Null Messages, Information and Coordination:
Preliminary version

Raïssa Nataf
Technion, Israel
Guy Goren
Technion, Israel
Yoram Moses
Technion, Israel

Abstract
This paper investigates the transfer of information in fault-prone synchronous systems using null messages. The notion of an \(f\)-resilient message block is defined to capture the fundamental communication pattern for knowledge transfer. This pattern may involve null messages in addition to explicit messages, and hence, it provides a fault-tolerant extension of the classic notion of a message-chain. Based on the above, we provide tight necessary and sufficient characterizations of the generalized communication patterns, including actual messages and null messages, that can serve to solve the distributed tasks of (nice-run) Signalling and Ordered Response.

2012 ACM Subject Classification

Keywords and phrases null messages, fault tolerance, coordination, information flow.

Funding Guy Goren: Guy Goren was partly supported by a grant from the Technion Hiroshi Fujiwara cyber security research center and the Israel cyber bureau, as well as by a Jacobs fellowship. Yoram Moses: Yoram Moses is the Israel Pollak academic chair at the Technion. His work, as well as that of Raïssa Nataf, was supported in part by the Israel Science Foundation under grant 2061/19, and by the U.S.-Israel Binational Science Foundation under grant 2015820.

1 Introduction

In a classic paper [11], Lamport states that in systems in which processes have clocks and there are bounds on message transmission times it is possible to “convey information by not doing something”. He suggests that, in synchronous systems of this type, not sending a message can be viewed as sending a null message. While null message have been successfully employed to optimize communication in useful protocols (early examples that come to mind are [11, 8]), the question of how null messages convey information, and what information they convey, has only been partly addressed.

For a null message to be informative, sending an actual message must have been considered a possibility, as far as the intended receiver is concerned. For a simple example, suppose that \(p\) has a binary value \(v \in \{0, 1\}\) and messages take one step to be delivered. It can inform its neighbor \(q\) of the value of \(v\) by sending a message in case \(v = 0\) and sending a null message if \(v = 1\). Alternatively, \(p\) could send a real message iff \(v = 1\). More generally, we can use null messages to shift communication costs from executions in which they are more damaging (such as commonly occurring circumstances) to ones in which they are less so.

When failures can occur, the use of null messages can become rather challenging. If \(q\) does not receive a message from \(p\) at time \(m + 1\) in such a setting, it might not be able to distinguish whether \(p\) purposely refrained from sending at time \(m\), or whether \(p\) failed. Nevertheless, recent work [9] has shown that if the number of failures is bounded, it is possible to use null messages and “communicate by silence” in the common case, in applications of interest.
Null Messages, Information and Coordination: Preliminary version

Our purpose in this paper is to initiate a systematic analysis of the role of null messages, that would be applicable in particular when processes can fail. Specifically, we are interested in identifying communication patterns that protocols can employ to make effective use of null messages in “nice” runs, in which no failures actually occur. Consider the following example.

Figure 1 shows a weighted graph describing a network connecting two processes, a source \( s \) and an intended recipient \( j \) via \( p_1 \), \( p_2 \) and \( p_3 \). The weight of an edge denotes the cost of sending a message along the channel depicted by the edge. The source \( s \) has a binary value \( v_s \in \{0, 1\} \), and there is a bound of \( f = 2 \) on the number of processes that can fail. Note that a disparity in costs as in Figure 1 may arise if the channels from the \( p_i \)'s to \( j \) are over much more expensive communication lines such as external carriers or very long distances.

Figure 2 presents the communication graph of a protocol, in which a solid edge depicts that \( s \) should send a real message at time 0 reporting on the value of \( v_s \), while the dashed edges depict null messages that are sent under the following conditions: \( s \) should send a null message, i.e., should not send a real message, to \( p_1 \) iff \( v_s = 1 \) and \( p_1 \) should send a null message to \( j \) iff it received no message from \( s \) at time 1. Processes \( p_2 \) and \( p_3 \) should send \( j \) a null message iff they receive a message reporting that \( v_s = 1 \) at time 1. In this example, when a null message is not sent along a dashed edge, we can assume that, say, a ping message is sent instead. In a “nice” execution in which \( v_s = 1 \) and no failures occur, \( j \) will learn that \( v_s = 1 \) at a cost of 2. Note that this protocol is optimized for nice executions.

In other runs, the cost of an execution can be as high as 3003. A protocol that uses a simple message chain from \( s \) to \( j \) in this network would also inform \( j \) that \( v_s = 1 \) in the absence of failures, but at a cost of 1001. If most executions are nice, using null messages in this way yields considerable savings.

The main contributions of this paper are:

1. We capture the role played by null messages in information transfer. We characterize the information that a null message conveys, based on the circumstances in which it is sent. We define weak message chains consisting of both real and null messages, and show that information transfer depends on the recipient knowing that a weak chain has reached it. Based on weak chains, we introduce the notion of a \( f \)-resilient message block, which generalizes the classic message chain in a synchronous fault-prone setting.

2. We show that, in a precise sense, the construction of \( f \)-resilient message blocks is necessary and sufficient for ensuring information transfer in the common case, for the synchronous
crash-failure model. Moreover, we consider a coordination problem called Ordered Response in which processes must perform actions in a linear temporal order, and provide a rigorous characterization of the message patterns that can be used to solve it efficiently in the common case. Altogether, these results advance our understanding of how null message can be effectively used.

1.1 Related Work

Lamport’s seminal paper [10] focuses on the role of message chains in asynchronous message passing systems. Indeed, Chandy and Misra showed in [3] that the only way in which knowledge about the state of the system at remote sites can be gained in asynchronous systems is via the construction of message chains. As discussed above, in his later paper [11], Lamport emphasized that in synchronous systems information can also be conveyed using null messages. In a more recent paper [2], Ben Zvi and Moses analyzed knowledge gain and coordination in a model in which processes are reliable and share a global clock, and there are upper bounds (possibly greater than 1) on message transmission times along each of the channels in the network. This model allows the use of null messages. They extend the notion of a message chain to so-called syncausal message chains, that consist of explicit messages and time intervals that correspond to the upper bounds. The bound edges in a chain can account for null messages but they are also used for other means. They show that syncausal chains are necessary and sufficient for point-to-point information transfer. They define a coordination problem called Ordered Response (which we revisit in Section 4) and show that a communication pattern they call a centipede, which generalizes message chains for their model, is necessary and sufficient for solving this problem.

Our paper follows in the footsteps of a paper by Goren and Moses [5], which considered how silence can be used to reduce communication costs in the synchronous crash failure model. They consider the problem of informing a process about the value of a given variable local to a remote process without creating a message that reaches the process. Their “Silent Choir” theorem implies that in a failure-free execution, the only way that a process can learn s’s value without receiving an explicit message chain from s is for there to be a particular round m in which f + 1 processes (called a silent choir that received such a chain from s do not send a message to j. While this theorem provides insight into how silence can be used in a crash-failure model, it determines a necessary condition that is far from being sufficient. The existence of a silent choir does not guarantee that the value is learned by j, even in failure-free runs. Moreover, such information can be delivered in silence without the members of the silent choir sending null messages to j. In the current paper we take the further step of characterizing necessary and sufficient properties of communication patterns that solve their problem, and the more general Ordered Response coordination problem.

