Knowledge and Common Knowledge in a Distributed Environment

Joseph Y. Halpern

IBM Research Laboratory,
San Jose, CA 95193

Yoram Moses

Computer Science Department
Stanford University, Stanford, CA 94305

Abstract: We argue that the right way to understand distributed protocols is by considering how messages change the state of knowledge of a system. We present a hierarchy of knowledge states that a system may be in, and discuss how communication can move the system's state of knowledge of a fact up the hierarchy. Of special interest is the notion of common knowledge. Common knowledge is an essential state of knowledge for reaching agreements and coordinating action. We show that in practical distributed systems, common knowledge is not attainable. We introduce various relaxations of common knowledge that are attainable in many cases of interest. We describe in what sense these notions are appropriate, and discuss their relationship to each other. We conclude with a discussion of the role of knowledge in distributed systems.

1. Introduction

The notion of knowledge in a distributed environment is fundamental to many issues in distributed computing. Many tasks in a distributed system directly involve the achievement of specific states of knowledge and others crucially depend on a variety of constraints on the state of knowledge of the parties involved. Reasoning about such states of knowledge plays an essential role in understanding the correctness of distributed communication protocols such as "Handshake" protocols, distributed agreement protocols such as the "Byzantine Agreement" [PSL,DS] and "Transaction Commit" [Li,LF], as well as cryptographic protocols such as the "Oblivious Transfer" protocol [HR]. Communication in a distributed system can be viewed (and often *should* be viewed) as the act of transforming the system's state of knowledge.

The general concept of knowledge has received considerable attention in a variety of fields, ranging from Philosophy [Hi] and Artificial Intelligence [Mc], to Game Theory [Au] and Psychology [CM]. Although in different contexts knowledge is assumed to mean different things, one property that is generally required of knowledge is that only true things be known. More formally, knowledge satisfies the axiom

$$K_i p \supset p$$

i.e., if an individual i knows p, then p is true.

In the presence of many individuals, an individual may have knowledge about other individuals' knowledge, in addition to his knowledge about the physical world. This often requires care in distinguishing subtle differences between seemingly similar states of knowledge. A classical example is the "dirty children" puzzle – a variant of the well known "wise men" or "cheating wives" puzzles. The version given here is taken from [Ba]:

> Imagine n children playing together. The mother of these children has told them that if they get dirty there will be severe consequences. So, of course, each child wants to keep clean, but each would love to see the others get dirty. Now it happens during their play that some of the children, say k of them, get

mud on their foreheads. Each can see the mud on others but not on his own forehead. So, of course, no one says a thing. Along comes the father, who says, "At least one of you has mud on your head," thus expressing a fact known to each of them before he spoke (if $k > 1$). The father then asks the following question, over and over: "Can any of you prove you have mud on your head?" Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen?

There is a "proof" that the first $k-1$ times he asks the question, they will all say "no" but then the kth time the dirty children will answer "yes."

The "proof" is by induction on k. For $k = 1$ the result is obvious: the dirty child sees that no one else is muddy, so he must be the muddy one. Let us do $k = 2$. So there are just two dirty children, a and b. Each answers "no" the first time, because of the mud on the other. But, when b says "no," a realizes that he must be muddy, for otherwise b would have known the mud was on his head and answered "yes" the first time. Thus a answers "yes" the second time. But b goes through the same reasoning. Now suppose $k = 3$; so there are three dirty children, a, b, c. Child a argues as follows. Assume I don't have mud on my head. Then, by the $k = 2$ case, both b and c will answer "yes" the second time. When they don't, he realizes that the assumption was false, that he is muddy, and so will answer "yes" on the third question. Similarly for b and c.

Assuming $k > 1$, the father didn't tell the children anything they didn't know already. Yet it is easy to see that without his statement the children with mud on their heads would never be able to deduce this. What was the role of the father's statement? We will return to this question.

How does the notion of knowledge generalize from an individual's knowledge to a group's knowledge? In other words, what does it mean to say that a group G of individuals knows a fact p? More than one possibility is reasonable, with the appropriate choice depending on the application:

(i) $I_G p$ (read "the group G has *Implicit Knowledge* of the fact p"): We say that G has implicit knowledge of p iff by pooling together their knowledge, the members of G can conclude p. For instance, if one member of G knows q and another knows $q \supset p$, the group G can be said to have implicit knowledge (abbrev. I-knowledge) of p.

(ii) $S_G p$ (read "Someone in G knows p", S-knowledge in short): We say that G has S-knowledge of p iff some member of G knows p. More formally,

$$S_G p \equiv \bigvee_{i \in G} K_i p.$$

(iii) $E_G p$ (read "Everyone in G knows p", E-knowledge in short): G is said to have E-knowledge of p iff all members of G know p. More formally,

$$E_G p \equiv \bigwedge_{i \in G} K_i p.$$

(iv) $E_G^k p$, $k \geq 2$ (read "p is E^k-knowledge in G"): $E_G^k p$ is defined by

$$E_G^1 p = E_G p,$$

$$E_G^{k+1} p = E_G E_G^k p, \text{ for } k \geq 1.$$

p is said to be E^k-knowledge in G if "everyone in G knows that everyone in G knows that ... that everyone in G knows that p is true" is true, where the phrase "everyone in G knows that" appears in the the sentence k times. Equivalently,

$$E_G^k p \equiv \bigwedge_{i_j \in G, 1 \leq j \leq k} K_{i_1} K_{i_2} \cdots K_{i_k} p.$$

(v) $C_G p$ (read "p is *Common Knowledge* in G"): p is said to be common knowledge in G if p is true, and is E_G^k-knowledge for all $k \geq 1$. In other words,

$$C_G p \equiv p \wedge E_G p \wedge E_G^2 p \wedge \cdots \wedge E_G^n p \wedge \cdots$$

In particular, $C_G p$ implies all formulas of the form $K_{i_1} K_{i_2} \cdots K_{i_n} p$, where the i_j are all members of G, for any finite n.

