Two Generals' Problem

From Wikipedia, the free encyclopedia

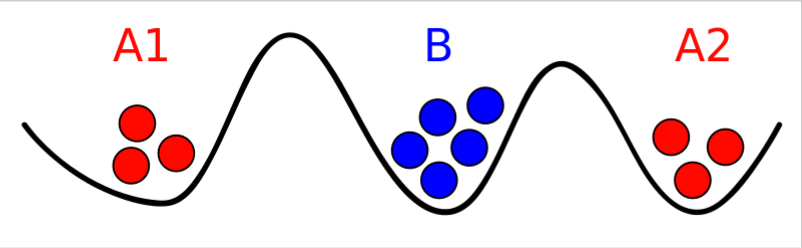
In computing, the **Two Generals Problem** is a thought experiment meant to illustrate the pitfalls and design challenges of attempting to coordinate an action by communicating over an unreliable link. It is related to the more general Byzantine Generals Problem (though published long before that later generalization) and appears often in introductory classes about computer networking (particularly with regard to the Transmission Control Protocol where it shows that TCP can't guarantee state consistency between endpoints and why), though it applies to any type of two party communication where failures of communication are possible. A key concept in epistemic logic, this problem highlights the importance of common knowledge. Some authors also refer to this as the **Two Generals Paradox**, the **Two Armies Problem**, or the **Coordinated Attack Problem**.[1][2] The Two Generals Problem was the first computer communication problem to be proved to be unsolvable. An important consequence of this proof is that generalizations like the Byzantine Generals problem are also unsolvable in the face of arbitrary communication failures, thus providing a base of realistic expectations for any distributed consistency protocols.

Contents

Definition

Two armies, each led by a general, are preparing to attack a fortified city. The armies are encamped near the city, each in its own valley. A third valley separates the two hills, and the only way for the two generals to communicate is by sending messengers through the valley. Unfortunately, the valley is occupied by the city's defenders and there's a chance that any given messenger sent through the valley will be captured.

While the two generals have agreed that they will attack, they haven't agreed upon a time for attack. It is required that the two generals have their armies attack the city at the same time in order to succeed, else the lone attacker army will die trying. They must thus communicate with each other to decide on a time to attack and to agree to attack at that time, and each general must know that the other general knows that they have agreed to the attack plan. Because acknowledgement of message receipt can be lost as easily as the original message, a potentially infinite series of messages is required to come to consensus.



Positions of the armies. Armies A1 and A2 need to communicate but their messengers may be captured by army B.

The thought experiment involves considering how they might go about coming to consensus. In its simplest form one general is known to be the leader, decides on the time of attack, and must communicate this time to the other general. The problem is to come up with algorithms that the generals can use, including sending messages and processing received messages, that can allow them to correctly conclude:

> Yes, we will both attack at the agreed-upon time.

Allowing that it is quite simple for the generals to come to an agreement on the time to attack (i.e. one successful message with a successful acknowledgement), the subtlety of the Two Generals' Problem is in the impossibility of designing algorithms for the generals to use to safely agree to the above statement.

Illustrating the problem

The first general may start by sending a message "Attack at 0900 on August 4." However, once dispatched, the first general has no idea whether or not the messenger got through. This uncertainty may lead the first general to hesitate to attack due to the risk of being the sole attacker.

To be sure, the second general may send a confirmation back to the first: "I received your message and will attack at 0900 on August 4." However, the messenger carrying the confirmation could face capture and the second general may hesitate, knowing that the first might hold back without the confirmation.

Further confirmations may seem like a solution—let the first general send a second confirmation: "I received your confirmation of the planned attack at 0900 on August 4." However, this new messenger from the first general is liable to be captured too. Thus it quickly becomes evident that no matter how many rounds of confirmation are made, there is no way to guarantee the second requirement that each general be sure the other has agreed to the attack plan. Both generals will always be left wondering whether their last messenger got through.

Proof

For deterministic protocols with a fixed number of messages

Because this protocol is **deterministic**, suppose there is a **sequence** of a fixed number of messages, one or more successfully delivered and one or more not. The assumption is that there should be a *shared certainty for both generals to attack*.

Consider the last such message that was successfully delivered. If that last message had not been successfully delivered, then one general at least (presumably the receiver) would decide not to attack. From the viewpoint of the sender of that last message, however, the **sequence** of messages sent and delivered is exactly the same as it would have been, had that message been delivered.

Since the protocol is **deterministic**, the general sending that last message will still decide to attack. We've now created a situation where the suggested protocol leads one general to attack and the other not to attack—contradicting the assumption that the protocol was a solution to the problem.

For nondeterministic and variable-length protocols

A **nondeterministic** protocol with a variable message count can be compared to a **finite** tree, where each leaf or branch (node) in the tree represents an explored example up to a specified point.