Focusing on the design of protocols that are optimized for the common case has a long tradition in distributed computing (see, e.g., [12, 9]). In our synchronous model, it has been used to design efficient protocols for Consensus by Amdur, Weber, Hadzilacos and Halpern [1] [8] and for Atomic Commitment [7, 5] by Guerraoui and Wang and others. Efficient solutions for Consensus in a synchronous Byzantine model the common case appeared in [6], which also makes explicit use of null messages there. While null messages have been effectively used to design efficient protocols in synchronous models, how null messages can be used for information transfer and coordination have not been completely characterized in a formal way.
2 Model and Preliminary Definitions

Our model is similar to the one of [5]. We consider the standard synchronous message-passing model with benign crash failures. We assume a set $\mathbb{P}$ of $n > 2$ processes. For convenience, one of the processes will be denoted by $s$ and called the source. Processes are connected via a communication network defined by a directed graph $\mathcal{N} = (\mathbb{P}, \mathcal{C})$ where an edge from process $i$ to process $j$ is called a channel, and denoted by $\mathcal{C}_{i,j}$. We assume that the receiver of a message detects the channel over which it was delivered, and thus knows the identity of the sender. All processes share a discrete global clock that starts at time $0$ and advances by increments of one. Communication in the system proceeds in a sequence of rounds, with round $m + 1$ taking place between time $m$ and time $m + 1$, for $m \geq 0$. A message sent at time $m$ (i.e., in round $m + 1$) from a process $i$ to $j$ will reach $j$ by time $m + 1$, i.e., the end of round $m + 1$. (This is the sense in which the model is synchronous.) In every round, each process performs local computations, sends a set of messages to other processes, and finally receives messages sent to it by other processes during the same round. A faulty process in a given execution fails by crashing in some round $m \geq 1$. In this case, the process behaves correctly in the first $m - 1$ rounds and it performs no actions and sends no messages from round $m + 1$ on. When crashing on round $m$, a process sends messages to an arbitrary subset of the processes to whom it is supposed to send a message in round $m$ according to the protocol. At any given time $m \geq 0$, a process is in a well-defined local state. For simplicity, we assume that the local state of a process $i$ consists of its initial value $v_i$, the current time $m$, and the sequence of the events that $i$ has observed (including the messages it has sent and received) up to that time. In particular, its local state at time $0$ has the form $(v_i, 0, \{\})$. We will also assume that once a process has crashed, its local state becomes the fixed state $\perp$. We focus on deterministic protocols, so a protocol $Q$ describes what messages a process sends and what decisions it takes, as a function of its local state. A process is said to be active at time $m$ if it has not crashed by time $m - 1$ and it correctly follows its protocol up to and including time $m$.

We will consider the design of protocols that are required to tolerate up to $f$ crashes. Thus, given $1 \leq f < n$, we denote by $\gamma^f$ the model described above in which it is guaranteed that no more than $f$ processes fail in any given run. We assume that a protocol has access to the values of $n$ and $f$ as well as to the communication network $\mathcal{N}$, all typically passed as parameters to the protocol.

A run is a description of a (possibly infinite) execution of the system. We call a set of runs $R$ a system. We will be interested in systems of the form $R_Q = R(Q, \gamma^f)$ consisting of all runs of a given protocol $Q$ in which no more than $f$ processes fail. For the purpose of our analysis in this paper we will restrict attention to the case in which the source $s$ has a binary initial value $v_s \in \{0, 1\}$, while the initial values of all other processes $i \neq s$ are fixed. Thus, there are only two distinct initial global states in a system $R_Q$. Moreover, since $Q$ is deterministic it has a single run in which $v_s = 1$ and no failures occur. We call this $Q$’s nice run, and denote it by $\hat{r}(Q)$, or simply $\hat{r}$ when $Q$ is clear from context.

2.1 Defining Knowledge

Our analysis makes use of a formal theory of knowledge to capture how null messages affect what processes do or do not learn. We sketch the theory here; see [4] for more details and a general introduction to the topic. In general, a process $i$ can be in the same local state in different runs of the same protocol. We shall say that two runs $r$ and $r'$ are indistinguishable to process $i$ at time $m$ if $r_i(m) = r'_i(m)$. Since the current time $m$ is represented in the local
state $r_i(m)$, however, $r_i(m) = r_i'(m')$ can hold only if $m = m'$. Notice that since we assume that processes follow deterministic protocols, if $r_i(m) = r_i'(m')$ then process $i$ is guaranteed to perform the same actions at time $m$ in both $r$ and $r'$ if it is active at time $m$.

**Definition 1 (Knowledge).** Fix a system $R$, a run $r \in R$, a process $i$ and a fact $\varphi$. We say that $K_i\varphi$ (which we read as “process $i$ knows $\varphi$”) holds at time $m$ in $r$ iff $\varphi$ is true of all runs $r' \in R$ such that $r_i(m) = r'_i(m)$.

Definition 1 immediately implies the so-called Knowledge property: If $K_i\varphi$ holds at (any) time $m$ in $r$, then so does $\varphi$. The logical notation for “the fact $\varphi$ holds at time $m$ in the run $r$ with respect to the system $R$” is $(R, r, m) \models \varphi$. Often, the system is clear from context and is not stated explicitly. In this paper, the system will typically consist of all the runs of a given protocol $Q$ in the current model of computation, which will be denoted by $R_Q$.

We use Boolean operators such as $\neg$ (Not), $\wedge$ (And), and $\vee$ (Or) freely in the sequel. While the basic facts $\varphi, \psi$, etc. of interest are properties of the run, knowledge can change over time. Thus, for example, $K_j(v_i = 1)$ may be false at time $m$ in a run $r$ and true at time $m + 1$, based perhaps on messages that $i$ does or does not receive in round $m + 1$.

An essential connection between knowledge and action in distributed protocols, called the **knowledge of preconditions principle** (KoP), is provided in [13]. It states that whatever must be true when a particular action is performed by a process $i$ must be known by $i$ when the action is performed. This is one way of capturing the role of indistinguishability discussed above. More formally, we say that a fact $\varphi$ is a necessary condition for an action $\alpha$ in a system $R$ if for all runs $r \in R$ and times $m$, if $\alpha$ is performed at time $m$ in $r$ then $\varphi$ must be true at time $m$ in $r$. For deterministic protocols in synchronous models such as $\gamma^f$, the KoP can be stated as follows:

**Theorem 2 (KoP, [13]).** Fix a protocol $Q$ for $\gamma^f$, let $\alpha$ be an action of process $i$ in $R_Q$. If $\varphi$ is a necessary condition for $\alpha$ in $R_Q$ then $K_i\varphi$ is a necessary condition for $\alpha$ in $R_Q$.

### 2.2 Null Messages

As Lamport mentioned, null messages may be used to convey information. Recall that we are considering synchronous settings, in which the processes share a global clock. Consequently, at time $t + 1$ a process $j$ may be able to detect the fact that $i$ did not send a message over $ch_{i,j}$ at time $t$. But what information is there in the fact that a message was not sent? That, it turns out, depends crucially on the protocol. More precisely, it depends on the conditions under which $i$ would not send this message. To account for this, we make the following definition.

**Definition 3 (Null message sent in case $\varphi$).** We say that process $i$ sends a null message over $ch_{i,j}$ at time $t$ in case $\varphi$ if for every run $r \in R_Q$ in which $i$ is active at $(r, t)$, it does not send a message over $ch_{i,j}$ at $(r, t)$ if and only if $(R_Q, r, t) \models \varphi$.

Formally, if $i$ never sends $j$ a message at time $t$ in runs of $Q$, then it is considered by the definition to be sending the null message in case $\text{true}$. Clearly, such a null message cannot provide $j$ with any nontrivial information. We consider a null message at time $t$ to be genuine if there is at least one run of the protocol $Q$ in which a message is sent over $ch_{i,j}$ at time $t$. In the sequel, unless mentioned otherwise, whenever we use the term “null message” we mean a genuine null message. We are now ready to discuss the information that a null message can provide its “recipient.”