(We will omit the subscript G when the group G is understood from context.)

Returning to the "dirty children" puzzle, we denote by $\mathbf{m}$ the fact *"There are children with mud on their foreheads"*, and observe the state of the children's knowledge of $\mathbf{m}$. Before the father spoke $E^{k-1}\mathbf{m}$ was true, and $E^k\mathbf{m}$ wasn't. Let us see this in the case $k = 3$. There are 3 dirty children, and therefore each child sees at least 2 dirty children. In particular, a dirty child sees exactly two dirty children. He therefore knows that each other child knows of at least one dirty child. So, we have $E^2\mathbf{m}$, but not $E^3\mathbf{m}$.

We leave it to the reader to check that when there are k dirty children, $E^k\mathbf{m}$ suffices to ensure that the dirty children will be able to prove their dirtyness, whereas $E^{k-1}\mathbf{m}$ does not. After the father tells the children $\mathbf{m}$, $\mathbf{m}$ becomes *common knowledge*. This implies $E^k\mathbf{m}$, and therefore the dirty children succeed in proving their dirtyness. (A more detailed analysis of this argument, and variants of it, will appear in [DHM].)

It is clear that the notions of group knowledge introduced above form a hierarchy, with

$$Cp \supset \cdots \supset E^{k+1}p \supset \cdots \supset Ep \supset Sp \supset Ip \supset p.$$

However, depending on the circumstances, these notions might not be distinct. For example, consider a model of parallel computation in which a collection of n processors share a common memory. If their knowledge is stored in memory then we arrive at a situation in which $Cp \equiv E^k p \equiv Ep \equiv Sp \equiv Ip$. By way of contrast, in a distributed system in which n processors are connected via some communication network and each one of them has its own memory, it is clear that the above hierarchy is strict. Moreover, in such a system every two levels in the hierarchy can be separated by an actual task, in the sense that there will be an action for which one level in the hierarchy will suffice, but no lower level will. It is quite clear that this is the case with $Ep \supset Sp \supset Ip$, and the "dirty children" puzzle shows that $E^k p$ is strictly stronger than $E^{k-1}p$, for $k > 1$. The fact that Cp is stronger than $E^k p$ follows, because $Cp \supset E^{k+1}p$.

In the case of a distributed system a very important question arises: how does the state of knowledge of a fact p change by the communication process? How can this state of knowledge climb up our hierarchy? The vast majority of the communication in a distributed system can be viewed as the act of improving the state of knowledge (in the sense of "climbing up the hierarchy") of certain facts. This is an elaboration of the view of communication in a network as the act of "sharing knowledge". Taking this view, two notions come to mind. One is *fact discovery* – the act of changing the state of knowledge of a fact p from being implicit knowledge to levels of explicit knowledge (usually S-knowledge or E-knowledge), and the other is *fact publication* – the act of changing the state of knowledge of a fact that is known to at least one individual, but is not common knowledge, to common knowledge. An example of fact discovery is detecting global properties of a system, such as deadlock. The knowledge the system initially has of the deadlock is I-knowledge, and the detection algorithm improves this state to S-knowledge. An example of fact publication is the introduction of a new communication convention. Here the initiator(s) of the convention wish to make the new convention common knowledge.

Both fact discovery and fact publication are interesting notions that are worth investigation. However, in the rest of this paper we restrict our attention to common knowledge and fact publication. Common knowledge is a fairly basic notion in people's everyday life. Clark and Marshall [CM] show that it is used fairly extensively in interpersonal communication in natural language, e.g. the term "the president" assumes common knowledge of which person is being referred to. The custom of shaking hands to seal an agreement essentially corresponds to making the agreement common knowledge. Common knowledge is also inherent in the notion of conventions. Something cannot be a convention among a group of people if it is not common knowledge to them. Having common knowledge of a large number of facts allows for better and shorter communication. It seems quite reasonable that two branches of a bank making transactions over a computer network will not seal a million dollar transaction before the transaction is common knowledge.

In the next section, we first show that if communication is not guaranteed (i.e., messages might not reach their destinations), then common knowledge is not attainable. We then show that, even if communication is guaranteed, common knowledge is not attainable if we cannot guarantee simultaneous actions. We introduce various relaxations of common knowledge that are attainable in many cases of interest. We describe in what sense these notions are appropriate, and discuss their relationship to each other. We conclude with a discussion of the role of knowledge in distributed systems.

We think of the processors as state machines with *clocks*, where a clock is a monotone increasing function of real time. A processor's state at a given time is determined by its initial state, its clock time, and its *message history*: the sequence of messages it has sent and received and the times (on the processor's clock) in which it sent or received them.

We now consider what it means for a processor to know p, where p is a formula that talks about the real world and the processors' knowledge of it. We assume some consistent notion of truth of formulas, without going into details of semantics. A processor has some initial information, that we assume to be (forever) true, and it acquires additional information through message exchange. It then deduces what it knows using this information and some set of rules (e.g. the axioms of [Le]). These rules are required to be monotonic in the information and in time, meaning that if a formula p is deducible using the rules given a certain set of information at time t on a processor's clock, then it will still be deducible from any superset of that information at t or any later time. The rules must also be *sound*, i.e., whenever the information is true, then so are any conclusions drawn from it according to the rules. Moreover, we assume that processors do not forget, so information is never lost. Thus, once a processor deduces p, it can deduce p at any later time. Processor i knows p exactly if it can deduce p from its information, using the rules. In this case $K_i p$ is true. Finally, we assume, for now, that that all processors are *honest*, so that a processor may send a message stating p only if it knows p.

