Section 3 defines consistent simultaneous choice problems. Section 4 defines a processor's state of knowledge, including common knowledge. Section 5 analyzes solutions to consistent coordination problems in terms of common knowledge and presents several optimal solutions to these problems in several variants of the omissions failure model. Section 6 shows that these problems cannot be implemented in certain systems with general omission failures. Section 7 contains some concluding remarks and mentions some open problems.

2 Model of a System

In this section, we define our model of a distributed system. Our model is closely related to others used in studying knowledge and coordination [1,5,9].

This paper considers synchronous systems of unreliable processors. Such a system consists of a finite collection $P = \{p_1, \ldots, p_n\}$ of processors, each pair of which is connected by a two-way communication link. All processors share a discrete global clock that starts at time 0 and advances in increments of one. Computation proceeds in a sequence of *rounds*, with round k taking places between time $k-1$ and time k. Each processor has a read-only *input register*, which is initialized to some value referred to as the processor's *initial input*. At time 0, each processor starts in some *initial state*, which includes its initial input. Then, in every round, the processor performs some local computation and perhaps some other actions, sends messages to a set of other processors, and receives the set of messages sent to it by other processors in that round. Each message is assumed to be tagged with the identities of the sender and intended receiver of the message, as well as the round in which it is sent. At any given time, a processor's *message history* consists of the list of messages it has received from the other processors.

A processor's *local state* (or *view*) at any given time consists of its initial input, its message history, the time on the global clock, and the processor's identity. A *global state* is a tuple $\langle s_1, \ldots, s_n, s_e \rangle$ of local states, one local state s_i for each processor p_i and one state s_e for the *environment*. Intuitively, the environment includes everything about the state of the system that cannot be deduced from the local states of the processors, such as the failure pattern and the protocol the processors follow (see below). A *run* of the system is an infinite sequence of global states. An ordered pair $\langle \rho, l \rangle$, where ρ is a run and l is a natural number, is called a *point*, and represents the state of the system after the first l rounds of ρ. Processor q's view at time l in the run ρ (at the point $\langle \rho, l \rangle$) is denoted by $v(q, \rho, l)$.

Processors in a system follow a *protocol*, which specifies what messages a processor is required to send during a round (and what other actions the processor is required to perform) as a deterministic function of its local state. In other words, a protocol is a function from a processor's local state to its next state, a list of local actions it is to perform, and a list of messages it is to send to other processors. While all processors change state and perform actions as required by the protocol, some processors may be *faulty*; such a processor may fail to send any of the messages the protocol requires it to send or fail to receive some of the messages sent to it by other processors.[1]

More precisely, a *faulty behavior* for a processor p_i is a pair of functions S_i and R_i from round numbers to sets of processors. Intuitively, $S_i(k)$ is the set of processors to which p_i fails to send a message in round k, and $R_i(k)$ is the set of processors from which p_i fails to receive a message. A *failure pattern* is a collection of faulty behaviors $\langle S_i, R_i \rangle$, one for each processor p_i. The *failure pattern* of a run is a failure pattern such that for every round k and every processor p_i

- p_i sends no messages to processors in $S_i(k)$ in round k but sends all required messages to processors not in $S_i(k)$, and
- p_i receives no messages from processors in $R_i(k)$ in round k but receives all messages sent to it by processors not in $R_i(k)$.

We assume that a run's failure pattern is encoded in the state of the environment. Since it is possible for several failure patterns to be consistent with the pattern of messages sent during a given

[1]This paper does not consider processors that may fail *arbitrarily* [7]; Neiger and Toueg [11] give formal definitions of a range of different failure models.

execution, it is possible for several runs to differ only in the failure pattern encoded in the state of the environment. Processor p_i is *correct* if $S_i(k) = R_i(k) = \emptyset$ for all k.

This work considers three failure models:

1. the *crash* model [3,14], in which $R_i(k)$ is empty for every k, and in which $S_i(k)$ nonempty implies $S_i(k') = P$ for every $k' > k$;[2]

2. the *send omissions* model [3,8], in which $R_i(k)$ is empty for every k, and there are no restrictions on $S_i(k)$; and

3. the *general omissions* model [9,13], in which there are no restrictions on either $R_i(k)$ or $S_i(k)$.

Thus, in the crash model, faulty processors simply halt at some point (stop sending messages); in the send omissions model, faulty processors intermittently fail to send some messages; and in the general omissions model, faulty processors intermittently fail both to send and receive messages.

Given a run ρ, if ι_i is p_i's initial input in ρ, then $\bar{\iota} = \langle \iota_1, \ldots, \iota_n \rangle$ is the *input* to ρ. A pair $\langle \pi, \bar{\iota} \rangle$, where π is a failure pattern and $\bar{\iota}$ is an input, is called an *operating environment.* Note that a run is uniquely determined by a protocol and an operating environment. Two runs of two different protocols are said to be *corresponding runs* if they have the same operating environment. The fact that an operating environment is independent of the protocol allows the comparison of different protocols according to their behavior in corresponding runs.

At several places in this paper, it will be convenient to refer to runs that differ only in some aspect of their operating environment. We say that two runs ρ and ρ' of a protocol P *differ only in* some aspect of their operating environment if and only if ρ and ρ' are the result of running P in operating environments that differ only in the given aspect. As an example, if ρ and ρ' differ only in the input to p_i, then their operating environments are identical except for p_i's input, although the actual messages sent by processors may depend on p_i's input and hence be very different in the two runs. As another example, suppose ρ and ρ' differ only in that every processor sends a message to p_i in round l (in the operating environment) of ρ'. It is possible that P does not require processors to send any messages to p_i in round l, in which case the messages actually sent in ρ and ρ' are the same; we simply mean that the operating environment itself does not keep processors from sending messages to p_i in round l of ρ'.

This work studies the behavior of protocols in the presence of a bounded number of failures (of a particular type) and a given set of possible initial inputs to the processors. It is therefore natural to identify a *system* with the set of all possible runs of a given protocol under such circumstances. Formally, a system is identified with the set of runs of a protocol $\mathcal{P}$ by n processors, at most $t \leq n-2$ of which may be faulty (in the sense of a particular failure model $\mathcal{M}$), where the initial input of each processor p_i is an element of set $\mathcal{I}$. This set of runs is denoted by the tuple $\Sigma = \langle n, t, \mathcal{P}, \mathcal{M}, \mathcal{I} \rangle$.

While a protocol may be thought of as a function of a processor's view, protocols for distributed systems are typically written for systems of arbitrarily large size. In this sense, the actions and messages required of a processor by a protocol actually depend on the number of processors in the system (and, perhaps, the bound on the number of failures) as well as the view of the processor. Therefore, a *protocol* is formally defined to be a function from n, t, and a processor's view to a list of actions the processor is required to perform, followed by a list of messages that the processor is required to send in the following round. Since each protocol is defined for systems of arbitrary size, it is natural to define a *class* of systems to be a collection of systems $\{\Sigma(n,t) \mid n \geq t+2\}$, where $\Sigma(n,t) = \langle n, t, \mathcal{P}, \mathcal{M}, \mathcal{I} \rangle$ for some fixed protocol $\mathcal{P}$, failure model $\mathcal{M}$, and input set $\mathcal{I}$.

In order to analyze systems, it is convenient to have a logical language in which we can make statements about the system. We will define such a language in Section 4. A *fact* (or *formula*) in this language will be interpreted as a property of points: a fact φ will be either true or false at a given point $\langle \rho, l \rangle$, which we denote by $\langle \rho, l \rangle \models \varphi$ or $\langle \rho, l \rangle \not\models \varphi$, respectively. Sometimes, however,

[2]It is common to assume, in addition, that p_i cannot perform an action after it has failed (i.e., once $S_i(k)$ is nonempty).

it is convenient to refer to facts as being about objects other than points (e.g., properties of runs). In general, a fact φ is said to be a *fact about* X if fixing X determines the truth (or falsity) of φ. For example, a fact φ is said to be a fact *about the nonfaulty processors* if fixing the identities of the nonfaulty processors determines whether nor not φ holds. That is, given any set $N \subseteq P$, either φ holds at all points $\langle \rho, l \rangle$ in which N is the set of nonfaulty processors, or at no such point. The meaning of a fact being *about the operating environment*, *about the initial states*, etc. are defined similarly.

3 Consistent Simultaneous Choices

Moses and Tuttle define a class of *simultaneous choice problems* in which nonfaulty processors are required to choose an action (such as deciding on an output bit) and perform it simultaneously. Their definition does not restrict the behavior of the faulty processors in any way. In this section, we define a class of *consistent simultaneous choice problems* (or simply *consistent choices*) in which faulty and nonfaulty processors must behave consistently: if a faulty processor performs any action at all in a run, it must perform the same action that the nonfaulty processors perform, and must perform it at the same time that the nonfaulty processors do. The definition of such a problem must tell us when each action may be performed, and describe the operating environments in which a choice among the actions must be made (that is, tell us what initial inputs and what types of processor failures are allowed).

Formally, a *simultaneous action* a_i is an action with an associated *enabling condition* ok_i, which is a fact about the input and the faulty processors. A *consistent simultaneous choice* problem $\mathcal{C}$ is determined by a set $\{a_1, \ldots, a_m\}$ of simultaneous actions (and their associated enabling conditions), together with a failure model $\mathcal{M}$ and set $\mathcal{I}$ of initial inputs. Intuitively, every run ρ of a protocol implementing $\mathcal{C}$ must satisfy the following conditions:

1. each processor performs at most one of the a_i's and does so at most once,

2. if a_i is performed by some processor at some time, then a_i is performed by all nonfaulty processors at that time,

3. if a_i is performed by some processor, then ρ satisfies ok_i, and

4. if no processors fail in ρ, then all processors perform an action.