Since we are assuming a model in which processes can crash, it is possible for a message not to be sent because the intended sender crashed, regardless of whether it would have sent
Thus, (i) $i,j$ \text{sends a null message over $\text{ch}_{i,j}$ in case $\varphi$ \text{the most that $j$ can infer from the fact that it has not received a (real) message from $i$ is that if $i$ has not failed then $\varphi$ was true at time $t$. We say that a fact $\varphi$ is \text{stable} if, once true it remains true. Facts about the past such as “$\varphi$ was true at time $t$” are stable. More formally, our above discussion yields:}

\begin{itemize}
  \item \textbf{Lemma 4.} Let $\varphi$ be a stable property, and suppose that process $i$ sends $j$ a null message at time $t$ in case $\varphi$ in protocol $Q$. If $j$ does not receive a message from $i$ at time $t+1$ in the run $r$ of $Q$ then $K_j(\text{failed},i,j)$ holds at $(r,t+1)$.
  \\
  \textbf{Proof.} The proof follows directly from Definition 3. \hfill \blacktriangle
\end{itemize}

Notice that if $f = 0$, i.e., if the system is reliable and no process ever fails, the fact $\text{failed}, \varphi$ is equivalent to $\varphi$, and so keeping silent at time $t$ implies that $\varphi$ is true. Moreover, if $j$ knows that $i$ was not faulty at time $t$, then $K_j(\text{failed},i,j)$ actually implies that $K_j(\varphi)$. In fact, an immediate consequence of Lemma 4 provides us with a variant of the Silent Choir theorem from \cite{5}:

\begin{itemize}
  \item \textbf{Corollary 5.} Suppose that, in protocol $Q$, every process $i$ in some set $I$ sends a null message over $\text{ch}_{i,j}$ at time $t$ in case $\varphi_i$ (where $\varphi_i$ is stable). If $|I| > f$ and $j$ receives no messages from processes $I$ at $(r,t+1)$, then $K_j(\bigvee_{i \in I} \varphi_i)$ holds in $r$ at time $t + 1$.
\end{itemize}

### 3 Nice-run Signalling

Consider the following scenario. Suppose that the source process $s$ starts out with a binary initial value $v_s$, and that some other process, say $j$ has a write-once variable $d_j$, that is initially unwritten. We would like to design a protocol satisfying the following two properties:

(i) $j$ eventually sets $d_j$ to 1 in the nice run of $Q$ (i.e., when $v_i = 1$ and no failures occur), and (ii) it never sets $d_j$ to 1 in runs of $Q$ in which $v_i \neq 1$.

Notice that in the problem just described, $v_s = 1$ is a necessary condition for setting $d_j$ to 1. Hence, the Knowledge of Preconditions principle (Theorem 2) implies that $j$ must know that $v_s = 1$ in order to do so. Moreover, notice that the knowledge property ensures that if $K_j(v_s = 1)$ then $v_s = 1$ is indeed true. It follows that our problem is equivalent to the problem of ensuring that, in nice runs, $j$ will eventually come to know that $v_s = 1$. This leads to the following definition of the Nice-run Signalling problem mentioned in the Introduction.

\begin{itemize}
  \item \textbf{Definition 6 (Nice-run Signalling).} A protocol $Q$ is said to solve nice-run signalling between $s$ at time 0 and $j$ at time $m$ if $(R_Q, \hat{r}, m) \models K_j(v_s = 1)$, where $\hat{r} = \hat{r}(Q)$ is $Q$’s nice run. Thus, $j$ learns that $v_s = 1$ by time $m$ in $\hat{r}$.
  \\
  We think of the fact that $v_s = 1$ as a “signal,” and wish to ensure that $j$ receives the signal in $\hat{r}$. If failures are rare and the nice run is common, then NS requires the protocol to successfully transmit the signal in the common case.
  \\
  We wish to study the communication patterns that protocols solving NS and related problems can use in their nice runs. Our analysis will make use of “communication graphs” of the following type:
  \\
  \begin{itemize}
    \item \textbf{Definition 7 (Communication Graph of $r$ w.r.t. $Q$).} For a run $r$ of a protocol $Q$, we define $CGG_Q(r) \triangleq (V,E)$, with $V = \mathbb{P} \times \mathbb{N}$ and $E = E_l \cup E_a(r) \cup E_n(r)$, where
      \begin{align*}
        E_l &= \{((i,t),(i,t+1)) : (i,t) \in V\},
        \\
        E_a(r) &= \{((i,t),(j,t+1)) : i \text{ sends a (genuine) null message over ch}_{i,j} \text{ at time } t \text{ in } r\},
      \end{align*}
  \end{itemize}
\end{itemize}
There is a weak message chain from

We construct

To establish our claim regarding

r
cases, showing that every process

messages at time

node

⟨

construction of

are the same in

null messages. The locality edges in

account for the fact that an active process can recalls
information that it has previously observed.

Definition 8 (message chains). Let Q be a protocol and let r be a run of Q.
1. A path π in CG_Q(r) is called a weak message chain, while
2. A path π in CG_Q(r) that contains no null messages is called a simple message chain.

Simple message chains correspond to traditional message chains in the literature [10]. Note that a weak message chain can contain both real messages and null messages. It is well known that in asynchronous systems, the only way to transfer information among sites is using simple message chains [3]. This is not the case in synchronous systems such as ours. Nevertheless, weak messages are essential for knowledge gain in synchronous systems, as illustrated by the following:

Lemma 9. Let Q be a protocol and let r be a run of Q. Then K_j(v_s = 1) at (r, m) only if there is a weak message chain from ⟨s, 0⟩ to ⟨j, m⟩ in CG_Q(r).

Proof. Assume, by way of contradiction, that K_j(v_s = 1) at (r, m), and that there is no message chain from ⟨s, 0⟩ to ⟨j, m⟩ in CG_Q(r). Denote

S ≜ \{⟨p, t⟩ ∈ V : There is a path from ⟨s, 0⟩ to ⟨p, t⟩ in CG_Q(r)\}

We construct r′ as follows: The initial global state r′(0) differs from r(0) only in the value of the variable v_s (thus, v_s = 0 in r′), which appears in s’s local state. All other local states are the same in r′(0) and r(0). We now prove by induction on time t that r_i(t) = r′_i(t) holds for all nodes ⟨i, t⟩ ∉ S.

Base: t = 0. Assume that ⟨i, 0⟩ ∉ S. By definition of S, it follows that i ≠ s, and by construction of r′ we immediately have that r_i(0) = r′_i(0), as required.

Step: Let t > 0 and assume that the claim holds for all nodes ⟨j, t′⟩ with t′ < t. Fix a node ⟨i, t⟩ ∉ S. Clearly, ⟨i, t − 1⟩ ∉ S, and so by the inductive hypothesis r_i(t − 1) = r′_i(t − 1). To establish our claim regarding ⟨i, t⟩, it suffices to show that i receives exactly the same messages at time t in both runs. Recall that the synchrony of the model implies that the only messages that i can receive at time t are ones sent at time t − 1. Hence, we reason by cases, showing that every process i sends the same messages at time t − 1 in both runs.

Suppose that ⟨z, t − 1⟩ ∉ S. Then by the inductive assumption r_z(t − 1) = r′_z(t − 1), i.e., it has the same local state at time t − 1 in both runs. Since Q is deterministic, z sends i a message in r at time t − 1 in r′ if and only if it does so in r. Moreover, if it sends a message, it sends the same message in both cases.

Suppose that ⟨z, t − 1⟩ ∈ S. Since ⟨i, t⟩ ∉ S we have that z does not send a message to i at time t − 1. Assume by way of contradiction that i receives such a message in r′. Recall that, by definition, if an active process p does not send q a message in r and it does in some other run r′, then p sends q a null message in r. Hence, by definition of CG_Q(r) there is an edge ⟨⟨z, t − 1⟩, ⟨i, t⟩⟩ ∈ E_n(r). This contradicts the fact that, by assumption, ⟨i, t⟩ ∉ S. It follows that, in both r and r′, process i does not receive any message from z at time t.