The roots of this tree are labeled with the possible starting messages, and the branch nodes stemming from these roots are labeled with the possible next messages. Leaf nodes represent examples which end after sending the last message. A protocol that terminates before sending any messages is represented by a null tree.

Suppose there exists a **nondeterministic** protocol which solves the problem. Then, by a similar argument to the **deterministic** example in the previous section, where the one can be obtained from the other by removing all leaf nodes, the **deterministic** protocol must then also solve the problem.

Since the **nondeterministic** protocol is **finite**, it then follows that the protocol represented by the empty tree would solve the problem. Clearly this is not possible. Therefore a **nondeterministic** protocol which solves the problem cannot exist.[3]

Engineering approaches

A pragmatic approach to dealing with the Two Generals' Problem is to use schemes that accept the uncertainty of the communications channel and not attempt to eliminate it, but rather mitigate it to an acceptable degree. For example, the first general could send 100 messengers, anticipating that the probability of all being captured is low. With this approach the first general will attack no matter what, and the second general will attack if any message is received. Alternatively the first general could send a stream of messages and the second general could send acknowledgments to each, with each general feeling more comfortable with every message received. As seen in the proof, however, neither can be certain that the attack will be coordinated. There's no algorithm that they can use (e.g. attack if more than four messages are received) which will be certain to prevent one from attacking without the other. Also, the first general can send a marking on each message saying it is message 1, 2, 3 ... of n. This method will allow the second general to know how reliable the channel is and send an appropriate number of messages back to ensure a high probability of at least one message being received. If the channel can be made to be reliable, then one message will suffice and additional messages do not help. The last is as likely to get lost as the first.

Assuming that the generals must sacrifice lives every time a messenger is sent and intercepted, an algorithm can be designed to minimize the number of messengers required to achieve the maximum amount of confidence the attack is coordinated. To save them from sacrificing hundreds of lives to achieve a very high confidence in coordination, the generals could agree to use the absence of messengers as an indication that the general who began the transaction has received at least one confirmation, and has promised to attack. Suppose it takes a messenger 1 minute to cross the danger zone, allowing 200 minutes of silence to occur after confirmations have been received will allow us to achieve extremely high confidence while not sacrificing messenger lives. In this case messengers are used only in the case where a party has not received the attack time. At the end of 200 minutes, each general can reason: "I have not received an additional message for 200 minutes; either 200 messengers failed to cross the danger zone, or it means the other general has confirmed and committed to the attack and has faith I will too".

History

The Two Generals Problem and its impossibility proof was first published by E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber in 1975 in "Some Constraints and Trade-offs in the Design of Network Communications",[4] where it is described starting on page 73 in the context of communication between two groups of gangsters.

This problem was given the name the *Two Generals Paradox* by Jim Gray[5] in 1978 in "Notes on Data Base Operating Systems"[6] starting on page 465. This reference is widely given as a source for the definition of the problem and the impossibility proof, though both were published previously as above.

References

1. Gmytrasiewicz, Piotr J.; Edmund H. Durfee (1992). "Decision-theoretic recursive modeling and the coordinated attack problem" (http://dl.acm.org/citation.cfm?id=139492.139503). *Proceedings of the first international conference on Artificial intelligence planning systems*. San Francisco: Morgan Kaufmann Publishers: 88–95. Retrieved 27 December 2013.
2. The coordinated attack and the jealous amazons (http://www.dsi.uniroma1.it/~asd3/dispense/attack+amazons.pdf) Alessandro Panconesi. Retrieved 2011-05-17.
3. Kennard, Fredrick. *Thought Experiments: Popular Thought Experiments in Philosophy, Physics, Ethics, Computer Science & Mathematics* (https://books.google.nl/books?id=sX-pCQAAQBAJ). Lulu.com. p. 346. ISBN 9781329003422. Retrieved 15 September 2015.
4. "Some constraints and trade-offs in the design of network communications" (http://hydra.infosys.tuwien.ac.at/teaching/courses/AdvancedDistributedSystems/download/1975_Akkoyunlu,%20Ekanadham,%20Huber_Some%20constraints%20and%20tradeoffs%20in%20the%20design%20of%20network%20communications.pdf) (PDF). Portal.acm.org. doi:10.1145/800213.806523 (https://doi.org/10.1145%2F800213.806523). Retrieved 2010-03-19.
5. "Jim Gray Summary Home Page" (http://research.microsoft.com/~Gray/JimGrayHomePageSummary.htm). Research.microsoft.com. 2004-05-03. Retrieved 2010-03-19.
6. "Notes on Data Base Operating Systems" (http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=723863). Portal.acm.org. Retrieved 2010-03-19.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Two_Generals%27_Problem&oldid=780382208"

Categories: Distributed computing problems | Theory of computation