From our assumptions above, it follows that the system will always be in a *knowledge consistent* state: whenever $K_i p$ is true, p is true (i.e., the axiom $K_i p \supset p$ is satisfied). Notice that all the facts that can be known are *stable*: facts that, once true, remain true. This is a weaker restriction than it may seem, since given any fact q, facts such as "q held sometime in the past", or "q holds at time t on i's clock", "q holds throughout time interval I", are all stable. It is not reasonable for a processor i to send a message to processor j stating that p is true, if p might become false by the time the message reaches j. It is reasonable for i to send a message stating that p is true at time t on its clock, however, or even a message stating that p is true, and will remain true throughout the future (this, in fact, can be accepted to be the default meaning of a message stating simply p).

A *protocol* is a (possibly nondeterministic) distributed algorithm. A protocol essentially determines what messages can be sent by the processors, as a function of their internal state. Even if a protocol is deterministic, the system may behave nondeterministically, due to uncertainty in message delivery times and to the possibility that messages might fail to be delivered. A *run* of the protocol is a particular execution of the protocol.

We say that a processor i *supports* Cp, if i knows all the $K_{i_1} K_{i_2} \cdots K_{i_n} p$ formulas that constitute Cp (see introduction). A system has *attained* Cp if all its members support Cp.

Lemma 1: A processor i supports Cp iff the system has attained Cp.

Proof: The 'if' direction follows from the definition. For the other direction, assume, to the contrary, that i supports Cp and j does not. There is a formula $q = K_{i_1} K_{i_2} \cdots K_{i_n} p$, such that j does not know q, but i claims to know $K_j q$. The system is therefore not knowledge consistent, a contradiction. □

A *protocol for attaining common knowledge* is a protocol with the following property: if some processor knows a fact p, it can initiate the protocol, and is guaranteed that the system will then eventually attain Cp.

A fact p is said to be *C-undetermined* in a system at a given point in time if, without any further communication in the system, Cp will never hold.

We are now ready to investigate fact publication in distributed systems. Following the coordinated attack example, we first consider systems in which communication is not guaranteed.

Theorem 1: There is no protocol for attaining common knowledge if communication is not guaranteed. In particular, if q is a C-undetermined fact, Sq holds and communication is not guaranteed, then there is no protocol that is guaranteed to attain Cq.

Proof: Assume, to the contrary, that P is such a protocol. Consider a run of P in which no message sent by any processor is delivered to its destination. If Cq is attained, it is attained without any communication, contradicting q's being C-undetermined. □

We think of the processors as state machines with *clocks*, where a clock is a monotone increasing function of real time. A processor's state at a given time is determined by its initial state, its clock time, and its *message history*: the sequence of messages it has sent and received and the times (on the processor's clock) in which it sent or received them.

We now consider what it means for a processor to know p, where p is a formula that talks about the real world and the processors' knowledge of it. We assume some consistent notion of truth of formulas, without going into details of semantics. A processor has some initial information, that we assume to be (forever) true, and it acquires additional information through message exchange. It then deduces what it knows using this information and some set of rules (e.g. the axioms of [Le]). These rules are required to be monotonic in the information and in time, meaning that if a formula p is deducible using the rules given a certain set of information at time t on a processor's clock, then it will still be deducible from any superset of that information at t or any later time. The rules must also be *sound*, i.e., whenever the information is true, then so are any conclusions drawn from it according to the rules. Moreover, we assume that processors do not forget, so information is never lost. Thus, once a processor deduces p, it can deduce p at any later time. Processor i knows p exactly if it can deduce p from its information, using the rules. In this case $K_i p$ is true. Finally, we assume, for now, that that all processors are *honest*, so that a processor may send a message stating p only if it knows p.

From our assumptions above, it follows that the system will always be in a *knowledge consistent* state: whenever $K_i p$ is true, p is true (i.e., the axiom $K_i p \supset p$ is satisfied). Notice that all the facts that can be known are *stable*: facts that, once true, remain true. This is a weaker restriction than it may seem, since given any fact q, facts such as "q held sometime in the past", or "q holds at time t on i's clock", "q holds throughout time interval I", are all stable. It is not reasonable for a processor i to send a message to processor j stating that p is true, if p might become false by the time the message reaches j. It is reasonable for i to send a message stating that p is true at time t on its clock, however, or even a message stating that p is true, and will remain true throughout the future (this, in fact, can be accepted to be the default meaning of a message stating simply p).

A *protocol* is a (possibly nondeterministic) distributed algorithm. A protocol essentially determines what messages can be sent by the processors, as a function of their internal state. Even if a protocol is deterministic, the system may behave nondeterministically, due to uncertainty in message delivery times and to the possibility that messages might fail to be delivered. A *run* of the protocol is a particular execution of the protocol.

We say that a processor i *supports* Cp, if i knows all the $K_{i_1} K_{i_2} \cdots K_{i_n} p$ formulas that constitute Cp (see introduction). A system has *attained* Cp if all its members support Cp.

Lemma 1: A processor i supports Cp iff the system has attained Cp.

Proof: The 'if' direction follows from the definition. For the other direction, assume, to the contrary, that i supports ς_p and j does not. There is a formula $q = K_{i_1} K_{i_2} \cdots K_{i_n} p$, such that j does not know q, but i claims to know $K_j q$. The system is therefore not knowledge consistent, a contradiction. □

A *protocol for attaining common knowledge* is a protocol with the following property: if some processor knows a fact p, it can initiate the protocol, and is guaranteed that the system will then eventually attain Cp.