Since r_i(t − 1) = r′_i(t − 1) process i performs the same actions at time t − 1 in both runs. Since, in addition, i receives exactly the same messages at time t in r′ as it does in r as we have shown, it follows that r_i(t) = r′_i(t).
The inductive argument above showed that, for all processes $i$ and all times $t \leq m$, if $i$ is active at time $t$ and $(i, t) \notin S$, then $r_i(t) = r'_i(t)$. Since $(j, m) \notin S$ by assumption, it follows that, in particular, $r_j(m) = r'_j(m)$. Since $v_s \neq 1$ in $r'$ we obtain that $K_j(v_s = 1)$ at time $m$ in $r$ by the definition of the knowledge operator. This contradicts the assumption that $K_j(v_s = 1)$ holds at time $m$ in $r$, completing the proof. ▶

**Notation 1.** For all $j \in \mathbb{P}$ and times $m$, we define the fact $\sigma(j, m) \triangleq \text{“There is a weak message chain from $(s, 0)$ to $(j, m)$ in the communication graph of the current run”}$. As a straightforward implication of Lemma 9 and Definition 1 we obtain:

**Corollary 10.** Let $r$ be a run of a protocol $Q$. Then $K_j(v_s = 1)$ holds at $(r, m)$ only if $v_s = 1$ holds in $r$ and $K_j(\sigma(j, m))$ holds at $(r, m)$.

**Proof.** Suppose that $(R_Q, r, m) \models K_j(v_s = 1)$, and let $r' \in R_Q$ be a run such that $r'_j(m) = r_j(m)$. Definition 1 implies that $(R_Q, r', m) \models K_j(v_s = 1)$, and by Lemma 9 there is a message chain from $(s, 0)$ to $(j, m)$ in $r'$. I.e., $(R_Q, r', m) \models \sigma(j, m)$. It now follows by Definition 1 that $(R_Q, r, m) \models K_j(\sigma(j, m))$, as claimed. ▶

Consider the implications of Corollary 10. In order to know that $v_s = 1$, a process (say $j$ at time $m$) must know that it has been reached by at least one weak message chain. In case this happens to be a simple message chain, the process may be able to detect this based on one of its incoming messages. But what happens if there is no simple message chain from $(s, 0)$ to $(j, m)$? Then $j$ is being informed using (genuine) null messages. Of course, it cannot typically distinguish such null messages from cases in which a sender just crashed. Recall that there is a bound of $f$ on the number of failures. Hence, $j$ must know that, for any possible assignment of up to $f$ failures to the processes that is consistent with $j$’s view, there must be a weak message chain from $(s, 0)$ to $(j, m)$. This property will drive our analysis of solutions to the NS problem, and later on of our analysis of linear coordination in nice runs. From this point on, we will turn to study solutions focusing characterizing the communication patterns that they must employ in nice runs.

**Notation 2.** For ease of exposition, given a protocol $Q$ we denote the graph $CG_Q(\bar{r})$ for $\bar{r} = \hat{r}(Q)$ by NCG($Q$), and call it “$Q$’s nice communication graph.”

In light of Corollary 10 we are now ready to characterize the properties of the nice communication graphs of protocols that solve NS. We begin by defining the following useful notation.

**Notation 3 (B$_d$ path).** Fix a protocol $Q$, let $r$ be a run of $Q$, and let $B$ be a set of processes. A path $\pi$ in $CG_Q(r)$ that does not contain null messages sent by processes in the set $B$ is called $B$ null free (we write that “$\pi$ is B$_d$” for brevity).

A B$_d$ path can contain null messages, but not ones “sent” by members of $B$. As a result, if the adversary crashes the members of $B$, this path remains a legal weak message chain. A useful primitive that can guarantee information transfer in the face of up to $f$ failures can now be defined as follows.

**Definition 11 (f-resilient message block).** Let $\theta, \theta' \in \mathbb{P} \times \mathbb{N}$ be two nodes in $CG_Q(r)$. We say that there is an $f$-resilient message block from $\theta$ to $\theta'$ in a communication graph $CG_Q(r)$ if for all sets $B \subset \mathbb{P}$ such that $|B| \leq f$ there is a B$_d$ path from $\theta$ to $\theta'$ in $CG_Q(r)$. 

Observe that a simple message chain is a particular example of an $f$-resilient message block between two nodes. Indeed, in a run in which crash failures do not occur, the recipient may be able to detect that a simple chain has been completed. $f$-resilient blocks that do not consist of a simple chain will serve to capture information transfer that makes essential use of null messages. Indeed, we can now show:

**Theorem 12.** Let $\hat{r}$ be the nice run of a protocol $Q$. If $K_j(\sigma(j, m))$ holds at time $m$ in $\hat{r}$, then there must be an $f$-resilient message block from $\langle s, 0 \rangle$ to $\langle j, m \rangle$ in $NCG(Q)$.

Before we prove this theorem we define for each process its “critical time”.

**Definition 13 (Critical Time).** Let $r$ be a run of a given protocol $Q$, let $B$ be a set of processes and let $p$ be a process. For every pair $\theta$ and $\theta'$ of nodes of $CG(Q)$, we define the critical time $t_p = t_p(\theta, \theta')$ wrt. $(CG_Q(r), B)$ to be the minimal time $m_p$ such that $CG_Q(r)$ contains a $B_p$ path from $\theta$ to $\langle p, m_p \rangle$ as well as a path from $\langle p, m_p \rangle$ to $\theta'$. If no such time $m_p$ exists, then $t_p = \infty$.

We are now ready to proceed with the proof of Theorem 12.

**Proof.** Assume by way of contradiction that there is a set $B$ such that $|B| \leq f$ and every path from $\langle s, 0 \rangle$ to $\langle j, m \rangle$ in $NCG(Q)$ contains a null message from a process in $B$. Let $B$ be such a set and assume w.l.o.g. that $B$ is a minimal set with this property. Let $S_B$ be the set

$$S_B = \{ \langle p, t \rangle \in \forall : \text{There is a } B_p \text{ path from } \langle s, 0 \rangle \text{ to } \langle p, t \rangle \text{ in } NCG(Q) \}$$

Notice that our assumption about $\langle j, m \rangle$ implies that $\langle j, m \rangle \notin S_B$. Moreover, observe that if $\langle i, t \rangle \notin S_B$, then $\langle i, t' \rangle \notin S_B$ for all earlier times $0 \leq t' < t$. Figure 3 gives an example of such a set $S_B$. The highlighted nodes are in $S_B$ while the others are not in $S_B$. For the sake of clarity we do not represent all the nodes of the communication graph, e.g., we do not represent all the nodes that are connected by edges in $E_l$.

![Figure 3](image.png)

**Figure 3** A communication graph and its corresponding set $S_B$ for $B = \{q, p', l\}$

We show that there exists a run $r'$ of $Q$ such that there is no message chain from $\langle s, 0 \rangle$ to $\langle j, m \rangle$ in $CG_Q(r')$ and $r'_j(m) = \hat{r}_j(m)$. This will contradict the fact that $K_j(\sigma(j, m))$ holds at time $m$ in $\hat{r}$. We construct $r'$ as follows: All the local states of the processes are the same in $r'(0)$ and $\hat{r}(0)$. Each process $b \in B$ crashes in $r'$ at its critical time $t_b(\hat{r}(s, 0), \langle j, m \rangle)$ wrt. $(NCG(Q), B)$ without sending any messages from time $t_b$ on. We now prove by $t$ that for all $\langle i, t \rangle \notin S_B$, if $i$ has not crashed by time $t$ in $r'$, then $\hat{r}_i(t) = r'_i(t)$.

**Base:** $t = 0$. Every process starts with the same local state in both $\hat{r}$ and $r'$. Thus, it holds in particular for every $\langle i, 0 \rangle \notin S_B$ that $r'_i(0) = \hat{r}_i(0)$.

**Step:** Let $t > 0$ and assume that the claim holds for all nodes $\langle j, t' \rangle$ with $t' < t$. Fix a node $\langle i, t \rangle \notin S_B$. Clearly, $\langle i, t-1 \rangle \notin S_B$, and so by the inductive hypothesis $\hat{r}_i(t-1) = r'_i(t-1)$.
To establish our claim regarding \( \langle i, t \rangle \), it suffices to show that \( i \) receives exactly the same messages at time \( t \) in both runs. Recall that the synchrony of the model implies that the only messages that \( i \) can receive at time \( t \) are ones sent at time \( t - 1 \). Hence, we reason by cases, showing that every process \( z \neq i \) sends \( i \) the same messages at time \( t - 1 \) in both runs.