More formally, a protocol $\mathcal{P}$ and a consistent simultaneous choice $\mathcal{C}$ *determine* a class of systems $\{\Sigma(n,t) \mid n \geq t+2\}$, where $\Sigma(n,t) = \langle n, t, \mathcal{P}, \mathcal{M}, \mathcal{I} \rangle$. $\mathcal{P}$ *implements* $\mathcal{C}$ if every run of every system in the class determined by $\mathcal{P}$ and $\mathcal{C}$ satisfies the conditions 1–4 above.[3] A choice $\mathcal{C}$ is *implementable* if there is some protocol that implements it.

One example of a consistent simultaneous choice problem is a consistent, simultaneous version of the *Byzantine Agreement* problem mentioned in the introduction (cf. [1,12]). Recall that each processor begins with an input bit and chooses an output bit, subject to the condition that all output bits have the same value and that this value is the value of some processor's input bit. This problem involves a choice between two actions: a_0 corresponds to choosing 0 as the output bit, and a_1 corresponds to choosing 1. Since the value of a processor's output bit must be the value of some

[3]This is analogous to the definition of a simultaneous choice of Moses and Tuttle [9]. The differences are the following: their definition places restrictions only on the nonfaulty processors; their enabling conditions are derived from *pro* and *con* conditions, which are facts about the input and the existence of failures (the results of Moses and Tuttle still hold if their definition of choices is extended to allow these conditions to be facts about the input and the faulty processors); and the liveness condition given here (that processors must act in failure-free runs) is a weak version of their notion of strictness. For a further discussion of this last point, see the discussion after the definition of an optimal protocol. We sometimes refer to a simultaneous choice of Moses and Tuttle as a *general simultaneous choice*.

processor's input bit, the enabling condition ok_0 for a_0 holds if and only if some processor's input bit is 0, and ok_1 is defined analogously.

This paper concerns optimal solutions to consistent simultaneous choice problems. A protocol $\mathcal{P}$ is an *optimal* protocol for a consistent simultaneous choice $\mathcal{C}$ if (i) $\mathcal{P}$ implements $\mathcal{C}$ and (ii) given any other protocol $\mathcal{P}'$ implementing $\mathcal{C}$, for every pair of corresponding runs of the two protocols, $\mathcal{P}$ has the nonfaulty processors perform actions no later than $\mathcal{P}'$.

Note that this is a very strong definition of optimality. Whereas most definitions of optimality require only that a protocol perform as well as any other protocol in their respective worst case runs, this definition requires that a protocol do so in *every* run. It is because we are interested in this strong definition of optimality that we can make the very weak liveness condition (condition 4) in the definition of a consistent choice. This condition says that processors need only perform an action in failure-free runs. One might want to require processors to perform an action in more runs, say in all runs. Note, however, that if there is any protocol for a problem in which the nonfaulty processors perform an action in every run, then there is an *optimal* protocol in which the nonfaulty processors perform an action in every run. Consequently, because the definition of an optimal protocol is so strong and the definition of a protocol solving a consistent choice is so weak, the existence of optimal protocols (which we prove in some cases) is a very strong result, as is the impossibility of any protocol at all (which we prove in another).

4 Definitions of Knowledge

The analysis in this paper depends on a processor's knowledge at a given point of an execution. This section defines such a notion of knowledge. For the sake of this section, fix a particular system. All runs mentioned will be runs of this system, and all points will be points in such runs. The treatment here is a modification and expansion of those given by Moses and Tuttle [9] and by others [1,5].

This definition of knowledge requires a logical language that can express statements about knowledge. Recall that a *fact* (or *formula*) in this language is a property of points: a fact φ is either true or false at a given point $\langle \rho, l \rangle$, denoted $\langle \rho, l \rangle \models \varphi$ or $\langle \rho, l \rangle \not\models \varphi$, respectively. A fact is said to be *valid in the system* (for a given system) if it is true of all points in the system. A fact is said to be *valid* if it is valid in all systems. Assume the existence of an underlying logical language for representing all relevant *ground* facts—facts about the system that do not explicitly mention processors' knowledge (for example, "*the value of register x is* 0" or "*processor p_i failed in round* 3"). Formally, a ground fact φ will be identified with a set of points $\tau(\varphi)$. Intuitively, this is the set of points at which the fact holds. The truth of a ground fact φ is determined by $\langle \rho, l \rangle \models \varphi$ if and only if $\langle \rho, l \rangle \in \tau(\varphi)$. This language is then closed under the standard boolean connectives $\wedge$ and $\neg$, as well as various knowledge operators. The meaning of the boolean connectives is defined in the usual way. The meaning of the knowledge operators is the subject of the rest of this section.

The intuition underlying the standard definition of knowledge [5] is that, in any given global state, a processor considers a number of other global states to be *possible*, namely those global states in which it has the same local state; a processor knows a fact φ if and only if φ holds in all global states it considers possible. Formally, p_i considers a point $\langle \rho', l' \rangle$ to be *possible* at $\langle \rho, l \rangle$ if and only if $v(p_i, \rho, l) = v(p_i, \rho', l')$.[4] We say that p_i *knows* φ at $\langle \rho, l \rangle$, denoted $\langle \rho, l \rangle \models \mathsf{K}_i\varphi$, if and only if $\langle \rho', l' \rangle \models \varphi$ for all points $\langle \rho', l' \rangle$ that p_i considers possible at $\langle \rho, l \rangle$. Thus, a processor knows any fact that follows from the information recorded in its local state. In particular, it knows any such fact regardless of the complexity of computing that it follows from its local state.

In addition to the knowledge of individual processors, our analysis depends on knowledge among a group of processors. In particular, the state of *common knowledge* [5] plays a central role in this paper due to the close relationship between it and the simultaneous performance of an action by a group of processors (see Lemma 5). Roughly speaking, a fact φ is common knowledge to a given group if everyone in the group knows φ, everyone in the group knows that everyone knows φ, and

[4]Note that this implies $l = l'$ because the global clock is part of p_i's view.

so on *ad infinitum.* Given a fixed group G of processors, it is customary to define "*everyone in G knows φ*," denoted $\mathsf{E}_G\varphi$, by

$$\mathsf{E}_G\varphi \overset{\text{def}}{=} \bigwedge_{p_i \in G} \mathsf{K}_i\varphi.$$

One now defines "*φ is common knowledge to G*," denoted $\mathsf{C}_G\varphi$, by

$$\mathsf{C}_G\varphi \overset{\text{def}}{=} \mathsf{E}_G\varphi \wedge \mathsf{E}_G\mathsf{E}_G\varphi \wedge \cdots \wedge \mathsf{E}_G^m\varphi \wedge \cdots.$$

In other words, $\langle\rho, l\rangle \models \mathsf{C}_G\varphi$ if and only if $\langle\rho, l\rangle \models \mathsf{E}_G^m\varphi$ for all $m \geq 1$, where $\mathsf{E}_G^1\varphi \overset{\text{def}}{=} \mathsf{E}_G\varphi$ and $\mathsf{E}_G^{m+1}\varphi \overset{\text{def}}{=} \mathsf{E}_G(\mathsf{E}_G^m\varphi)$.

In systems of unreliable processors, however, the groups of interest are not always fixed subsets of P like G above. In this paper, the most interesting facts will be those that are common knowledge to the set $\mathcal{N}$ of nonfaulty processors, and the value of this set changes from run to run. We use $\mathcal{N}(\rho, l)$ to denote the set of nonfaulty processors at the point $\langle\rho, l\rangle$ (and therefore in the run ρ). Moses and Tuttle argue that the generalization of common knowledge to nonconstant sets such as $\mathcal{N}$ is more subtle than simply defining

$$\mathsf{E}_{\mathcal{N}}\varphi \overset{\text{def}}{=} \bigwedge_{p_i \in \mathcal{N}} \mathsf{K}_i\varphi,$$

and they propose a generalization of $\mathsf{E}_{\mathcal{N}}\varphi$, which we denote by $\mathsf{F}_{\mathcal{N}}\varphi$ (although they continue to write $\mathsf{E}_{\mathcal{N}}\varphi$). They argue that one must define

$$\mathsf{F}_{\mathcal{N}}\varphi \overset{\text{def}}{=} \bigwedge_{p_i \in \mathcal{N}} \mathsf{K}_i(p_i \in \mathcal{N} \Rightarrow \varphi).$$

That is, every nonfaulty processor knows φ if and only if every nonfaulty processor knows that φ is true, given that it (the processor itself) is nonfaulty. Since it is possible for p_i to be contained in $\mathcal{N}$ without knowing this, the definition of $\mathsf{F}_{\mathcal{N}}\varphi$ is weaker than $\mathsf{E}_{\mathcal{N}}\varphi$, and the definition of common knowledge based on $\mathsf{F}_{\mathcal{N}}\varphi$ is therefore weaker than the definition above based on $\mathsf{E}_{\mathcal{N}}\varphi$. For this reason, we distinguish these two definitions of common knowledge as *weak* and *strong common knowledge*, and we denote weak and strong common knowledge of φ by $\mathsf{W}_{\mathcal{N}}\varphi$ and $\mathsf{S}_{\mathcal{N}}\varphi$, respectively. (Moses and Tuttle write $\mathsf{C}_{\mathcal{N}}\varphi$ for $\mathsf{W}_{\mathcal{N}}\varphi$.) Formally,

$$\mathsf{W}_{\mathcal{N}}\varphi \overset{\text{def}}{=} \mathsf{F}_{\mathcal{N}}\varphi \wedge \mathsf{F}_{\mathcal{N}}\mathsf{F}_{\mathcal{N}}\varphi \wedge \cdots \wedge \mathsf{F}_{\mathcal{N}}^m\varphi \wedge \cdots$$

and, as above,

$$\mathsf{S}_{\mathcal{N}}\varphi \overset{\text{def}}{=} \mathsf{E}_{\mathcal{N}}\varphi \wedge \mathsf{E}_{\mathcal{N}}\mathsf{E}_{\mathcal{N}}\varphi \wedge \cdots \wedge \mathsf{E}_{\mathcal{N}}^m\varphi \wedge \cdots.$$

Moses and Tuttle prove that weak common knowledge is a necessary and sufficient condition for the performance of simultaneous actions. In this work, we prove that strong common knowledge is a necessary and sufficient condition for the performance of *consistent* simultaneous actions. This is somewhat surprising, since it says that the solution of the harder problem depends on the simpler, less subtle definition of common knowledge. On the other hand, we will show that, in most of the cases considered by Moses and Tuttle, the two definitions of common knowledge are actually equivalent. In one case where the two definitions are different, we will show that consistent performance of simultaneous actions is impossible. Thus, this work shows precisely where the added subtlety in the definition of weak common knowledge is required for the results of Moses and Tuttle. Understanding this relationship between strong and weak common knowledge is one of the primary technical contributions of this work.