A fact p is said to be *C-undetermined* in a system at a given point in time if, without any further communication in the system, Cp will never hold.

We are now ready to investigate fact publication in distributed systems. Following the coordinated attack example, we first consider systems in which communication is not guaranteed.

Theorem 1: There is no protocol for attaining common knowledge if communication is not guaranteed. In particular, if q is a C-undetermined fact, Sq holds and communication is not guaranteed, then there is no protocol that is guaranteed to attain Cq.

Proof: Assume, to the contrary, that P is such a protocol. Consider a run of P in which no message sent by any processor is delivered to its destination. If Cq is attained, it is attained without any communication, contradicting q's being C-undetermined. □

As the proof of Theorem 1 shows, demanding that we be guaranteed to attain Cq regardless of the system's behavior is too strong. It is reasonable, however, to hope for a protocol that will succeed in attaining Cq in runs during which a sufficient part of its communication succeeds. Theorem 2 shows that this too is impossible.

Theorem 2: If q is a C-undetermined fact, no run of any protocol ever attains Cq.

Proof: Let P be a protocol. We prove by induction on n that P has no run that attains Cq in which exactly n messages are delivered to their destinations upto (and including) the time Cq is attained. The case $n = 0$ follows by assumption, since q is C-undetermined. Suppose we have proved the result for $n = k$, and assume that P has a run r that attains Cq using $k+1$ messages. Let t_C be the (say) real time in which Cq is first attained in r. Let m_r be the last message that was delivered in r no later than t_C (or one of them, in case of a tie). Let i_r be m_r's sender, and let t_r be the time at which m_r is delivered. Denote by r^- a run of P that behaves exactly like r until t_r, except that m_r is not delivered, and in which no message is delivered after t_r. Since i_r's information at time t_C is identical in r and in r^-, and i_r supports Cq at t_C in run r, it follows that processor i_r will also support Cq at t_C in r^-. By Lemma 1, it follows that Cq is attained at time t_r in r^-, contradicting the induction hypothesis. □

Returning to the "coordinated attack" problem, recall that the generals required common knowledge of the attack before attacking. Theorem 2 shows that they have no hope of that. The reader may suspect that something in our definitions, or in the statement of the notion of agreement required in the "coordinated attack" problem is behind this result Perhaps, if the generals were not required to worry so much about their states of knowledge, there might be a protocol that would ensure that they attack only in tandem, and that does allow them to attack under some circumstances. This is not the case. A slight extension of the proof of Theorem 2 proves:

Proposition 2: If communication is not guaranteed, any protocol that guarantees that if any party attacks, they both attack, is a protocol in which necessarily neither party attacks (!).

The previous results show that, in a strong sense, common knowledge is not attainable in a system in which communication is not guaranteed. However, even when communication *is* guaranteed, common knowledge can be elusive. To see this, consider a system consisting of two processors R2 and D2, connected by a communication link. Assume that their clocks run at the same rate, but might not be showing the same times at a given instant. Assume that communication (delivery) is guaranteed, and that furthermore it is (say commonly) known that any message sent from R2 to D2 reaches D2 either immediately or after exactly ϵ seconds. At time t_R on R2's clock, R2 sends D2 a message stating that m is true. D2 receives the message at time t_D on his own clock. (Recall that we are assuming that m is a stable fact.) D2 doesn't know m initially. How does $\{R2, D2\}$'s state of knowledge of m change with time?

R2 cannot be sure that D2 knows m before $t_R + \epsilon$. D2, on the other hand, cannot be sure that R2 knows D2 knows m until $t_D + \epsilon$ (because the message could have arrived immediately, and R2 waits until $t_R + \epsilon$ before deciding that D2 knows m). R2 cannot be sure that D2 knows R2 knows D2 knows m before $t_R + 2\epsilon$ (since the message could have been delivered exactly ϵ seconds after being sent, and D2 waits until $t_D + \epsilon$, ϵ seconds after it arrives). In other words, if we denote R2's knowledge by K_R and D2's knowledge by K_D, we have m true at t_R, $K_R K_D m$ true at $t_R + \epsilon$ (and no sooner), and $K_R K_D K_R K_D m$ true at $t_R + 2\epsilon$ (and no sooner). This argument can be easily extended to show that $(K_R K_D)^n m$ holds at $t_R + n\epsilon$ and no sooner(!). Since

$$E^{2n} m \supset (K_R K_D)^n m \supset E^{2n-1} m,$$

it follows that $E^{2n+1} m$ will not hold before $t_R + n\epsilon$. (It will hold by $t_R + (n+1)\epsilon$, though.) In order to have Cm, all formulas of the form $E^k m$, $k \geq 1$, must be true. From the above analysis it is clear that m will never become common knowledge.

Now consider what would happen if R2 and D2 shared the same clock (or, equivalently, if it were common knowledge that their clocks show the same time at any given instant). R2 could send D2 the following message: "I am sending you this message at time t_R, knowing it will reach you by $t_R + \epsilon$ at the latest; m is true." Since they commonly know that their clocks read the same time, m would become common knowledge at time $t_R + \epsilon$.

What is the essential difference between these two situations? It seems that what made achieving common knowledge easy in the latter case was the possibility of making the transition from non-common knowledge to common knowledge simultaneously. The impossibility of doing so in the former case was the driving force behind the extra cost in time incurred in attaining every extra level of E-knowledge. Indeed, the following theorem confirms this intuition.

Lemma 2: If Cp is to be attained, all processors must start supporting it simultaneously.