Suppose that \( \langle z, t - 1 \rangle \notin S_B \).

If \( z \notin B \) then it is active at time \( t - 1 \) in \( r' \). We have by the inductive assumption that, in particular, \( z \) sends \( i \) a message at time \( t - 1 \) in \( r' \) iff it does so in \( r \). Moreover, if it sends a message, it sends the same message in both cases.

We show that if \( z \in B \), it is active at time \( t - 1 \). Assume by way of contradiction that \( z \in B \) has failed in \( r' \) by time \( t - 1 \). It clearly does not send \( i \) a message at \( \langle z, t - 1 \rangle \) in \( r' \). We now show that then \( z \) does not send a message there in \( \hat{r} \) as well. By minimality of \( B \) each process \( p \in B \) has a corresponding finite critical time \( t_p = t_p((s, 0), (j, m)) \) wrt. \( \mathrm{NCG}(Q), B \), at which it crashes in \( r' \). Since \( z \in B \) and \( z \) has failed by time \( t - 1 \), we have that \( t_z \leq t - 1 \). By definition of \( t_z \), there is a \( \mathrm{B}_d \) path \( \pi \) from \( (s, 0) \) to \( \langle z, t_z \rangle \) in \( \mathrm{NCG}(Q) \). Denote by \( \pi_{\text{pref}} \) the prefix of \( \pi \) from \( (s, 0) \) to \( \langle z, t_z \rangle \). Clearly, \( \pi_{\text{pref}} \) is a \( \mathrm{B}_d \) path in \( \mathrm{NCG}(Q) \). Moreover, there is a path in \( \mathrm{NCG}(Q) \) from \( \langle z, t_z \rangle \) to \( \langle z, t - 1 \rangle \) consisting of locality edges (from \( E_l \)). Together, these two paths form a \( \mathrm{B}_d \) path from \( (s, 0) \) to \( \langle z, t - 1 \rangle \) in \( \mathrm{NCG}(Q) \), contradicting the assumption.

Finally, assume that \( z \in B \) is active in \( r' \) at time \( t - 1 \). Then, as in the previous case, the inductive assumption and the fact that \( Q \) is deterministic imply that exactly communication occurs between \( \langle z, t - 1 \rangle \) and \( \langle i, t \rangle \) in both runs.

Now suppose that \( \langle z, t - 1 \rangle \in S_B \). Since \( \langle i, t \rangle \notin S_B \) we have that \( z \) does not send a message to \( i \) at time \( t - 1 \).

If \( z \in B \). By definition of critical time, \( z \) fails at time \( t - 1 \) without sending any messages and in particular does not send \( i \) any message at time \( t - 1 \) in \( r' \) either.

Otherwise, \( z \notin B \). Then, \( z \) has not crashed by time \( t - 1 \) in \( r' \). Recall that, by assumption, \( \langle i, t \rangle \notin S_B \), and so, \( \langle z, t - 1 \rangle \in S_B \) it follows that \( i \) does not send a real message from \( z \) to \( i \) at time \( t \) in \( \hat{r} \). Assume by way of contradiction that it does receive such a message in \( r' \). Recall that, by definition, if an active process \( p \) does not send \( q \) a message in \( \hat{r} \) and it does in some other run \( r'' \), then \( p \) sends \( q \) a null message in \( \hat{r} \). Hence, by definition of \( \mathrm{NCG}(Q) \) there is an edge \( \langle (z, t - 1), (i, t) \rangle \in \mathcal{E}_n(r) \). This contradicts the fact that \( \langle i, t \rangle \notin S_B \). It follows that, in both \( \hat{r} \) and \( r' \), process \( i \) does not receive any message from \( z \) at time \( t \).

Since \( \hat{r}_i(t) = r'_i(t) \) process \( i \) performs the same actions at time \( t - 1 \) in both runs. Since, in addition, \( i \) receives exactly the same messages at time \( t \) in \( r' \) as it does in \( \hat{r} \) as we have shown, it follows that \( \hat{r}_i(t) = r'_i(t) \).

The inductive argument above showed that, for all processes \( i \) and all times \( t \leq m \), if \( i \) is active at time \( t \) and \( \langle i, t \rangle \notin S_B \), then \( \hat{r}_i(t) = r'_i(t) \). Since, \( \langle j, m \rangle \notin S_B \) by assumption, it follows that, in particular, \( \hat{r}_j(m) = r'_j(m) \). Since \( \neg \sigma(j, m) \) in \( \mathrm{CG}(r') \) we obtain that \( \neg K_j(\sigma(j, m)) \) at time \( m \) in \( \hat{r} \) by the definition of the knowledge operator. This contradicts the assumption that \( K_j(\sigma(j, m)) \) holds at time \( m \) in \( \hat{r} \), completing the proof.

Based on Corollary 10 and Theorem 12, we can now show that protocols solving our signalling problem must construct \( f \)-resilient message blocks:
Theorem 14 (NS necessity). Let $Q$ be a protocol that solves NS. If $K_j(v_s = 1)$ holds at time $m$ in $\hat{r}(Q)$, then there must be an $f$-resilient message block from $(s,0)$ to $(j,m)$ in $\text{NCG}(Q)$.

Proof. Let $Q$ be a protocol such that $K_j(v_s = 1)$ holds at time $m$ in the nice run $\hat{r}$ of $Q$. It follows from Corollary 10 that $K_j(\sigma(j,m))$ must hold at $(\hat{r}, m)$. The claim now follows directly from Theorem 12.

Recall that by the definition of the NS problem, $K_j(v_s = 1)$ must hold in $\hat{r}$. We show that, in NS protocols, messages only need to be convey whether the sender has detected that the run is not nice. To consider this more formally, Given a protocol $Q$ we denote by $\psi_{\text{nice}}$ the fact “the current run is $\hat{r}(Q)$”. Typically, if $f > 0$, it may be impossible for a process to know that $\psi_{\text{nice}}$ is true. Nevertheless, it may be quite common for a process to know its negation $\neg\psi_{\text{nice}}$, if it knows of a failure or detects that $v_s = 0$. Of course, because of the knowledge property, in the nice run $\hat{r}$ itself, no process will ever know $\neg\psi_{\text{nice}}$.

Definition 15 (Nice-based Message Protocols). Let $Q$ be a protocol. We say that $Q$ is a Nice-based Message protocol (NbM protocol) if (i) All real messages sent are single bit messages, and whenever a process $p$ sends a message, it sends a 0 if $K_p(\neg\psi_{\text{nice}})$ and sends a 1 otherwise and (ii) for all processes $p$, each null message sent by $p$ over any channel is a null message in case $\neg K_p(\neg\psi_{\text{nice}})$.

Notice that the only messages sent in the nice run of an NbM protocol are '1'-messages. We can now show that $f$-resilient message blocks are not only necessary for solving NS, they are also sufficient.

Theorem 16 (NS Sufficiency). $f$-resilient message blocks are sufficient for solving NS. Namely, if a communication graph $CG$ contains an $f$-resilient message block between $(s,0)$ and $(j,m)$, then there exists a protocol $Q$ that solves NS between $s$ at time $0$ and $j$ at time $m$ such that $\text{NCG}(Q) = CG$.

Proof. The assumptions guarantee that there will always be at least one path from $(s,0)$ to $(j,m)$ in $CG$ along which no “silent” process fails. We define $Q$ to be the nice-based message protocol such that $\text{NCG}(Q) = CG$. We show by induction that in all runs in which $v_s = 0$ each process along the path will detect that the run is not nice. In particular, $j$ will be able to distinguish the run from the nice one by time $m$. It follows that $K_j(v_s = 1)$ holds in $\hat{r}$ at time $m$.