This section concludes with a few observations about properties of the two definitions of common knowledge [5]. A useful tool for thinking about $\mathsf{F}_{\mathcal{N}}^m\varphi$ and $\mathsf{W}_{\mathcal{N}}\varphi$ is a graph whose nodes are all points of the system and in which there is an edge between points $\langle\rho, l\rangle$ and $\langle\rho', l\rangle$ if and only if there is some processor $p \in \mathcal{N}(\rho, l) \cap \mathcal{N}(\rho', l)$ such that $v(p, \rho, l) = v(p, \rho', l)$. That is, there is a processor that is nonfaulty at both points and that has the same view at both points. This graph is called the *similarity graph*. An easy argument by induction on m shows that $\langle \rho, l\rangle \models \mathsf{F}^m_{\mathcal{N}}\varphi$ if and only if $\langle \rho', l\rangle \models \varphi$ holds for all points $\langle \rho', l\rangle$ at a distance m from $\langle \rho, l\rangle$ in this graph. Two points $\langle \rho, l\rangle$ and $\langle \rho', l\rangle$ are *similar*, denoted $\langle \rho, l\rangle \sim \langle \rho', l\rangle$, if they are in the same connected component of the similarity graph. It is easy to see that the following lemma holds:

Lemma 1: *$\langle \rho, l\rangle \models \mathsf{W}_{\mathcal{N}}\varphi$ if and only if $\langle \rho', l\rangle \models \varphi$ for all $\langle \rho', l\rangle$ satisfying $\langle \rho, l\rangle \sim \langle \rho', l\rangle$.*

Now consider a directed graph whose nodes are all points of the system and in which there is an edge from $\langle \rho, l\rangle$ to $\langle \rho', l\rangle$ if and only if there is some processor $p \in \mathcal{N}(\rho, l)$ such that $v(p, \rho, l) = v(p, \rho', l)$. This means that there is a processor that is nonfaulty at $\langle \rho, l\rangle$ (but possibly faulty at $\langle \rho', l\rangle$) and that has the same view at both points. This graph is called the *directed similarity graph*. An easy argument by induction on m shows that $\langle \rho, l\rangle \models \mathsf{E}^m_{\mathcal{N}}\varphi$ if and only if $\langle \rho', l\rangle \models \varphi$ holds for all points $\langle \rho', l\rangle$ to which there is a path from $\langle \rho, l\rangle$ of length m in this graph. Let $\langle \rho, l\rangle \rightsquigarrow \langle \rho', l\rangle$ denote the fact that $\langle \rho', l\rangle$ is reachable from $\langle \rho, l\rangle$ in the directed similarity graph. The following lemma holds for strong common knowledge:

Lemma 2: *$\langle \rho, l\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$ if and only if $\langle \rho', l\rangle \models \varphi$ for all $\langle \rho', l\rangle$ satisfying $\langle \rho, l\rangle \rightsquigarrow \langle \rho', l\rangle$.*

Note that $\langle \rho, l\rangle \sim \langle \rho', l\rangle$ implies both $\langle \rho, l\rangle \rightsquigarrow \langle \rho', l\rangle$ and $\langle \rho', l\rangle \rightsquigarrow \langle \rho, l\rangle$; the converse does not always hold. It follows that $\mathsf{S}_{\mathcal{N}}\varphi \Rightarrow \mathsf{W}_{\mathcal{N}}\varphi$ is valid for all facts φ. (As previously noted, this also follows from the fact that $\mathsf{E}_{\mathcal{N}}\varphi \Rightarrow \mathsf{F}_{\mathcal{N}}\varphi$ is valid for all φ.)

The notions of knowledge and common knowledge defined above are closely related to modal logics [4]. Five properties of a modal operator M are

A1: the *knowledge axiom*: $\mathsf{M}\varphi \Rightarrow \varphi$;

A2: the *consequence closure axiom*: $\mathsf{M}\varphi \wedge \mathsf{M}(\varphi \Rightarrow \psi) \Rightarrow \mathsf{M}\psi$;

A3: the *positive introspection axiom*: $\mathsf{M}\varphi \Rightarrow \mathsf{M}\mathsf{M}\varphi$;

A4: the *negative introspection axiom*: $\neg\mathsf{M}\varphi \Rightarrow \mathsf{M}\neg\mathsf{M}\varphi$; and

R1: the *rule of necessitation*: if φ is valid in the system, then $\mathsf{M}\varphi$ is valid in the system.

A modal operator M *has the properties of the modal logic S5* in a system if the axioms A1–A4 are valid in the system and R1 holds. M *has the properties of the modal logic S4* in a system if the axioms A1–A3 are valid in the system and R1 holds.

It is easy to see that the knowledge and weak common knowledge operators defined above have the properties of S5 in all systems [4]:

Theorem 3: *The operators K_i and $\mathsf{W}_{\mathcal{N}}$ have the properties of S5 in all systems.*

Strong common knowledge can only be shown to satisfy S4:

Theorem 4: *The operator $\mathsf{S}_{\mathcal{N}}$ has the properties of S4 in all systems.*

Proof: A1 holds because the relation $\rightsquigarrow$ is reflexive. A2 holds because if $\varphi \Rightarrow \psi$ and φ both hold at all points $\langle \rho', l\rangle$ such that $\langle \rho, l\rangle \rightsquigarrow \langle \rho', l\rangle$, then so does ψ. A3 holds because the relation $\rightsquigarrow$ is transitive. It is obvious that R1 holds. Thus, $\mathsf{S}_{\mathcal{N}}$ has the properties of S4. $\square$

$\mathsf{S}_{\mathcal{N}}$ does not have the properties of S5 because the axiom A4 need not hold for $\mathsf{S}_{\mathcal{N}}$: it does not hold if the relation $\rightsquigarrow$ is not symmetric [4]. Section 6 below considers a specific system in which this occurs.

Strong and weak common knowledge also satisfy the following *fixed-point axioms*:

$$\mathsf{W}_{\mathcal{N}}\varphi \Leftrightarrow \mathsf{F}_{\mathcal{N}}\mathsf{W}_{\mathcal{N}}\varphi;$$

$$\mathsf{S}_{\mathcal{N}}\varphi \Leftrightarrow \mathsf{E}_{\mathcal{N}}\mathsf{S}_{\mathcal{N}}\varphi.$$

These axioms imply that a fact's being common knowledge is "public," in the sense that a fact is common knowledge if and only if everyone knows it is common knowledge. It is these two axioms, which imply that all processors in a group "learn" of common knowledge simultaneously, that make common knowledge so important for implementing simultaneous choices. Another useful fact about common knowledge is captured by the following *induction rules*:

- if $\varphi \Rightarrow \mathsf{F}_{\mathcal{N}}(\varphi \wedge \psi)$ is valid in the system, then $\varphi \Rightarrow \mathsf{W}_{\mathcal{N}}\psi$ is valid in the system;
- if $\varphi \Rightarrow \mathsf{E}_{\mathcal{N}}(\varphi \wedge \psi)$ is valid in the system, then $\varphi \Rightarrow \mathsf{S}_{\mathcal{N}}\psi$ is valid in the system.

Intuitively, taking ψ to be φ for the moment, these rules say that if φ is "public" in the sense that everyone knows φ whenever φ holds, then φ is actually common knowledge whenever φ holds. When ψ and φ are different, these rules say that if, in addition, φ implies ψ, then ψ itself is also common knowledge.

5 Optimal Protocols

In this section, we derive optimal protocols for consistent simultaneous choice problems in the crash, send omissions, and general omissions models. Previously, Dwork and Moses [1] derived optimal protocols for *Simultaneous Byzantine Agreement* in the crash failure model, and Moses and Tuttle [9] derived optimal protocols for simultaneous choice problems in the crash, send, and general omissions failure models. The derivation of these protocols is the result of an analysis of simultaneous choice problems in terms of weak common knowledge. In this section, we show how a similar analysis of *consistent* simultaneous choice problems in terms of *strong* common knowledge leads to optimal protocols for these problems as well.

The fundamental observation leading to the earlier derivation of optimal protocols is the strong relationship between common knowledge and simultaneous actions: when a simultaneous action is performed, it is weak common knowledge that this action is being performed and, thus, that it is enabled. The following lemma shows that an analogous result holds for consistent simultaneous actions when phrased in terms of strong common knowledge. This is a stronger statement that does not hold in general for simultaneous choice problems.