Proof: Follows trivially from Lemma 1. □

Simultaneity is said to be attainable in a system if there is a protocol that guarantees that all members of the system will execute a prescribed action, such as flipping a specific bit from 0 to 1, simultaneously (cf. [DHS]). Lemma 2 essentially states that for common knowledge to be attainable in a system, simultaneity must be attainable in the system.

Lemma 3: If clocks are not initially identical, and there is an uncertainty in message delivery times, then simultaneity cannot be guaranteed.

Proof: Follows from Theorem 4 of [DHS]. □

Theorem 3: If communication is guaranteed, but clocks are not initially identical, and there is an uncertainty in message delivery times, then there is no protocol for attaining common knowledge.

Proof: Immediate from Lemmas 2 and 3. □

Theorems 1–3, roughly speaking, say that attaining common knowledge is practically impossible in actual distributed systems. In such systems, we have the following situation: a fact p can be known to an individual without being common knowledge, or it can be common knowledge (in which case that individual also knows p), but there is no way of getting from the first case to the second. As long as we play according to the rules, that is.

Let us take a close look at the state of knowledge R2 and D2 have once the message m is sent. R2 and D2 commonly know that any message sent between them will arrive at most ϵ seconds after it is sent. Initially, R2 knows m. But, in fact, he knows more. He knows that within ϵ, both of them will know m. But he knows even more: within ϵ they will both know that within another ϵ they will both know m. This argument can be continued, and this leads us to the notion of *ϵ-common knowledge* denoted C^ϵ.

In order to define C^ϵ we need to introduce time. Let $\bigcirc$ correspond to "one time unit later", and $\bigcirc^\epsilon$ be "ϵ time units later". C^ϵ is defined by:

$$C^\epsilon p \equiv p \wedge \bigcirc^\epsilon Ep \wedge \cdots \wedge (\bigcirc^\epsilon E)^n p \wedge \cdots$$

So, in the preceding protocol R2 and D2 have ϵ-common knowledge of m *as soon as the message is sent*.Similarly, if it were common knowledge that a message from R2 to D2 is delivered after somewhere between δ and $\delta + \epsilon$ time units, the contents of the message would become C^ϵ exactly δ time units after the message was sent.

Communication in a distributed system is called *ϵ-close* if all copies of a message sent to all processors in the system are guaranteed to arrive within ϵ time units of one another.

A situation where ϵ-common knowledge is relevant is in the *broadcast* model of communication: all messages are sent to all processors, and it is (commonly) known that communication is ϵ-close. Every message is ϵ-common knowledge once it reaches any processor.

Notice that there is an interesting difference between C and C^ϵ. Whereas common knowledge is a static state of knowledge, that can be true of a point in time irrespective of its past or future, ϵ-common knowledge is a notion that is essentially temporal. Whether or not it holds at a point in time depends on the system's behavior in the future. [1]

Now consider what would happen if R2 "breaks the rules", and uses the following (illegal) "eager protocol": instead of sending D2 the message m, R2 sends D2 the message "Cm", and immediately starts supporting Cm. D2 behaves normally, i.e., supports Cm when it receives the message.

If the message Cm reaches D2 immediately, Cm is attained, and everything is fine. In the worst case, the message might arrive ϵ seconds later. So, for ϵ seconds, R2 may support Cm, when, in fact,

---

[1] This may, in principle, restrict the applicability of C^ϵ, e.g. in systems that go down periodically. However, we will see uses for C^ϵ for which this will not be the case.

Cm is false! [2] However, from time $t_R + \epsilon$ on, they will, in fact, have attained Cm. In a sense, Lemma 2 says that attaining common knowledge requires a certain kind of "natural birth"; it is not possible to attain it consistently unless simulatancity is attainable. But if one is willing to loosen the honesty requirement, and to give up knowledge consistency (abandon the $K_i p \supset p$ axiom) for short intervals of time, common knowledge can be attained. In the remainder of the paper, we change our model slightly, and allow processors to send a message stating Cp when they only know p. This implies that the system is no longer guaranteed to be knowledge consistent, and our previous results, which assume knowledge consistency, do not necessarily hold.

Notice that in the above protocol, R2 essentially acts under the assumption that the system will behave well during a short time interval after the message is sent. He commits to Cm at a point where he has no reason to believe that Cm actually holds. If the system crashes between t_R and $t_R + \epsilon$, it will crash in a state in which R2 and D2's knowledge is inconsistent. So, in fact, R2 is taking a "leap of faith" in using this protocol. The "eager protocol" can be generalized to an arbitrary system: A processor is allowed to send messages stating Cp if it knows p. When a processor does that, it starts supporting Cp at the first instant at which, according to its knowledge, it is possible for any other processor to receive the message.

We have established the fact that common knowledge cannot be attained in a practical distributed system without some risk. Using the "eager protocol" for attaining common knowledge is an example of the kind of risk taking that is required to overcome this hurdle. Using this protocol, the system is in a knowledge inconsistent state during an ϵ time interval, but this state "soon becomes consistent".

Lemma 4: If the eager protocol is used in a broadcast model in which communication is ϵ-close, then Cp is attained at most ϵ time units after the first processor starts supporting Cp.

Proof: Since the processors start supporting Cp when they receive a message stating p, and they are guaranteed to receive these messages no more than ϵ time units apart, Cp will be attained. □

We say that a system is *ϵ-knowledge-consistent* iff whenever processor i claims to know p, then p will become true within ϵ time units. ϵ-knowledge consistency is a relaxation of the axiom $K_i p \supset p$ discussed in the introduction. Processor i might claim $K_i p$ without p being true. Note that an ϵ-knowledge consistent system would seem consistent to an interviewer that is capable of querying the processors about their knowledge only one at a time, with queries at least ϵ time units apart. For sufficiently small ϵ, it may be the case that the inconsistency in an ϵ-knowledge consistent system will never be detected from within the system. We say that a distributed system is *internally knowledge consistent* if it is not possible for a processor to detect that the system is not knowledge consistent from within the system.