Let $\text{NCG}(Q) = (V,E)$. Fix a set $B$ of processes of size $|B| \leq f$, and let $r'$ be a run in which the set of faulty processes is $B$. Let $\pi$ be a $B_d$ path in $\text{NCG}(Q)$, which is guaranteed to exist by the assumption. Let $r'$ be a run in which $v_s = 0$. We now prove by induction on time that for each node $(p,t)$ in $\pi$ it holds that $K_p(\neg\psi_{\text{nice}})$ holds at $(r',t)$.

Base: $t = 0$. In this case, $p = s$. Since $v_s$ appears in $i$'s local state and its value differs to its value in the nice run, $K_i(\neg\psi_{\text{nice}})$ holds at time 0.

Step: $t > 0$. We consider the nodes $(q,t-1)$ and $(p,t)$ in $\pi$. By the induction hypothesis $K_q(\neg\psi_{\text{nice}})$ holds at $t-1$ in $r'$. We now reason by cases according to the kind of the edge $((q,t-1),(p,t))$ in $\text{NCG}(Q)$.

- If $((q,t-1),(p,t)) \in E_1$ then $p = q$ and since the fact $\neg\psi_{\text{nice}}$ is a stable property we have by the induction hypothesis that $K_p(\neg\psi_{\text{nice}})$ holds at $(r',t)$.
- If $((q,t-1),(p,t)) \in E_m(\hat{r})$:
  - If in the run $r'$ process $q$ does not send $p$ a message, then $p$ detects that the run is not $\hat{r}$ (in which, by assumption, it would receive a message from $q$).
If \( q \) does send a message to \( p \) in \( r' \) then, by the induction assumption and the fact that \( q \) sends 0 if \( K_q(\neg \psi_{\text{nice}}) \) it follows that \( p \) receives a different message in \( r' \) and in \( r \), and so \( K_p(\neg \psi_{\text{nice}}) \) holds at time \( t \).

If \( ((q, t - 1), (p, t)) \in E_p(\hat{r}) \) we have by the choice of \( \pi \) that \( q \) does not fail in \( r' \) and by the induction assumption \( K_q(\neg \psi_{\text{nice}}) \) holds at time \( t - 1 \). Recall that, by assumption, in \( Q \) process \( q \) can send a null message only in case \( \neg (K_q(\neg \psi_{\text{nice}})) \). Since, by the inductive assumption on time \( t - 1 \) this is not the case, \( q \) must send \( p \) a ‘0’-message. Since such messages are never sent in \( \hat{r} \), we again conclude that \( K_p(\neg \psi_{\text{nice}}) \) holds at time \( t \) in \( r' \), as desired.

We have shown that for all runs \( r' \) in which \( v_s \neq 1 \) it is the case that \( r'_j(m) \neq \hat{r}_j(m) \). Consequently, \( v_s = 1 \) for all runs \( r \) such that \( r_j(m) = \hat{r}_j(m) \) and so, by definition of the knowledge operator, we obtain that \( K_j(v_s = 1) \) holds at \( (\hat{r}, m) \), as claimed.

Combining Theorem [13] with Theorem [18] we obtain a tight characterization of the communication patterns needed to solve \( \text{NS} \): Every solution must construct an \( f \)-resilient message block, and for every \( f \)-resilient message block, there exists an \( \text{NS} \) protocol that uses only the paths in this block in its nice run.

4 Ordered Response

In this section we consider a coordination problem called Ordered Response (OR) and characterize the communication patterns that are needed to solve it. This problem was originally defined in [2], and it requires a sequence of actions to be performed in linear temporal order, in response to a signal from the environment. In our case the signal will be determined by the value of \( v_s \). We assume that each process \( i_h \in \{i_1, i_2, \ldots, i_k\} \) has a specific action \( a_h \) to perform, and that the actions should be performed in order, provided that initially \( v_s = 1 \).

Definition 17 (Ordered Response). We say that a protocol \( Q \) is consistent with the instance \( \text{OR} = \langle v_s = 1, a_1, \ldots, a_k \rangle \) of the Ordered Response (OR) problem if it guarantees that \( a_h \) occurs in a run only if \( v_s = 1 \) and \( a_1, \ldots, a_{h-1} \) have occurred. In particular, if both \( a_h \) and \( a_{h+1} \) occur and they do so at times \( t_h \) and \( t_{h+1} \) respectively, then \( t_h \leq t_{h+1} \). The protocol \( Q \) solves this instance if, in addition, all of the actions \( a_h \) are performed in \( Q \)’s nice run.

We shall denote by \( \underline{a}_h \) the fact that the action \( a_h \) has (already) been performed. Since, by definition of OR, both \( v_s = 1 \) and \( \underline{a}_{h-1} \) are necessary conditions for performing \( a_h \), a direct application of the KoP (Theorem [2]) yields:

Lemma 18. Suppose that \( Q \) solves the instance \( \text{OR} = \langle v_s = 1, a_1, \ldots, a_k \rangle \) of OR. For every run \( r \) of \( Q \) and action \( a_h \) performed in \( r \), we have

1. \((R_Q, r, t_h) \models K_{i_h}(v_s = 1)\), and
2. \((R_Q, r, t_h) \models K_{i_h}(\underline{a}_{h-1})\) if \( h > 1 \).

For a protocol \( Q \) solving an instance of Ordered Response, every action \( a_h \), is performed at some specific time \( t_h \) in the nice run \( \hat{r} = \hat{r}(Q) \). For ease of exposition we denote the corresponding node of \( \text{NCG}(Q) \) by \( \theta_h \triangleq \langle i_h, t_h \rangle \).

We can use Lemma [18] to provide necessary conditions on the nice communication graph of protocols that solve Ordered Response. Lemma [18 1] implies that \( Q \) must perform Nice-run signalling to \( \theta_h \), for all actions \( a_h \). Lemma [18 2], in turn, implies that \( i_h \) needs to learn that
an−1 has been performed in order to perform its action. A straightforward way to do this is by direct signalling, i.e., by creating an f-resilient block between θan−1 and θn. While this is a possible solution, it is not the only way that ιb can obtain this knowledge. It can also learn about this action indirectly, which is where the Ordered Response problem goes beyond Nice-run signalling. We can show the following:

**Theorem 19 (OR Necessity).** Let Q be a protocol solving OR = \(\{v_s = 1, a_1, \ldots, a_k\}\). Then

1. NCG(Q) contains an f-resilient message block between \(s, 0\) and \(\theta_x\), for each \(x \leq k\); and
2. For every set \(B'\) of size \(|B'| \leq f - 1\) and for every \(x < k\) there is a \(B_d\) path from \(\langle i_x, t_x + 1\rangle\) to \(\theta_{x+1}\) with \(B = B' \cup \{i_x\}\).

**Proof.**

1. By Lemma 18 we have that for each \(b \leq k\), it holds that \((R_Q, r, \theta_b) \models K_{i_b}(v_s = 1)\).

   Theorem 14 implies that there is an f-resilient message block between \(\langle s, t_0\rangle\) and \(\theta_x\) for each \(x \leq k\), as claimed.

2. Recall that since Q solves OR, for all \(x \leq k - 1\) we have by Lemma 18 that \(K_{i_{x+1}}(a_x)\) must hold at time \(t_{x+1}\) in Q’s nice run \(\hat{r}\). Assume by way of contradiction that there exist \(x \leq k - 1\) and a set of processes \(B = B' \cup \{i_x\}\) with \(|B'| \leq f - 1\) and there is no \(B_d\) path in NCG(Q) from \(\langle i_x, t_x + 1\rangle\) to \(\theta_{x+1}\). In addition, assume w.l.o.g. that \(B\) is a minimal set with this property. The proof is very similar to that of Theorem 14 with \(\langle i_x, t_x + 1\rangle\) assuming the role played there by \(\langle s, 0\rangle\). Let \(S_B\) be the set

\[S_B \triangleq \{(p, t) \in V : \text{There is a } B_d \text{ path from } \langle i_x, t_x + 1\rangle \text{ to } \langle p, t\rangle \text{ in NCG(Q)}\}\]

Notice that our assumption about \(\theta_{x+1}\) implies that \(\theta_{x+1} \notin S_B\). We claim that there exists a run \(r'\) of Q in which \(a_x\) is not performed such that \(r'_{i_{x+1}}(m) = \hat{r}_{i_{x+1}}(m)\), where \(\hat{r}\) is Q’s nice run. This will contradict the fact that \(K_{i_{x+1}}(a_x)\) holds at time \(t_{x+1}\) in \(\hat{r}\).