Lemma 5: *Let ρ be a run of a protocol implementing a consistent simultaneous choice $\mathcal{C}$. If an action a_j of $\mathcal{C}$ is performed by any processor at time l in ρ, then $\langle \rho, l\rangle \models \mathsf{S}_{\mathcal{N}}ok_j$.*

Proof: Let φ be the fact "a_j is being performed by some processor." Note that $\langle \rho, l\rangle \models \varphi$. We now show that $\varphi \Rightarrow \mathsf{E}_{\mathcal{N}}(\varphi \wedge ok_j)$ is valid in the system. It will follow by the induction rule that $\varphi \Rightarrow \mathsf{S}_{\mathcal{N}}ok_j$ is also valid in the system and, thus, that $\langle \rho, l\rangle \models \mathsf{S}_{\mathcal{N}}ok_j$. Let $\langle \eta, l\rangle$ be any point such that $\langle \eta, l\rangle \models \varphi$. By condition 2 of the definition of a consistent choice, all processors in $\mathcal{N}(\eta, l)$ execute a_j at time l in η, so $\langle \eta, l\rangle \models \mathsf{E}_{\mathcal{N}}\varphi$ (every processor performing a_j certainly knows a_j is being performed). By condition 3, $\varphi \Rightarrow ok_j$ is valid in the system, so $\langle \eta, l\rangle \models \mathsf{E}_{\mathcal{N}}ok_j$ by the consequence closure axiom A2. Thus, $\langle \eta, l\rangle \models \mathsf{E}_{\mathcal{N}}(\varphi \wedge ok_j)$. Since $\langle \eta, l\rangle$ was chosen arbitrarily, $\varphi \Rightarrow \mathsf{E}_{\mathcal{N}}(\varphi \wedge ok_j)$ is valid in the system. $\square$

Lemma 5 states that no action can be performed until that action's enabling condition is strong common knowledge. Intuitively, any protocol that has processors act as soon as some enabling condition becomes strong common knowledge will have them act at least as quickly as any other protocol and should be an optimal protocol. In this sense, the derivation of optimal protocols reduces to the derivation of protocols that cause facts to become strong common knowledge as soon as possible. Both Dwork and Moses and Moses and Tuttle follow this strategy (using weak common knowledge). One class of protocols causing facts to become strong common knowledge as soon as possible are *full-information protocols*.

5.1 Full-Information Protocols

Recall that a protocol is a function specifying the actions a processor should perform and the messages that it should send as a function of n, t, and the processor's view. Thus, a protocol has two components: an *action* component and a *message* component. A protocol is said to be a *full-information protocol* [3] if its message component calls for it to send its entire view to all processors in every round. Since such a protocol requires that all processors send all the information available to them in every round, it gives each processor as much information about the operating environment as any protocol could. Consequently, if a processor cannot distinguish two operating environments during runs of a full-information protocol, then the processor cannot distinguish them during runs of any other protocol:

Lemma 6 (Moses and Tuttle): *Let τ and τ' be runs of a full-information protocol $\mathcal{F}$, and let ρ and ρ' be runs of an arbitrary protocol $\mathcal{P}$ corresponding to τ and τ', respectively. For all processors p and times l, if $v(p,\tau,l) = v(p,\tau',l)$, then $v(p,\rho,l) = v(p,\rho',l)$.*

The following corollary to Lemma 6 shows that facts about the operating environment become strong common knowledge during runs of full-information protocols at least as soon as they do during runs of any other protocol. Moses and Tuttle state a similar corollary that says that the same is true of weak common knowledge.

Corollary 7: *Let φ be a fact about the operating environment. Let τ and ρ be corresponding runs of a full-information protocol $\mathcal{F}$ and an arbitrary protocol $\mathcal{P}$, respectively. If $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$, then $\langle \tau, l\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$.*

Proof: Suppose $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$. To prove that $\langle \tau, l\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$, it suffices to show that $\langle \tau', l\rangle \models \varphi$ for all runs τ' of $\mathcal{F}$ such that $\langle \tau, l\rangle \rightsquigarrow \langle \tau', l\rangle$. Fix τ' and let ρ' be the corresponding run of $\mathcal{P}$. Lemma 6 and a simple inductive argument on the distance from $\langle \tau, l\rangle$ to $\langle \tau', l\rangle$ in the similarity graph show that $\langle \tau, l\rangle \rightsquigarrow \langle \tau', l\rangle$ implies $\langle \rho, k\rangle \rightsquigarrow \langle \rho', k\rangle$. Since $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$, we have $\langle \rho', k\rangle \models \varphi$. Since φ is a fact about the operating environment and τ' and ρ' have the same operating environment, $\langle \tau', l\rangle \models \varphi$, as desired. $\square$

Because the enabling conditions for simultaneous actions are facts about the operating environment, the remainder of the paper concentrates on full-information protocols, which achieve common knowledge of such facts as quickly as possible.

5.2 Crash and Send Omissions

Using their version of Corollary 7 (with $\mathsf{S}_{\mathcal{N}}$ replaced by $\mathsf{W}_{\mathcal{N}}$), Moses and Tuttle derive a full-information, optimal protocol $\mathcal{F}_{MT}$ for any implementable simultaneous choice $\mathcal{C}$ (see Figure 1). The protocol is *knowledge-based* since it is programmed in a language making explicit tests for common knowledge of various facts. In this protocol, each processor broadcasts its view every round until it detects that one of the enabling actions ok_j has become weak common knowledge to the nonfaulty processors. At this point, the processor performs the action a_j, where j is the least index such that ok_j is weak common knowledge. While broadcasting views may in general require sending exponentially large messages, Moses and Tuttle show that, in any of the (benign) failure models considered here, a processor's view can be encoded as a *communication graph* whose size is polynomial in n and the current round number; thus, this protocol requires only polynomially large messages.

In order to implement $\mathcal{F}_{MT}$, it is necessary to implement the tests for weak common knowledge that appear in $\mathcal{F}_{MT}$. Because processors have to be able to test for $\mathsf{W}_{\mathcal{N}} ok_j$ locally, any test for $\mathsf{W}_{\mathcal{N}} ok_j$ must determine whether $\mathsf{W}_{\mathcal{N}} ok_j$ holds at a point given only the view of a processor at that point. Furthermore, this must be true of every point in every system $\Sigma(n,t)$ determined by $\mathcal{F}_{MT}$ and the simultaneous choice $\mathcal{C}$ that $\mathcal{F}_{MT}$ solves. On the other hand, because the simultaneous choice $\mathcal{C}$ restricts only the performance of actions by *nonfaulty* processors, this test need only be correct

```
repeat every round
    broadcast view to every processor
until $\mathsf{W}_{\mathcal{N}} ok_j$ holds for some $j$;
$j \leftarrow \min\{k \mid \mathsf{W}_{\mathcal{N}} ok_k \text{ holds}\}$;
perform $a_j$;
halt.
```

Figure 1: The optimal protocol $\mathcal{F}_{MT}$

when given the view of a nonfaulty processor. Formally, given any fact ψ and any set S, a *test for* ψ *for* S within a class of systems $\{\Sigma(n,t) \mid n \geq t+2\}$ is a Turing machine M that

1. accepts as input the view of any processor at any point in any system $\Sigma(n,t)$ and returns either *true* or *false*, and
2. given the view $v(p_i, \rho, l)$ of any processor p_i in S at a point $\langle \rho, l \rangle$, returns *true* if and only if $\langle \rho, l \rangle \models \psi$.

Moses and Tuttle construct tests for $\mathsf{W}_{\mathcal{N}}\varphi$ for the set $\mathcal{N}$ of nonfaulty processors, and make a rather surprising technical observation implying that, in the crash and send omissions models, even the faulty processors can use these tests: given the view of *any* processor p_i (faulty or nonfaulty) at $\langle \rho, l \rangle$, their test for $\mathsf{W}_{\mathcal{N}}\varphi$ returns *true* if and only if $\langle \rho, l \rangle \models \mathsf{W}_{\mathcal{N}}\varphi$. That is, their test is a test for $\mathsf{W}_{\mathcal{N}}\varphi$ for the set P of all processors. We refer to such a test as, simply, a *test for* $\mathsf{W}_{\mathcal{N}}\varphi$. Thus, Moses and Tuttle effectively show that

Theorem 8 (Moses and Tuttle): *If $\mathcal{C}$ is an implementable, consistent simultaneous choice, then $\mathcal{F}_{MT}$ is an optimal protocol for $\mathcal{C}$.*

Proof: Let $\mathcal{P}$ be any protocol implementing $\mathcal{C}$, and let τ and ρ be corresponding runs of $\mathcal{F}_{MT}$ and $\mathcal{P}$. First note that nonfaulty processors perform actions in τ at least as early as any processor does so in ρ: if any processor performs a_j at time l in ρ, then $\mathsf{S}_{\mathcal{N}} ok_j$ holds at $\langle \rho, l \rangle$ by Lemma 5 and also at $\langle \tau, l \rangle$ by Corollary 7; since $\mathsf{S}_{\mathcal{N}} ok_j \Rightarrow \mathsf{W}_{\mathcal{N}} ok_j$ is valid, all nonfaulty processors perform an action at time l in τ if they have not already done so. To see that $\mathcal{F}_{MT}$ actually implements $\mathcal{C}$, note the following:

1. Each processor clearly performs at most one of the a_j in τ and does so at most once.
2. Suppose a_j is performed by some processor p at time l in τ. Recall that every processor is following a test M_k for $\mathsf{W}_{\mathcal{N}} ok_k$ to determine whether it should perform a_k; recall that M_k is actually a test for $\mathsf{W}_{\mathcal{N}} ok_k$ for all processors. Since p performed a_j at time l, it follows that l is the first round in which some ok_k is weak common knowledge, and that j is the least index k such that ok_k is weak common knowledge in round l. Thus, in particular, every nonfaulty processor will perform a_j at time l in τ.
3. Suppose a_j is performed by some processor p in τ. If p performs a_j at time l in τ, then $\mathsf{W}_{\mathcal{N}} ok_j$ holds at time l in τ, so τ satisfies ok_j (since ok_j is a fact about the run).
4. Suppose no processor fails in τ. Then the same is true in the corresponding run ρ of the protocol $\mathcal{P}$ implementing $\mathcal{C}$, so some action must be performed at some time l in ρ. Thus, some action must be performed by the nonfaulty processors no later than time l in τ by the argument above. Since all processors are nonfaulty in τ, all processors perform this action at time l.