Theorem 4: A broadcast model in which communication is ϵ-close, and in which the eager protocol is used, is ϵ-knowledge-consistent.

Proof: Assume that processor i claims to know p at time t. i must be able to deduce p. This deduction might rely on a number of facts of the form Cq, for which only $C^\epsilon q$ holds. However, by Lemma 4, at time $t + \epsilon$ Cq will hold for all these q's. Since the rules that processor i is using are monotone and sound, it follows that p will hold at $t + \epsilon$. □

It is often the case that a distributed system is *ϵ-insensitive:* within the system it is not possible to determine the temporal relationship between two events that occur in different sites of the system less than ϵ time units apart. A typical example of a system that is ϵ-insensitive is a distributed system in which the known bounds on the uncertainty in message delivery times are more than ϵ greater than the actual uncertainty (cf. [DHS]).

Theorem 5: A distributed system that is ϵ-knowledge consistent and is ϵ-insensitive, is internally knowledge consistent.

Proof: Follows directly from the definitions. □

Theorem 5 offers one explanation of how people, not being capable of simultaneity, can function under the assumption that they do achieve common knowledge. When people are copresent with the occurrence of a fact, the ϵ is so small that their assumption of common knowledge can never be contradicted.

[2] Notice that the worst case period of this knowledge inconsistency can be reduced further by R2's claiming Cm only at $t_R + \epsilon/2$.

3. Asynchronous communication

In asynchronous communication, the system guarantees only that every message sent will *eventually* reach its destination. No bound on transmission time is known to exist. Consider R2 and D2's state of knowledge when they communicate in such a network. When R2 sends D2 the message m, he knows only that D2 will eventually receive the message. Without any confirmation from D2, R2 will never know that D2 has received m. So, in asynchronous communication, $E^2 m$ is never attained unless messages are acknowledged. However, R2 knows a little more than just that D2 will eventually know m. He knows that when D2 receives m he will know that R2 knows that he will eventually know m, etc.

This state of knowledge, where, roughly speaking, m is true, and eventually everyone will know that m is true and that eventually everyone will know ... is very common in asynchronous systems. We now define a weak notion of knowledge that corresponds to this state, which we call *eventual common knowledge*.

We adopt the temporal logic symbol $\Diamond$ to stand for "eventually". $\Diamond$-*common knowledge* (read *eventual common knowledge*), denoted by $C^\Diamond$, is defined as follows:

$$C^\Diamond p \equiv p \wedge \Diamond Ep \wedge \cdots \wedge (\Diamond E)^n p \wedge \cdots$$

In R2 and D2's case, m becomes $C^\Diamond$ at the instant it is sent. In (guaranteed) asynchronous broadcast communication, every message becomes $\Diamond$-common knowledge from the instant it is sent.

Actual experience tends to support the claim that sending a message to a mailing list on a large network guarantees at best achieving $\Diamond$-common knowledge. In some versions of the "Byzantine Agreement" problem, the state of knowledge that is sought is $C^\Diamond$ (cf. [DDS]).

One can imagine using the eager protocol for common knowledge in an asynchronous system. Obviously, this can result in the system being in inconsistent states for long periods of time. However, we can define a rather weak notion of consistency that we call $\Diamond$-*knowledge consistency* that will be obeyed in such circumstances. We say that a system is $\Diamond$-knowledge consistent if whenever individual i claims $K_i p$, p will eventually hold.

Lemma 5: Whenever the eager protocol is used to convert $C^\Diamond p$ into Cp, Cp is eventually attained.

Proof: Since each processor supports Cp when it receives the message stating p, and we are guaranteed that they all eventually receive it, it follows that Cp will be attained once the last of the processors receives the message. □

Theorem 6: A broadcast model in which communication is asynchronous, and in which the eager protocol is used, is $\Diamond$-knowledge-consistent.

Proof: The proof is essentially the same as that of Theorem 4, using Lemma 5. □

4. Common knowledge revisited

The definitions of C, C^ϵ and $C^\Diamond$ are very similar. In this section we look for the unifying elements, exploit our experience from the previous sections, and come up with a general technique for generating notions of common knowledge appropriate for arbitrary network communication behaviors. In particular, it will be possible to analyze the state of knowledge attained when messages are sent in a system where communication is not guaranteed.

For the purpose of this section, we will assume that knowledge can be "factored" out of infinite conjunctions, in particular, $\bigwedge_n Ep_n \supset E\left(\bigwedge_n p_n\right)$ (where the conjunction may be infinite). With this assumption, it is easy to see that $Cp \supset ECp$, so that if p is common knowledge, then everyone knows that it is. In fact, $Cp \equiv ECp$, and Cp can be thought of as a fixpoint of the equation

$$Cp \equiv p \wedge ECp.$$

This view, in some sense, describes how it is that certain notions of agreement, and certain scenarios in everyday life (e.g. copresence) involve the infinite conjunction that constitutes Cp: they simply involve a state of knowledge that is a fixpoint of the E operator, in which p is true. With our above assumption, common knowledge can be characterized by the following three axioms (cf. [Le], and the analogous axioms for the PDL $*$ operator in [KP]):

(1) The "fixpoint" axiom:

$$Cp \supset p \wedge ECp.$$

(2) The "induction" axiom:

$$\Big(p \wedge C(p \supset Ep)\Big) \supset Cp.$$

3) The "consequence closure" axiom:

$$\Big(Cp \wedge C(p \supset q)\Big) \supset Cq.$$

The induction axiom states that if p is true, and t is common knowledge that whenever p is true, Ep s true, then p is common knowledge. It is called an nduction axiom, because using it we can prove by nduction that if p holds, then $E^n p$ holds, for all n. This axiom, in some sense, traces our line of reasoning when we claimed that the dirty children attain ommon knowledge of **m**, and when we claimed that he strong notion of agreement that people attain n signing a contract, or by shaking hands, implies ommon knowledge of the agreement.