We construct \(r'\) as follows: The global states of \(\hat{r}'\) and \(\hat{r}\) are the same from time 0 up to and including \(t_x\). In addition, each process \(b \in B\) crashes in \(r'\) at its critical time \(t_b(\theta_x, \theta_{x+1})\) wrt. \((\text{NCG}(Q), B)\) without sending any messages from time \(t_b\) on nor executing actions. Notice that, since \(i_x \in B\), we have in particular that process \(i_x\) fails in \(r'\) without executing \(a_x\). The rest of the proof proceeds exactly as in Theorem 14 showing by induction on time \(t\) that for all nodes \(\langle i, t\rangle \notin S_B\), if \(i\) has not crashed by time \(t\) in \(r'\), then \(\hat{r}(t) = r'(t)\). Since \(\theta_{x+1} \notin S_B\), we thus obtain that \(\hat{r}_{i_{x+1}}(t_{x+1}) = r'_{i_{x+1}}(t_{x+1})\).

Since \(a_x\) is not performed in \(r'\), the claim follows.

Observe that Item 2 of Theorem 19 implies the existence of an \((f - 1)\)-resilient message block in which \(i_x\) does not send null messages. This requirement is weaker than the existence of an f-resilient message block.

**Definition 20 (Conservative OR protocols).** Let Q be a deterministic protocol that solves OR = \(\{v_s = 1, a_1, \ldots, a_k\}\). We say that Q is conservative for OR if for every run \(r\) of Q and all \(j \leq k\) the following is true: Process \(i_j\) performs \(a_j\) at \(t_j\) only if \(\neg K_j(\neg \psi_{\text{nice}})\) holds at \((r, t_j)\).

Observe that in a conservative protocol, a process \(i_x\) that knows of a failure of some process \(b\) occurring at time \(m_b\) (and thus, in particular, that the run is not nice) is not allowed to perform its action. Since by Theorem 19 only \((f - 1)\)-resilient message blocks are required between two consecutive processes in the OR instance, it may occur that the failure of \(f - 1\) other processes disconnects \(\theta_x\) from \(\theta_h\) and \(\rho_h \triangleq \langle b, m_b\rangle\) from \(\theta_h\), and we would obtain that \(\theta_h\) does not distinguish the current run from the nice run and \(a_h\) would be performed. This is clearly a violation of OR. We illustrate the described above in Figure 4. We can show that in order to prevent such a scenario there must be a \(B_d\) path from \(\theta_x\) or
Figure 4 The problematic scenario that Theorem 21 solves. If this occurs then b’s failure at $\rho_b$ can cause $i_x$ not to perform $a_x$. The protocol must therefore provide an $f - 1$ resilient message block to $\theta_h$ from one of $\theta_x$ and $\rho_b$.

from b after its potential failing node $\rho_b$, to $\theta_h$. This way, if $b$ fails at $\rho_b$ and prevents $x$ from acting, then $i_h$ will distinguish the current run from the nice run at $\theta_h$. Acting conservatively, $i_h$ will also refrain from acting, and thus avoid causing a violation of the OR specification. Formally:

▶ Theorem 21. Let $Q$ be a conservative protocol solving $\text{OR} = \langle v_s = 1, a_1, \ldots, a_k \rangle$. For all nodes $\rho_b = \langle b, m_b \rangle$, indices $x < h \leq k$ and sets $B \subseteq \mathbb{P}$, if 1. there is a path $\pi$ from $\rho_b$ to $\theta_x$ in $\text{NCG}(Q)$ that starts with an edge $(\rho_b, \rho_q) \in E_m$ and contains no edges corresponding to null message by $b$, and 2. $b \in B$, $|B| \leq f$ and there is no $B_d$ path from $\theta_x$ to $\theta_h$ in $\text{NCG}(Q)$, then there is a $B_d$ path from $(b, m_b + 1)$ to $\theta_h$ in $\text{NCG}(Q)$.

The proof of Theorem 21 appears in the Appendix. In a precise sense, the necessary conditions in this theorem and in Theorem 19 are tight. As we now show, there exist protocols solving Ordered Response that satisfy precisely these conditions.

▶ Theorem 22 (Sufficient conditions for OR). Let $\text{OR} = \langle v_s = 1, a_1, \ldots, a_k \rangle$ be an instance of an OR problem. Suppose that $Q$ is a conservative and NbM protocol and, with respect to times $t_1 \leq t_2 \leq \cdots \leq t_k$, the protocol $Q$ satisfies the necessary conditions stated in Theorems 19 and 21 with respect to the nodes $\{\theta_j = \langle i_j, t_j \rangle\}_{1 \leq j \leq k}$. If, in $Q$, each process $i_h$ performs $a_h$ at time $t_h$ precisely if $\lnot K_{i_h} \lnot \psi_\text{nice}$ (i.e., if $i_h$ has not detected that the run is not nice), then $Q$ solves $\text{OR}$.

Taken together, Theorems 19, 21, and 22 provide a characterization of the communication patterns that can solve Ordered Response using null messages. This characterization is tight for communication patterns of conservative protocols that solve OR.

5 Conclusions and Future Work

Our results in Sections 3 and 4 provide tight necessary and sufficient conditions on the message patterns that can solve NS and OR in the common case. There are several natural ways in which this investigation can be extended. One is to extend our investigation into
solutions for problems that are efficient beyond the failure-free nice runs that we considered. Notice that Lemmas 4 and 9 as well as Corollaries 5 and 10 apply to null messages in arbitrary (not necessarily nice) runs. But once a process discovers a number of failures, the information that it can extract from null messages increases, and some rather challenging questions arise.

In the synchronous model we have considered, messages take precisely one round to be delivered. As shown in [2], assuming an upper bound on transmission times gives rise to a much richer structure, even in the absence of failures. Doing so in a fault-prone setting such as ours would necessarily lead to generalizations of message chains that extend both f-resilient message blocks and the centipedes of [2].

A completely different direction worth pursuing is seeking real-world applications in costly paths of communication can be traded by inexpensive ones, by judicious use of null messages. These may arise in models that do not quite fit the ones that have been studied, and developing the theory supporting efficient practice in such cases would be most worthwhile.

References

1 Eugene S. Amdur, Samuel M. Weber, and Vassos Hadzilacos. On the message complexity of binary byzantine agreement under crash failures. Distributed Computing, 5(4):175–186, 1992.
2 Ido Ben-Zvi and Yoram Moses. Beyond lamport’s happened-before: On time bounds and the ordering of events in distributed systems. Journal of the ACM (JACM), 61(2):1–26, 2014.
3 K. M. Chandy and J. Misra. How processes learn. Distributed Computing, 1(1):40–52, 1986.
4 Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. Reasoning About Knowledge. MIT Press, 1995. doi:10.7551/mitpress/5803.001.0001
5 Guy Goren and Yoram Moses. Silence. J. ACM, 67:3:1–3:26, 2020. doi:10.1145/3377883
6 Guy Goren and Yoram Moses. Optimistically tuning synchronous byzantine consensus: another win for null messages. Distributed Comput., 34(5):395–410, 2021. doi:10.1007/s00446-021-00393-8
7 Rachid Guerraoui and Jingjing Wang. How fast can a distributed transaction commit? In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017, pages 107–122. ACM, 2017. doi:10.1145/3034799
8 Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. Mathematical Systems Theory, 26(1):41–102, 1993.
9 Alex Kogan and Erez Petrank. A methodology for creating fast wait-free data structures. In ACM SIGPLAN Notices, volume 47, pages 141–150. ACM, 2012.
10 L. Lamport. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7):558–565, 1978.
11 Leslie Lamport. Using time instead of timeout for fault-tolerant distributed systems. ACM Trans. Program. Lang. Syst., 6:254–280, 1984. doi:10.1145/2993.2994
12 Barbara Liskov. Practical uses of synchronized clocks in distributed systems. Distributed Computing, 6(4):211–219, 1993.
13 Yoram Moses. Relating knowledge and coordinated action: The knowledge of preconditions principle. In Proceedings of TARK, pages 231–245, 2015. URL: https://doi.org/10.48550/arXiv.1606.07525
Null Messages, Information and Coordination: Preliminary version