Thus, $\mathcal{F}_{MT}$ is an optimal protocol for $\mathcal{C}$. □

To analyze the computational complexity of $\mathcal{F}_{MT}$, one must understand the complexity the tests M_j for $\mathsf{W}_{\mathcal{N}} ok_j$. To do so, one must also understand the basic complexity of determining the truth of the facts ok_j. Moses and Tuttle define the class of *practical* simultaneous choice problems to be a restriction of the class of simultaneous choice problems that, among other things, guarantees that it is possible to test in polynomial time, given the view of a processor at a point, whether this processor's view determines that ok_j must be true (or, in other words, whether this processor knows ok_j). Every natural simultaneous choice problem of which we are aware is practical, but we refer the reader to Moses and Tuttle [9] for the detailed definition. For such problems, they show that their tests for $\mathsf{W}_{\mathcal{N}} ok_j$ at a point $\langle \rho, l\rangle$ run in time polynomial in n and l in both the crash and send omissions model. Thus, Moses and Tuttle effectively show that

Theorem 9 (Moses and Tuttle): *If $\mathcal{C}$ is an implementable, practical, consistent simultaneous choice, then $\mathcal{F}_{MT}$ is a polynomial-time, optimal protocol for $\mathcal{C}$.*

5.3 General Omissions with $n > 2t$

According to Lemma 5, strong common knowledge is a necessary condition for consistently performing simultaneous actions, and yet the protocol $\mathcal{F}_{MT}$ is programmed using tests for weak common knowledge. More precisely, the technical observation made by Moses and Tuttle leading to the observation that both faulty and nonfaulty processors can follow their tests for $\mathsf{W}_{\mathcal{N}}\varphi$ also implies that weak common knowledge to the nonfaulty processors is equivalent to weak common knowledge to *all* processors: that is, $\mathsf{W}_{\mathcal{N}}\varphi \equiv \mathsf{W}_{P}\varphi$ for all facts φ. From this it follows that $\mathsf{W}_{\mathcal{N}}\varphi \Rightarrow \mathsf{S}_{\mathcal{N}}\varphi$, since $\mathsf{W}_{P}\varphi$ clearly implies $\mathsf{S}_{\mathcal{N}}\varphi$; we hence have $\mathsf{W}_{\mathcal{N}}\varphi \equiv \mathsf{S}_{\mathcal{N}}\varphi$, since we already observed that $\mathsf{S}_{\mathcal{N}}\varphi \Rightarrow \mathsf{W}_{\mathcal{N}}\varphi$.

It turns out that the same is true in the general omissions model, but only when $n > 2t$. To prove this fact, it is enough to show that the relations $\sim$ and $\rightsquigarrow$ are the same. Remember that when we make a statement like "ρ and ρ' differ only in that every processor sends a message to p_i in round l of ρ'," we mean that the operating environments of ρ and ρ' are identical except that the operating environment of ρ' does not itself keep processors from sending messages to p_i in round l.

Lemma 10: *Consider any system in the general omissions model with $n > 2t$. If $\langle \rho, l\rangle$ and $\langle \eta, l\rangle$ are two points of the system, then $\langle \rho, l\rangle \rightsquigarrow \langle \eta, l\rangle$ if and only if $\langle \rho, l\rangle \sim \langle \eta, l\rangle$.*

Proof: Since $\langle \rho, l\rangle \sim \langle \eta, l\rangle$ clearly implies $\langle \rho, l\rangle \rightsquigarrow \langle \eta, l\rangle$, we prove that $\langle \rho, l\rangle \rightsquigarrow \langle \eta, l\rangle$ implies $\langle \rho, l\rangle \sim \langle \eta, l\rangle$. It suffices to show that if $v(p,\rho,l) = v(p,\eta,l)$ for some $p \in \mathcal{N}(\rho, l)$, then $\langle \rho, l\rangle \sim \langle \eta, l\rangle$.

Suppose $v(p,\rho,l) = v(p,\eta,l)$ for some $p \in \mathcal{N}(\rho,l)$. Let $A = \mathcal{N}(\rho,l)$ and $B = \mathcal{N}(\eta,l)$. If $p \in B$, then p is nonfaulty at both points and, thus, $\langle \rho, l\rangle \sim \langle \eta, l\rangle$; suppose instead $p \in A - B$. Let $N = A \cap B$ and $F = P - (A \cup B)$; note that N is the set of processors nonfaulty in both runs and that F is the set of processors faulty in both runs. Furthermore, note that the sets $A - B$, $B - A$, N, and F partition of the set P of processors. Since all processors faulty in one or both runs are in $(A - B) \cup (B - A) \cup F$, we have $|(A - B) \cup (B - A) \cup F| \leq 2t$. Since $n > 2t$, we have $N \neq \emptyset$. Let q be a processor in N.

Let $\hat{\rho}$ and $\hat{\eta}$ be runs differing from ρ and η, respectively, only in that processors in $P - \{p\}$ receive messages from all processors in round l of $\hat{\rho}$ and $\hat{\eta}$ (and that all processors receive all messages after time l in $\hat{\rho}$ and $\hat{\eta}$). Note that A is the set of nonfaulty processors in ρ and $\hat{\rho}$ and that B is the set of nonfaulty processors in η and $\hat{\eta}$. Note also that p has the same view at time l in the four runs ρ, $\hat{\rho}$, $\hat{\eta}$, and η. It is clear that $\langle \rho, l\rangle \sim \langle \hat{\rho}, l\rangle$ since p is nonfaulty in both runs and has the same view at both points. To see that $\langle \eta, l\rangle \sim \langle \hat{\eta}, l\rangle$, let η' be a run differing from η only in that processors in $P - \{p, q\}$ receive messages from all processors in round l of η' (and assume that all processors receive all messages after time l in η'). Note that $\langle \eta, l\rangle \sim \langle \eta', l\rangle$ since q is nonfaulty in both runs and has the same view at both points. Similarly, note that $\langle \eta', l\rangle \sim \langle \hat{\eta}, l\rangle$ since there is a processor r distinct from p and q that is nonfaulty in both runs (r must exist since we always assume $n \geq t + 2$) and has the same view at both points. Thus, $\langle \eta, l\rangle \sim \langle \hat{\eta}, l\rangle$.

We claim that the only missing (omitted) messages in $\hat{\eta}$ are between the sets $A - B$ and $B - A$ and between the sets P and F. To prove this, it suffices to show that there are no missing messages among processors in A or among processors in B. First, since $B = \mathcal{N}(\hat{\eta}, l)$ is the set of nonfaulty processors in $\hat{\eta}$, there are certainly no missing messages between processors in B in $\hat{\eta}$. Second, since $A = \mathcal{N}(\hat{\rho}, l)$ is the set of nonfaulty processors in $\hat{\rho}$, there are no missing messages between processors in A in $\hat{\rho}$. In particular, p's view at $\langle\hat{\rho}, l\rangle$ records the fact that there are no missing messages between processors in A for the first $l - 1$ rounds of $\hat{\rho}$ and that there are no missing messages from processors in A to p in round l. Since p has the same view at $\langle\hat{\eta}, l\rangle$, the same is true of $\hat{\eta}$. Since, furthermore, all processors in $P - \{p\}$ receive messages from all processors in round l of $\hat{\eta}$ (and in particular from processors in A), there are no missing messages between processors in A in $\hat{\eta}$.