As expected, the notions of C^ϵ and $C^\diamond$ can e characterized in a similar fashion. If we denote $\supset^\epsilon E$ by E^ϵ (respectively, $\diamond E$ by $E^\diamond$), then axioms 1)–(3) characterize $C^\epsilon p$ (resp. $C^\diamond p$), if we replace E y E^ϵ and C by C^ϵ (resp. E by $E^\diamond$ and C by $C^\diamond$).[3] Recall that if it is common knowledge that every nessage broadcast reaches all processors within ϵ ime units (resp. eventually), then any such mesage, once sent, is ϵ-common knowledge (resp. $\diamond$-ommon knowledge). In fact, the induction axiom ells us that we do not need the message delivery beavior of the system to be common knowledge for his to hold. It suffices that it be C^ϵ (resp. $C^\diamond$).

Using these ideas, we can now define a number f other variants of common knowledge. For example, suppose we have a system where communication s not guaranteed, but it is common knowledge that ny message sent is likely to arrive, and delivery, vhen successful, is immediate. Let p be "the message m has been sent". We have $C(p \supset$ "Likely" Ep cf. [HR]). The notion of common knowledge that orresponds to this is called *likely* common knowledge, denoted C^L. If we denote "Likely Ep" by $E^L p$, $C^L p$ is defined by:

$$C^L p \equiv p \wedge \bigwedge_k (E^L)^k p.$$

Clearly, $C^L p \supset E^L C^L p$. However, if we denote "Likely q" by Lq, it is not, in general, the case that ikelihood satisfies a consequence closure axiom simlar to (3) (cf. [HR]). C^L therefore does not satisfy axiom (3). As a consequence, C^L does not satisfy the induction axiom. Rather, it satisfies the *weak* induction axiom:

$$(2) \quad \Big(p \wedge C(p \supset E^L p)\Big) \supset C^L p.$$

Here, $p \supset E^L p$ needs to be common knowledge for p to imply $C^L p$. Note that if it is common knowledge that a message is likely to arrive immediately, or if the message itself states this fact, then the weak induction axiom suffices for a processor to prove that the message is likely-common knowledge.

Continuing along these lines, you can now define notions of common knowledge corresponding to your favorite conditions for attaining Ep. Conditions that come to mind are "Likely within ϵ", "Likely eventually", "With probability π", "Unlikely", and so on. The fixpoint axiom and the weak induction axiom will hold for all of them, and whenever the condition satisfies a consequence closure axiom, the corresponding common knowledge will satisfy the (strong) induction axiom and the consequence axiom.

Corresponding to each such notion of common knowledge, one can imagine the use of an "eager protocol" to convert this "conditional" common knowledge to common knowledge. Along with it comes a notion of "conditonal"-knowledge consistency, that describes the inconsistency involved in using such an eager protocol.

5. Time stamping: a relativistic alternative

Many distributed systems work in phases (with or without a common clock). In other systems, synchronization algorithms (cf. [HSSD]) keep the processors' clocks within some bound of each other, and there are known bounds on message delivery times. In such systems, it is often natural to speak of the processors' state of knowledge at the begining of phase $t+1$, or at time T on their clock. Since time T on my clock may be quite different from time T on your clock, time becomes a private matter, and we are lead to a relativistic notions of knowledge.

Consider the following scenario: R2 knows that R2 and D2's clock differ by at most δ, and that any message R2 sends D2 will arrive within ϵ time units. Not having read the initial part of this paper, and attempting to attain Cm, R2 sends D2 the following message:

[3] This forms a complete characterization of C^ϵ and $C^\diamond$.

"It is now t_R on my clock. This message will reach you by $t_R + \epsilon + \delta$ on both of our clocks. m is true"

Let us denote $t_R + \epsilon + \delta$ by T_0. Now, at T_0, R2 would like to claim that they have attained common knowledge of m. Have they? Well, we know by now that they probably haven't, but let us analyze their situation. First, we need to introduce a relativistic formalism for knowledge, that we call *time-stamped* knowledge: We denote "at time T on his clock, i knows p" by $K_i^T p$. T is said to be the *time-stamp* associated with this knowledge. Furthermore,

$$E^T p \equiv \bigwedge_i K_i^T p.$$

$E^T p$ corresponds to everyone knowing p individually at time T on their own clocks. If we define $p =$ "R2 sent the above message to D2", we have $p \supset E^{T_0}$. It is now, of course, natural to define the corresponding notion of common knowledge, C^T, which we call *time-stamped* common knowledge:

$$C^T p \equiv p \wedge \bigwedge_k (E^T)^k p.$$

So, R2 and D2 have time-stamped common knowledge of m with time stamp T_0. It is easy to check that the condition "at time T on i's clock" satisfies a consequence axiom, and it therefore follows that C^T satisfies axioms (1)-(3) of the previous section.

It is interesting to investigate how C^T relates to C, C^ϵ, and $C^\diamond$. This is a good example of how looking at the conditions that define notions of common knowledge gives insight into their meaning, and the relations between these notions. Not surprisingly, the relative behavior of the clocks in the system plays a crucial role in determining the meaning of C^T.

Theorem 7:

(a) If it is common knowledge that all clocks show identical times, then at T on any processor's clock, $C^T p \equiv Cp$.