Appendix

Proof of Theorem 21

Proof. Assume, by way of contradiction, that assumptions (1) and (2) hold but there is no $B_g$ path from $(b, m_b + 1)$ to $\theta_h$ in NCG(Q). Let $x \leq h < k$, let $B$ be a set of processes such that $|B| \leq f$ and let $b \in B$ such that there is a path from $p_b = (b, m_b)$ to $\theta_x$ that starts with an edge $(p_b, p_q) \in E_m$ that does not contain null messages sent by $b$ in NCG(Q). In addition there is no $B_g$ path neither from $\theta_x$ to $\theta_h$ nor from $(b, m_b + 1)$ to $\theta_h$. The idea is to construct a run $r'$ in which $i_x$ knows the current run is not nice and since $Q$ is a conservative protocol, it follows that $a_{i_x}$ does not occur. In addition, the processes of $B$ fail in a way that $i_{h+1}$ does not differentiate $r'$ from $\hat{r}$; contradicting the fact that $K_{i_x}(a_x)$ holds at $(\hat{r}, t_h)$. We construct the run $r'$ to be identical to $\hat{r}$ up to and including time $m_b$ at which point $b$ fails without sending any messages along the paths reaching $\theta_x$ ($b$ does send the other messages it is supposed to send at time $m_b$). In addition, in $r'$ each process $b' \in B$ such that $b' \neq b$ fails at its critical time $t_{b'} = t_b(\theta_x, \theta_h)$ wrt. (NCG(Q), $B$) without sending any messages. Let $S_B = \{(p, t) \in \mathcal{V} : \text{There is a } B_g \text{ path from } \theta_x \text{ to } (p, t) \text{ in NCG(Q)}$

or a $B_g$ path from $(b, m_b + 1)$ to $(p, t)\}$

An argument analogous to that in Theorem 16 now shows that $K_{i_x}(\neg \psi_{nice})$ holds at time $t_x$ and then $i_x$ does not act. Observe that by the assumption, $\theta_h \notin S_B$. Moreover, the same argument as in the proof of Theorem 14 now shows by induction on $t$ that for each node $(i, t) \notin S_B$, it holds that $r'_i(t) = \hat{r}_i(t)$. In particular, since $(i_h, t_h) \notin S_B$ we can conclude that $\hat{r}_{i_h}(t_h) = r'_{i_h}(t_h)$, contradicting the fact that $K_{i_h}(a_x)$ holds at $(\hat{r}, t_h)$ as required by Lemma 18.

Proof of Theorem 22

Proof. Let OR = $\langle v_s = 1, a_1, \ldots, a_k \rangle$ be an instance of an OR problem. In addition, let $Q$ be an NbM communication protocol that satisfies the conditions stated in the Theorem. In particular in every run such that, $K_{i_h}(\neg \psi_{nice})$ holds at $(r, t_h)$, process $i_h$ performs $a_h$ at $t_h$.

Let $h \leq k$. We prove that

1. $K_{i_h}(v_s = 1)$ holds at $(\hat{r}, t_h)$,
2. $K_{i_h} v_h-1$ holds at $(\hat{r}, t_h)$ for every $h > 1$.

Let $r'$ be a run in which $v_s \neq 1$. We have from the first condition that there is an $f$-resilient message block from $(s, 0)$ to $\theta_h$ for each $h \leq k$. Hence, by Theorem 16 we have that $r'_i(t_h) \neq \hat{r}_i(t_h)$ so in $\hat{r} : K_{i_h}(v_s = 1)$ holds at time $t_h$.

Now assume $v_s = 1$. Let $1 < h < k$. We prove that if $i_h$ does not execute $a_h$ at $t_h$ then $r'_{i_{h+1}}(t_{h+1}) \neq \hat{r}_{i_{h+1}}(t_{h+1})$. Assume $i_h$ does not execute $a_h$. Since $v_h = 1$ there are two possibilities:

1. $i_h$ fails by time $t_h$ without performing $a_h$. Since that for every set $B'$ of processes of size $|B'|$ there is a $B_g$ path from $(i_h, t_h + 1)$ to $\theta_{h+1}$ with $B = B' \cup \{i_h\}$ we can show, as in Theorem 16 that $\hat{r}_{i_{h+1}}(t_{h+1}) \neq r'_{i_{h+1}}(t_{h+1})$.
2. $i_h$ does not fail by time $t_h$. Since $a_h$ is not performed, it follows that $K_{i_h}(\neg \psi_{nice})$ holds at time $t_h$ in $r'$. We separate into 2 cases:

a. There is a path $\pi$ from $(i_h, t_h + 1)$ to $\theta_{h+1}$ in NCG(Q) such that in $r'$ no process sends a null message in $\pi$ fails. Then, as in Theorem 15 we show by induction on $\pi$’s length that $r'_{i_{h+1}}(t_{h+1}) \neq \hat{r}_{i_{h+1}}(t_{h+1})$.\]
b. In every path from \( \langle i_h, t_h + 1 \rangle \) to \( \theta_{h+1} \) there is a process that sends a null message that fails. Then it means that by \( t_{h+1} \) there have been \( f \) failures (this results from the fact that for every set \( B \) of size \( |B| \leq f - 1 \) there is a \( B_d \) path between \( \langle i_h, t_h + 1 \rangle \) and \( \theta_{h+1} \)). Denote by \( B = \{b_1, b_2, \ldots, b_f\} \) the processes that have failed by time \( t_{h+1} \). Recall that the fact that \( i_h \) does not act at time \( t_h \) implies that \( K_{i_h}(\neg \psi_{nice}) \) holds at time \( t_h \) in \( r' \).

We look at the smallest \( 2 \leq j \leq h \) for which \( a_j \) has not been performed at \( t_j \) in \( r' \) (In particular it may be that \( j = h \)). By the choice of \( j \) and the protocol, we have that in this run \( r' \) there is a path starting by an edge \((\rho_b, \rho_q) \in E_m\) (denote \( \rho_b = (b, m_b) \)) to \( \theta_j \) in \( \text{NCG}(Q) \) such that \( b \) fails by time \( m_b \) (Otherwise the failure would not detected).

By condition 3, we have that there is at least a \( B_d \) path \( \pi \) in \( \text{NCG}(Q) \) from \( \langle i_j, t_j + 1 \rangle \) to \( \theta_{h+1} \) or a \( B_d \) path \( \pi \) in \( \text{NCG}(Q) \) from \( \langle b, l_b + 1 \rangle \) to \( \theta_{h+1} \). Then, as in Theorem 16 we can show by induction on path length that \( r'_{t_{h+1}}(t_{h+1}) \neq \hat{r}_{t_{h+1}}(t_{h+1}) \).

We hence have showed that
- \( K_{i_h}(v_s = 1) \) holds at \( (\hat{r}, t_h) \),
- \( K_{i_h} a_{h-1} \) holds at \( (\hat{r}, t_h) \) for every \( h > 1 \)

Since \( (RQ, \hat{r}, t_h) \models K_{i_h} a_{h-1} \) for all \( h > 1 \), it follows that \( t_h \geq t_{h-1} \) for all \( h > 1 \). Consequently, \( Q \) solves the Ordered Response problem, as claimed. \[\blacksquare\]