Since the only missing messages in $\hat{\eta}$ are between the sets $A - B$ and $B - A$ and between the sets P and F, it is consistent with this failure pattern that the faulty processors in $\hat{\eta}$ are the processors in $P - A = (B - A) \cup F$ (and not the processors in $P - B = (A - B) \cup F$ as is actually the case in $\hat{\eta}$). Let $\hat{\eta}'$ be the run identical to $\hat{\eta}$, except that $\mathcal{N}(\hat{\eta}', l) = A$ instead of $\mathcal{N}(\hat{\eta}, l) = B$. Since q is nonfaulty in both runs $\hat{\eta}$ and $\hat{\eta}'$—note that $q \in N = A \cap B = \mathcal{N}(\hat{\eta}, l) \cap \mathcal{N}(\hat{\eta}', l)$—and certainly has the same view at both points, $\langle\hat{\eta}, l\rangle \sim \langle\hat{\eta}', l\rangle$. Note, furthermore, that p is nonfaulty in both $\hat{\rho}$ and $\hat{\eta}'$ since $p \in A$, and that p also has the same view at $\langle\hat{\eta}', l\rangle$ and $\langle\hat{\rho}, l\rangle$, so $\langle\hat{\eta}', l\rangle \sim \langle\hat{\rho}, l\rangle$. By the transitivity of the similarity relation, therefore, we have $\langle\rho, l\rangle \sim \langle\eta, l\rangle$. □

It follows from Lemma 10 that strong and weak common knowledge are equivalent with respect to the nonfaulty processors:

Theorem 11: *Consider any system in the general omissions model with $n > 2t$. The formula $\mathsf{W}_{\mathcal{N}}\varphi \Leftrightarrow \mathsf{S}_{\mathcal{N}}\varphi$ is valid in the system for every fact φ.*

Proof: It suffices to show that $\mathsf{W}_{\mathcal{N}}\varphi \Rightarrow \mathsf{S}_{\mathcal{N}}\varphi$ is valid in the system, since we already noted in Section 4 that $\mathsf{S}_{\mathcal{N}}\varphi \Rightarrow \mathsf{W}_{\mathcal{N}}\varphi$ is valid. Suppose $\langle\rho, l\rangle \models \mathsf{W}_{\mathcal{N}}\varphi$. Lemma 2 implies that to prove that $\langle\rho, l\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$ we need only show that $\langle\rho', l\rangle \models \varphi$ for all $\langle\rho', l\rangle$ such that $\langle\rho, l\rangle \rightsquigarrow \langle\rho', l\rangle$. Since $\langle\rho, l\rangle \rightsquigarrow \langle\rho', l\rangle$ implies $\langle\rho, l\rangle \sim \langle\rho', l\rangle$ by Lemma 10, and since $\langle\rho, l\rangle \models \mathsf{W}_{\mathcal{N}}\varphi$ implies $\langle\rho', l\rangle \models \varphi$ for all $\langle\rho', l\rangle$ such that $\langle\rho, l\rangle \sim \langle\rho', l\rangle$ by Lemma 1, we have $\langle\rho', l\rangle \models \varphi$ for all $\langle\rho', l\rangle$ such that $\langle\rho, l\rangle \rightsquigarrow \langle\rho', l\rangle$. Thus, $\langle\rho, l\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$. □

Moses and Tuttle prove that $\mathcal{F}_{MT}$ is an optimal protocol for simultaneous choice problems in the general omissions model, and they show how to implement tests for $\mathsf{W}_{\mathcal{N}}\varphi$ for the nonfaulty processors in polynomial space, although they are not able to do so in polynomial time. The fact that weak and strong common knowledge are equivalent (when $n > 2t$) suggests that $\mathcal{F}_{MT}$ might also be an optimal protocol for *consistent* simultaneous choice problems in the general omissions model, as well as in the crash and send omissions models. But while they show that the nonfaulty processors can test for common knowledge of ok_j, it is not immediately clear that faulty processors can do the same. This raises two possible problem: (i) that the nonfaulty processors will know ok_1 and ok_2 are common knowledge while the faulty processors know only that ok_2 is common knowledge, resulting in the nonfaulty processors performing a_1 and the faulty processors performing a_2, and (ii) that the nonfaulty processors will know ok_1 is common knowledge at time k while the faulty processors will not know ok_1 is common knowledge until time $k+1$, resulting in the nonfaulty processors performing a_1 at time k and the faulty ones at time $k + 1$. Fortunately, we can use Lemma 10 again to prove that every processor—faulty or nonfaulty—knows the truth of $\mathsf{S}_{\mathcal{N}}\varphi$, provided it does not know it is faulty (that is, its local state does not prove it must be faulty):

Lemma 12: *Consider any system in the general omissions model with $n > 2t$. The formula $\neg\mathsf{K}_i(p_i \text{ is faulty}) \Rightarrow \mathsf{K}_i(\mathsf{S}_{\mathcal{N}}\varphi) \vee \mathsf{K}_i(\neg\mathsf{S}_{\mathcal{N}}\varphi)$ is valid in the system for every processor p_i and every fact φ.*

Proof: Suppose $\langle\rho, k\rangle \models \neg\mathsf{K}_i(p_i \text{ is faulty})$. We prove that $\langle\rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$ implies $\langle\rho, k\rangle \models \mathsf{K}_i(\mathsf{S}_{\mathcal{N}}\varphi)$, and an analogous argument proves that $\langle\rho, k\rangle \models \neg\mathsf{S}_{\mathcal{N}}\varphi$ implies $\langle\rho, k\rangle \models \mathsf{K}_i(\neg\mathsf{S}_{\mathcal{N}}\varphi)$.

```
$nonfaulty \leftarrow true$
**repeat** every round
    broadcast view to every processor;
    **if** $\mathsf{K}_i$($p_i$ is faulty) **then** $nonfaulty \leftarrow false$
**until** $nonfaulty$ and $\mathsf{S}_{\mathcal{N}} ok_j$ holds for some $j$;
$j \leftarrow \min\{k \mid \mathsf{S}_{\mathcal{N}} ok_k \text{ holds}\}$;
perform $a_j$;
**halt**.
```

Figure 2: The optimal protocol $\mathcal{F}'_{MT}$

To show that $\langle \rho, k\rangle \models \mathsf{K}_i(\mathsf{S}_{\mathcal{N}}\varphi)$, it is enough to show that $\langle \rho', k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$ for every point $\langle \rho', k\rangle$ such that p_i has the same view at $\langle \rho, k\rangle$ and $\langle \rho', k\rangle$. Given such a point $\langle \rho', k\rangle$, by Lemma 2, it is enough to show that $\langle \rho'', k\rangle \models \varphi$ for every point $\langle \rho'', k\rangle$ such that $\langle \rho', k\rangle \rightsquigarrow \langle \rho'', k\rangle$. Fix such a point $\langle \rho'', k\rangle$. Since $\langle \rho, k\rangle \models \neg\mathsf{K}_i$($p_i$ is faulty), there exists a point $\langle \eta, k\rangle$ such that p_i is nonfaulty in η and p_i has the same view at both $\langle \rho, k\rangle$ and $\langle \eta, k\rangle$, and hence also at $\langle \rho', k\rangle$ and $\langle \eta, k\rangle$; in other words, $\langle \eta, k\rangle \rightsquigarrow \langle \rho, k\rangle$ and $\langle \eta, k\rangle \rightsquigarrow \langle \rho', k\rangle$. By Lemma 10, $\langle \eta, k\rangle \rightsquigarrow \langle \rho, k\rangle$ implies $\langle \eta, k\rangle \sim \langle \rho, k\rangle$, which implies $\langle \rho, k\rangle \rightsquigarrow \langle \eta, k\rangle$. It follows that

$$\langle \rho, k\rangle \rightsquigarrow \langle \eta, k\rangle \rightsquigarrow \langle \rho', k\rangle \rightsquigarrow \langle \rho'', k\rangle\,,$$

and hence, by Lemma 2, that $\langle \rho'', k\rangle \models \varphi$ since $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$. □

One consequence of this result is that any test for $\mathsf{S}_{\mathcal{N}}\varphi$ for the set of nonfaulty processors is almost a test for $\mathsf{S}_{\mathcal{N}}\varphi$ for the set of *all* processors: such a test correctly computes whether $\mathsf{S}_{\mathcal{N}}\varphi$ holds when applied to the view of *any* processor—faulty or nonfaulty—provided the processor's view does not *prove* that it must be faulty.

Corollary 13: *Consider any system in the general omissions model with $n > 2t$. Let M_φ be a test for $\mathsf{S}_{\mathcal{N}}\varphi$ for the nonfaulty processors. If $\langle \rho, k\rangle \models \neg\mathsf{K}_i$($p_i$ is faulty), then M_φ on input $v(p_i, \rho, k)$ returns true if and only if $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$.*

Proof: Since $\langle \rho, k\rangle \models \neg\mathsf{K}_i$($p_i$ is faulty), there is a point $\langle \eta, k\rangle$ such that p_i is nonfaulty at $\langle \eta, k\rangle$ and has the same view at $\langle \rho, k\rangle$ and $\langle \eta, k\rangle$. By Lemma 12, processor p_i knows the truth of $\mathsf{S}_{\mathcal{N}}\varphi$ at $\langle \rho, k\rangle$, so $\mathsf{S}_{\mathcal{N}}\varphi$ holds at $\langle \rho, k\rangle$ if and only if $\mathsf{S}_{\mathcal{N}}\varphi$ holds at $\langle \eta, k\rangle$. Since the test M_φ must return the same answer when given p_i's view at these two points as input (the views are the same), and since M_φ returns true at $\langle \eta, k\rangle$ if and only if $\langle \eta, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$ (M_φ is a test for the nonfaulty processors), it follows that M_φ returns true at $\langle \rho, k\rangle$ if and only if $\langle \rho, k\rangle \models \mathsf{S}_{\mathcal{N}}\varphi$. □

In particular, the polynomial-space tests for $\mathsf{S}_{\mathcal{N}} ok_j$ for the nonfaulty processors given by Moses and Tuttle in the general omissions model are accurate tests for $\mathsf{S}_{\mathcal{N}} ok_j$ when given the view of any processor whose state does not prove that it is faulty. Notice that a processor p_i knows it is faulty (its view proves it is faulty) if and only if p_i is faulty in every failure pattern consistent with the messages recorded in p_i's view as missing, and this can also be checked in polynomial-space. Thus, the protocol $\mathcal{F}'_{MT}$ given in Figure 2 is an optimal protocol for a consistent choice running in polynomial space.