(b) Assume that the clocks all run at the rate of real time. If it is ϵ-common knowledge that all clocks are within ϵ time units of each other then, at T on any processor's clock, $C^T p \supset C^\epsilon p$.

(c) If it is $\diamond$-common knowledge that all clocks are monotone increasing with no bound, meaning that every clock eventually displays every possible reading, then $C^T p \supset C^\diamond p$. □

Theorem 7 demonstrates the conditions that allow interchanging the relativistic C^T with C, C^ϵ and $C^\diamond$. It also sheds light on the connection between the absolute time notions of common knowledge and the relative behavior of clocks in the system. Note that a weak converse of Theorem 7 holds as well. Suppose we allowed the processors to set their clocks to a common agreed upon time T, when Cp (resp. $C^\epsilon p$, $C^\diamond p$) is attained. Then it is easy to see that if Cp (resp. $C^\epsilon p$, $C^\diamond p$) is attainable, then so is $C^T p$.

6. Conclusions

Knowledge plays a fundamental role in distributed systems. Viewing communication in a network and goals of protocols through the "knowledge perspective" in many cases clarifies a number of issues. We have investigated several notions of group knowledge. We define a hierarchy of these notions, and show that it is strict in the case of distributed systems. A general taxonomy of distributed systems is required, which will specify the time and communication complexities involved in changing the state of knowledge of a fact from one level in our knowledge hierarchy to a higher one.

The weakest notion in the hierarchy is that of implicit knowledge, which corresponds to the knowledge a single individual would have, if he knew what all the members of the group know. This notion can be appropriate for worst case analysis of what an adversary group, such as the KGB, CIA, or Byzantine traitors, might know at a given point in time. This notion is also useful in the verification of cryptographical protocols, where proving that a group is ignorant of a fact amounts to showing that its members don't even have implicit knowledge of the fact.

The strongest state of knowledge we discussed is common knowledge. It is, in a sense, the state of knowing a fact and knowing that the other members of the group have the same knowledge regarding the fact as you do.

One of our main results shows that common knowledge is not attainable in real world systems. However, various weaker notions, such as ϵ-common knowledge, time-stamped common knowledge, and likely-common knowledge, are often attainable, appropriate substitutes for common knowledge. The

use of an "eager" protocol, that treats ϵ-common knowledge as if it were common knowledge, results in a system that is inconsistent, but is ϵ-knowledge-consistent. For small ϵ, this inconsistency may go undetected, and the system may be indistinguishable from a consistent one.

We believe that the "knowledge perspective" will give us a better understanding of distributed communication protocols and distributed consensus protocols, and provides a clear formalism in which to specify and verify them.

Acknowledgements: Many people commented on different versions of this work. Of special value were discussions with Dave Chelberg, Steve Deering, Danny Lehman, Tim Mann, Ray Strong and Moshe Vardi. Ron Fagin, Dick Gabriel, Yoni Malachi and Joe Weening were very helpful in proofreading early versions, and in the typesetting of the paper.

Bibliography

[Au] R. J. Aumann, Agreeing to disagree, *Annals of Statistics*, **4:6**, 1976.

[Ba] J. Barwise, Scenes and other situations, *Journal of Philosophy*, Vol. LXXVIII, No. 7, 1981, 369-397.

[CM] H. H. Clark and C. R. Marshall, Definite reference and mutual knowledge, in A. K. Joshi, B. L. Webber, and I. A. Sag (Eds.), *Elements of Discourse Understanding*, Cambridge University Press, 1981.

[DDS] D. Dolev, C. Dwork, and L. Stockmeyer, On the minimal synchronization needed for distributed consensus, *FOCS*, 1983, pp. 369-397.

[DHM] D. Dolev, J. Y. Halpern, and Y. Moses, Cheating husbands and other stories: a case study of common knowledge, unpublished manuscript, 1984.

[DHS] D. Dolev, J. Y. Halpern, and H. R. Strong, On the possibility and impossibility of achieving clock synchronization. *STOC* 1984, pp. 504-511.

[DS] D. Dolev and H. R. Strong, Byzantine agreement, *IBM Research Report RJ* 3714, 1982.

[HR] J. Y. Halpern and M. O. Rabin, A logic to reason about likelihood, STOC, 1983, pp. 310-319.

[HSSD] J. Y. Halpern, B. Simons, and H. R. Strong, D. Dolev, Fault tolerant clock synchronization, *PODC* 1984.

[Hi] J. Hintikka, Knowledge and Belief, *Cornell University Press*, 1962.

[KP] D. Kozen and R. Parikh, An elementary proof of the completeness of PDL, *Theoretical Computer Science*, **14:1**, 1981, pp. 113-118.

[Le] D. Lehman, Knowledge, common knowledge, and related puzzles, *PODC* 1984.

[Li] B. Lindsay et al., Notes on Distributed Databases, *IBM Research Report RJ* 2571, 1979.

[LF] L. Lamport and M. Fischer, Byzantine generals and transaction commit protocols, *SRI International Report*, 1982.

[LM] L. Lamport and P. M. Melliar-Smith, Synchronizing clocks in the presence of faults, *SRI International Report*, 1982.

[Mc] J. McCarthy, Lecture Notes 1968 — 1974.

[Mo] R.C. Moore, Reasoning about knowledge and action, *Artificial Intelligence Center Technical Note* 191, SRI International, 1980.

[PSL] M. Pease, R. Shostak, and L. Lamport, Reaching agreement in the presence of faults, *JACM*, **27:2**, 1980, pp. 228-234.

[S] D. Schwabe, Formal specification and verification of a connection establishment protocol, *Seventh IEEE Data Communications Symposium*, 1981, pp. 11-26.