Theorem 14: *If $\mathcal{C}$ is an implementable, consistent simultaneous choice in the general omissions model with $n > 2t$, then $\mathcal{F}'_{MT}$ is an optimal protocol for $\mathcal{C}$. Furthermore, if $\mathcal{C}$ is practical, then $\mathcal{F}'_{MT}$ runs in polynomial space.*

Proof: Let $\mathcal{P}$ be any protocol implementing $\mathcal{C}$, and let τ and ρ be corresponding runs of $\mathcal{F}'_{MT}$ and $\mathcal{P}$. Remember that every processor is following a test M_k for $\mathsf{S}_{\mathcal{N}} ok_k$ to determine whether it should perform a_k, where M_k is a test for $\mathsf{S}_{\mathcal{N}} ok_k$ that returns true at a point if and only if $\mathsf{S}_{\mathcal{N}} ok_k$ holds at that point, given as input the view of a processor that does not know it is faulty. Furthermore, remember that a processor performs an action only if it does not know it is faulty (and hence only when these tests M_k are correct). First note that nonfaulty processors perform actions in τ at least as early as any processor does so in ρ: if any processor performs a_j at time l in ρ, then $\mathsf{S}_{\mathcal{N}} ok_j$ holds at $\langle \rho, l \rangle$ by Lemma 5 and also at $\langle \tau, l \rangle$ by Corollary 7, so all nonfaulty processors (who certainly don't know they are faulty) perform an action at time l in τ if they have not already done so. To see that $\mathcal{F}'_{MT}$ actually implements $\mathcal{C}$, note the following:

1. Each processor clearly performs at most one of the a_j in τ and does so at most once.

2. Suppose a_j is performed by some processor p at time l in τ. Since p performed a_j at time l, it does not know it is faulty at time l or at any earlier time, so the tests M_k are correct through time l. It follows that l is the first round in which some ok_k is strong common knowledge, and that j is the least index k such that ok_k is strong common knowledge in round l. Thus, in particular, every nonfaulty processor will perform a_j at time l in τ.

3. Suppose a_j is performed by some processor p in τ. If p performs a_j at time l in τ, then $\mathsf{S}_{\mathcal{N}} ok_j$ holds at time l in τ, so τ satisfies ok_j (since ok_j is a fact about the run).

4. Suppose no processor fails in τ. Then the same is true in the corresponding run ρ of the protocol $\mathcal{P}$ implementing $\mathcal{C}$, so some action must be performed at some time l in ρ. Thus, some action must be performed by the nonfaulty processors no later than time l in τ by the argument above. Since all processors are nonfaulty in τ, all processors perform this action at time l.

Thus, $\mathcal{F}'_{MT}$ is an optimal protocol for $\mathcal{C}$. □

Moses and Tuttle prove that testing for weak common knowledge (and thus for strong common knowledge, since the definitions are equivalent) is NP-hard in the general omissions model when $n > 2t$. Consequently, assuming $P \neq NP$, the protocol $\mathcal{F}'_{MT}$ cannot be implemented in polynomial time in this model. In fact, Moses and Tuttle prove that any optimal protocol for general simultaneous choice problems requires processors to perform NP-hard computations, and, thus, that optimal protocols are inherently intractable. Given the equivalence of weak and strong common knowledge, a reduction similar to that given by Moses and Tuttle shows that the same is true of optimal protocols for consistent simultaneous choice problems.

6 An Impossibility Result

While weak and strong common knowledge are equivalent in the general omissions model when $n > 2t$, they are *not* equivalent when $n \leq 2t$. It is not hard to show that some fact about the initial state must become weak common knowledge by time $t + 1$ at the latest in any run [1,5,9]. This section shows that, in many runs, such facts *never* become strong common knowledge; this is true even in runs in which no failures occur. This implies that consistent simultaneous choice problems cannot be implemented in these systems.

This distinction between general and consistent simultaneous choices comes about because of the fact that, when $n \leq 2t$, the system can become partitioned into two sets of at most t processors such that processors within the same set communicate with no trouble, but processors in different sets never communicate. Since it is consistent with this failure pattern that either set of processors is the set of faulty processors, no processor can know whether or not it is faulty. In the context of general simultaneous choices, this is not a problem, since the behavior of the faulty processors is unimportant, so each set can simply behave as if it is the set of nonfaulty processors. In the context of *consistent* choices, however, this behavior is critical: because the processors may be isolated from important information and not know whether or not they are faulty, the correct processors can become "paralyzed" and unable to act.

To make this precise, we say that two sets A and B *partition* the set of processors if A and B are two nonempty, disjoint sets of processors such that $A \cup B = P$ and $|A|, |B| \leq t$. Given a failure-free run ρ, let ρ_k be the run identical to ρ except that, for every round $l > k$, no processor in B sends or receives a message to or from any processor in A in round l of ρ_k; note that A is the set of nonfaulty processors in ρ_k. We say that ρ_k *the result of partitioning* ρ *into* A *and* B *from time* k. The following lemma says that $\langle \rho, l\rangle \rightsquigarrow \langle \rho_k, l\rangle$ for every $k \geq 0$, and in particular for $k = 0$.

Lemma 15: *Consider any system in the general omissions model with $n \leq 2t$, and let ρ be any failure-free run of this system. Suppose A and B partition the set of processors, and suppose ρ_k is the result of partitioning ρ into A and B from time k. Then $\langle \rho, l\rangle \rightsquigarrow \langle \rho_k, l\rangle$ and $\langle \rho_k, l\rangle \rightsquigarrow \langle \rho, l\rangle$ for every $k \geq 0$.*

Proof: The proof is by reverse induction on k. For the base case of $k = l$, the result is trivially true since every processor has the same view at time l in ρ and ρ_l. In this case, we actually have $\langle \rho, l\rangle \sim \langle \rho_l, l\rangle$.

For $k < l$, suppose the inductive hypothesis holds for $k+1$; that is, $\langle \rho, l\rangle \rightsquigarrow \langle \rho_{k+1}, l\rangle$ and $\langle \rho_{k+1}, l\rangle \rightsquigarrow \langle \rho, l\rangle$. Let η_1 be a run differing from ρ_{k+1} only in that no processor in B receives a message for any processors in A in round k of η_1. Note that A is the set of nonfaulty processors in both runs and that every nonfaulty processor has the same view at time l in both runs, so $\langle \rho_{k+1}, l\rangle \rightsquigarrow \langle \eta_1, l\rangle$ and $\langle \eta_1, l\rangle \rightsquigarrow \langle \rho_{k+1}, l\rangle$. Since the only missing messages in η_1 are between processors in A and processors in B, and since both A and B are of size at most t, it is consistent with the failure pattern in η_1 that either A or B is the set of faulty processors. Let η_2 be a run identical to η_1, except that now A is the set of faulty processors and B is the set of nonfaulty processors. Since every nonfaulty processor in η_1 has the same view at time l in the two runs, $\langle \eta_1, l\rangle \rightsquigarrow \langle \eta_2, l\rangle$. Similarly, since every nonfaulty processor in η_2 has the same view at time l in the two runs, $\langle \eta_2, l\rangle \rightsquigarrow \langle \eta_1, l\rangle$.[5] Now let η_3 be a run differing from η_2 only in that no (faulty) processor in A receives a message from any processor in B in round k of η_3. Because the set of nonfaulty processors is the same in η_2 and η_3, and because every nonfaulty processor has the same view at time l in both runs, we have $\langle \eta_2, l\rangle \rightsquigarrow \langle \eta_3, l\rangle$ and $\langle \eta_3, l\rangle \rightsquigarrow \langle \eta_2, l\rangle$. Note that η_3 is the result of partitioning ρ into B and A from time k, while the desired ρ_k is the result of partitioning ρ into A and B from time k; that is, the only difference between the two runs in that A is the set of nonfaulty processors in ρ_k, while B is the set of nonfaulty processors in η_3. On the other hand, since every processor (faulty or nonfaulty) has the same view at time l in both runs, it is clear that $\langle \eta_3, l\rangle \rightsquigarrow \langle \rho_k, l\rangle$ and $\langle \rho_k, l\rangle \rightsquigarrow \langle \eta_3, l\rangle$. It follows that $\langle \rho, l\rangle \rightsquigarrow \langle \rho_k, l\rangle$ and $\langle \rho_k, l\rangle \rightsquigarrow \langle \rho, l\rangle$, as desired. □

Using Lemma 15, we can now show that, even in failure-free runs, no fact about the input and the faulty processors—and thus no enabling condition ok_j—can become strong common knowledge.

Lemma 16: *Consider any system in the general omissions model with $n \leq 2t$. Let φ be a fact about the input and the faulty processors that is not valid in the system. If $\langle \rho, l\rangle$ is any point of any failure-free run ρ of the system, then $\langle \rho, l\rangle \not\models \mathsf{S}_{\mathcal{N}}\varphi$.*

Proof: Let $\bar{\iota}$ be the input to ρ. Since φ is a fact about the input and the faulty processors that is not valid in the system, there is an input vector $\bar{\iota}' \in \mathcal{I}^n$ and a set $N' \subseteq P$ such that $\langle \rho', l\rangle \not\models \varphi$ for any run ρ' with input $\bar{\iota}'$ and with N' as its set of nonfaulty processors. Two input vectors $\bar{\iota}_a$ and

[5] Note that we do not have $\langle \eta_1, l\rangle \sim \langle \eta_2, l\rangle$, since there is no processor that is nonfaulty in both runs. For this reason, the proof does not apply when $\rightsquigarrow$ is replaced by $\sim$ and, thus, the consequences of this lemma apply only to strong common knowledge and not to weak common knowledge.

$\bar{\iota}_b$ are said to be *adjacent*—denoted $\bar{\iota}_a \leftrightarrow \bar{\iota}_b$—if they differ on the initial state of only one processor; that is, for some processor p_i, we have $\iota_{j,a} = \iota_{j,b}$ for all $j \neq i$. It is not hard to see that for some $m \geq 0$ there are vectors $\bar{\iota}_0, \bar{\iota}_1, \ldots, \bar{\iota}_m$ such that $\bar{\iota} = \bar{\iota}_0 \leftrightarrow \bar{\iota}_1 \leftrightarrow \cdots \leftrightarrow \bar{\iota}_m = \bar{\iota}'$. For every j, let ρ_j be the failure-free run with input $\bar{\iota}_j$.

To complete the proof, it is enough to show that $\langle \rho, l\rangle \rightsquigarrow \langle \rho_j, l\rangle$ for every $j \geq 0$. In particular, suppose $\langle \rho, l\rangle \rightsquigarrow \langle \rho_m, l\rangle$. Let ρ' be the run differing from ρ_m only in that no processor in $P - N'$ receives any message in round l of ρ'. Since $\bar{\iota}'$ is the input to ρ' and N' is the set of nonfaulty processors in ρ', we have $\langle \rho', l\rangle \not\models \varphi$. Furthermore, since processors in N' have the same view at time l in both runs, we have $\langle \rho_m, l\rangle \rightsquigarrow \langle \rho', l\rangle$. It follows that $\langle \rho, l\rangle \rightsquigarrow \langle \rho', l\rangle$ and yet $\langle \rho', l\rangle \not\models \varphi$; thus, $\langle \rho, l\rangle \not\models \mathsf{S}_{\mathcal{N}}\varphi$ by Lemma 2.

We proceed by induction on j to show that $\langle \rho, l\rangle \rightsquigarrow \langle \rho_j, l\rangle$ for every $j \geq 0$. For $j = 0$, we are done since $\rho = \rho_0$. For $j > 0$, suppose the inductive hypothesis holds for $j-1$; that is, $\langle \rho, l\rangle \rightsquigarrow \langle \rho_{j-1}, l\rangle$. By definition, ρ_{j-1} and ρ_j differ only in the input to some processor p. Consider any partition of P into sets A and B with $p \in B$, and let η_{j-1} and η_j be the result of partitioning ρ_{j-1} and ρ_j, respectively, into A and B from time 0. By Lemma 15, we have $\langle \rho_{j-1}, l\rangle \rightsquigarrow \langle \eta_{j-1}, l\rangle$ and $\langle \eta_j, l\rangle \rightsquigarrow \langle \rho_j, l\rangle$. Because η_{j-1} and η_j differ only in the input to p, and because there is no message from $p \in B$ to any processor in A in either run, all (nonfaulty) processors in A have the same view at time l in both runs, and we have $\langle \eta_{j-1}, l\rangle \rightsquigarrow \langle \eta_j, l\rangle$. It follows that $\langle \rho, l\rangle \rightsquigarrow \langle \rho_j, l\rangle$, as desired. □

It follows from Lemma 16 that there is no solution to any consistent simultaneous choice in this model:

Theorem 17: *In the general omissions model with $n \leq 2t$, no consistent simultaneous choice is implementable.*

Proof: Suppose by way of contradiction that some protocol $\mathcal{P}$ implements some consistent simultaneous choice $\mathcal{C}$ in the general omissions model with $n \leq 2t$. Let ρ be a failure-free run of $\mathcal{P}$. Since $\mathcal{P}$ implements $\mathcal{C}$, all processors must perform some action a_j at some time l in ρ, and $\langle \rho, l\rangle \models \mathsf{S}_{\mathcal{N}} ok_j$ by Lemma 5. But Lemma 16 says that this is impossible, since ok_j is a fact about the input and faulty processors. It follows that $\mathcal{P}$ does not implement $\mathcal{C}$ after all. □

Theorem 17 states that consistent simultaneous choices cannot be implemented in systems with general omission failures in which half or more of the processors can fail. This is in marked contrast to the results of Section 5.3, which show that such choices can be implemented in systems in which fewer than half of the processors can fail.

7 Conclusions

In this paper, we have studied optimal solutions to consistent simultaneous choice problems in a variety of simple failure models. Consistent simultaneous choice problems differ from more general simultaneous choice problems in that faulty processors must perform an action (if they perform any action at all) that is consistent and simultaneous with the actions performed by the nonfaulty processors. This additional requirement seems a natural one to make in systems with benign failures. In this paper, we have presented a complete study of optimal solutions to such problems in a number of simple failure models. In the crash and send omissions failure models, optimal protocols for simultaneous choice problems derived elsewhere [1,9] are also optimal protocols for consistent simultaneous choice problems and run in polynomial time. We have shown that, in the general omissions failure model, when fewer than half the processors are faulty (that is, when $n > 2t$), a simple modification of these optimal protocols are optimal protocols in this model as well. These protocols require exponential time, but any optimal protocol in this model is inherently intractable (it requires processors to perform NP-hard computations). Furthermore, in each of these models, the optimal protocols for the consistent version of a simultaneous choice problem halt at precisely the same time as the optimal protocols for the original version. Thus, we can obtain consistency in these models at no extra cost. Finally, we have shown that, in the general omissions failure model, when half or more of the processors may fail (that is, when $n \leq 2t$), there is no solution to any consistent simultaneous choice problem: processors cannot consistently coordinate actions even in failure-free runs.

One of the technical contributions of this work is exploring the relationship between the definition of common knowledge used by Moses and Tuttle [9] and the original definition proposed by Halpern and Moses [5]. Moses and Tuttle define weak common knowledge, and Halpern and Moses essentially define strong common knowledge (although their definition is formulated in terms of fixed sets and not nonconstant sets such as the set of nonfaulty processors). Because we have shown that the two definitions are equivalent in most of the cases considered by Moses and Tuttle, we have shown that their work could have been simplified slightly by using the less subtle definition of common knowledge originally proposed by Halpern and Moses. On the other hand, where we have shown the two definitions are different (in the general omissions model with $n \leq 2t$), consistent simultaneous choice problems cannot be solved, whereas Moses and Tuttle have shown that the original simultaneous choice problems *can* be solved. In this sense, our work shows precisely where the subtlety in the definition of strong common knowledge is required by the work of Moses and Tuttle; it also shows that the difference between strong and weak common knowledge is at the heart of the difference between the consistent and original versions of simultaneous choice problems.

In this work we have analyzed problems requiring simultaneous coordination of actions, but nonsimultaneous coordination is also of interest. For example, Halpern, Moses, and Waarts [6] have considered the *Eventual Byzantine Agreement* problem, a problem in which correct processors must agree on the value of their output bits but need not choose these output bits at the same time. They show that solving such problems requires a variant of common knowledge called *continual common knowledge* and give a two-step method for transforming any solution to this problem into a round-optimal solution (optimal in the sense that no other solution outperforms it in all operating environments). Since they study this problem in two of the benign failure models considered in our paper, it is again interesting to consider consistent formulations of this problem. Using a definition of continual common knowledge strengthened in a way similar to the way we have defined strong common knowledge, Neiger [10] has generalized their work to a general class of nonsimultaneous choice problems requiring consistent coordination. In particular, he has generalized their transformation and so can transform any solution to a consistent coordination problem into an optimal solution. The number of steps in this transformation depends on the number of actions from which the processors must choose.

There still remain a few open problems to consider. First, the precise complexity of solutions to consistent simultaneous choice problems in the general omissions model when $n > 2t$ is still an open question: all we have established is that it is somewhere between NP and PSPACE. As noted by Moses and Tuttle [9], the precise complexity of solutions to *general* simultaneous choice problems in this model is also an open problem. Furthermore, the complexity of solutions to general problems is still open even when $n \leq 2t$, although we have established that solutions to *consistent* problems are impossible. Finally, we conjecture that the impossibility result of Section 6 applies to the nonsimultaneous *consistent coordination* problems mentioned above [10].

Acknowledgements

We thank Yoram Moses for enlightening discussions on the relationship between this work and the work of Moses and Tuttle and for pointing out a minor error in an earlier version of $\mathcal{F}'_{MT}$.

References

[1] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, October 1990.

[2] Ajei Gopal and Sam Toueg. Reliable broadcast in synchronous and asynchronous environments (preliminary version). In J.-C. Bermond and M. Raynal, editors, *Proceedings of the Third International Workshop on Distributed Algorithms*, volume 392 of *Lecture Notes on Computer Science*, pages 110–123. Springer-Verlag, September 1989.

[3] Vassos Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations*. Ph.D. dissertation, Harvard University, June 1984. Department of Computer Science Technical Report 11-84.

[4] Joseph Y. Halpern and Yoram Moses. A guide to the modal logic of knowledge and belief. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 480–490. Morgan-Kaufmann, August 1985.

[5] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, July 1990.

[6] Joseph Y. Halpern, Yoram Moses, and Orli Waarts. A characterization of eventual Byzantine agreement. In *Proceedings of the Ninth ACM Symposium on Principles of Distributed Computing*, pages 333–346, August 1990.

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[8] C. Mohan, R. Strong, and S. Finkelstein. Methods for distributed transaction commit and recovery using Byzantine agreement within clusters of processors. In *Proceedings of the Second ACM Symposium on Principles of Distributed Computing*, pages 89–103, August 1983.

[9] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1):121–169, 1988.

[10] Gil Neiger. Using knowledge to achieve consistent coordination in distributed systems. In preparation, July 1990.

[11] Gil Neiger and Sam Toueg. Automatically increasing the fault-tolerance of distributed algorithms. *Journal of Algorithms*, 11(3):374–419, September 1990.

[12] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[13] Kenneth J. Perry and Sam Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, 12(3):477–482, March 1986.

[14] Richard D. Schlichting and Fred B. Schneider. Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems*, 1(3):222–238, August 1983.