

Protocols

Mutual exclusion

n participants, one critical section

Requirements: correctness, absence of deadlocks, fairness

In case of shared memory: Peterson, Bakery (see Herlihy slides)

In case of message passing:

- Central authority

- Distributed agreement

- Token ring

For message passing, we first discuss the problem of unreliable channels, using the reader-writer problem.

The reader-writer problem

Alice produces a set of messages m_0, m_1, \dots and sends them to Bob.

Bob can only process one message at a time.

So Alice cannot send the message all at once.

Alice wants to make sure all her messages are processed,

→ needs to wait for Bob's acknowledgements

Assumptions about the communication

Messages take some arbitrary time to deliver
(physical delays, recipient is busy → asynchronous communication).

Messages can get lost without notice
(otherwise the solution is easy).

However, we assume that from time to time message to arrive
(channel is lossy, but not completely faulty).

Order of messages is preserved
(can be enforced with timestamps if necessary – sender adds
timestamp/sequence number to message, recipient ignores out-of-order
messages).

Solution: Alternating-bit protocol

Alice and Bob have their own sequence bits s_A and s_B , initially 0.

Alice has a message counter i , initially 0.

Alice sends messages of the form (m_i, s_A) .

Bob sends messages of the form s_B .

The protocol

Alice:

1. Alice sends (m_i, s_A) and repeats this message from time to time.
- 2a. If Alice receives s_A , she increases i and flips s_A , then goes back to step 1.
- 2b. If Alice receives the message $\neg s_A$, she ignores it and continues at 1.

Bob:

1. Bob sends $\neg s_B$ and repeats this message from time to time.
- 2a. If Bob receives (m, s_B) , he processes the message m , then flips s_B and goes back to step 1.
- 2b. If Bob receives $(m, \neg s_B)$, he ignores it and continues at 1.

Correctness

Requirements:

Bob must not skip any message.

Bob must not mistake one message for another.

Informally: Must avoid confusion between two different messages from Alice with the same sequence bit, e.g. $(m, 0)$ and $(m', 0)$.

Invariants:

When Alice sends message m_j , Bob has processes at least messages m_0 to m_{j-1} .

When Bob receives a message (m, s) , all subsequent receptions of messages tagged with s refer to the same message (until another message with $\neg s$ comes along).

One shows that these invariants hold initially and are preserved; correctness follows.

Conclusion

The alternating-bit protocol shows how to simulate a reliable channel on top of an unreliable one.

Therefore, we can from now on assume w.l.o.g. that communication channels are reliable (no message loss, duplication, out-of-order delivery, or manipulation).

The only remaining problem is that communication is asynchronous, i.e. we must assume that message take an arbitrary (but finite!) time to be delivered.

Central authority

One of the participants (or an external party) is appointed as the “leader”.

When a participant wants to enter the critical section, he/she must send a request to the leader and wait for positive response.

Leader will accept the first request and queue the others.

Advantage: low message complexity, suitable e.g. for print server; other participants do not need to know each other.

Disadvantage: single point of failure.

Distributed agreement

If a participant wants to enter the critical section, he sends a request tagged with a timestamp to all other processes and waits until everybody responds with “ok”.

When a participant receives a request, then:

- if he does not currently want to enter the critical section, he responds “ok”;

- if he currently is in the critical section, then he queues the request on a waiting list and responds “ok” once he leaves the critical section;

- if he is waiting to enter the critical section himself, then he compares the timestamp on the message with his own. If the received message has an earlier timestamp, he responds “ok”. Otherwise he treats it like in the previous case.

The importance of agreeing on the time

Suppose that the participants have different clocks. Then mutual exclusion is not guaranteed:

3 participants A, B, C

A sends a request with timestamp 5

B responds with ok

Then B sends a request with timestamp 3

A compares timestamp, responds ok

C responds ok to both A and B

Both A and B enter the critical section.

Common time required!

Solutions for agreeing on time

Time server (but that introduces a common authority...)

“Virtual time” (only relative order of events matters):

Each participants has his own “time”.

Time is increased whenever an action in the protocol is executed.

When receiving a message with timestamp m and m greater than own time, set own time to $m + 1$.

Advantages/Disadvantages??

Token ring

Participants organized in a ring, each participant knows the next.

In the beginning, a leader is elected who gets a “token”.

If the current token holder does not want to enter the critical section, then he passes the token on.

If a participant wants to enter the critical section, then he waits until he gets the token, then enters, and passes the token one after leaving.

Advantages/disadvantages??

Leader-election protocol

The following protocol is due to [Dolev, Klawe, Rodeh \(1982\)](#).

The protocol consists of n participants (where n is a parameter). The participants are connected by a ring of unidirectional message channels. Communication is asynchronous, and the channels are reliable. Each participant has a unique ID (e.g., some random number).

Goal: The participants communicate to elect a “leader” (i.e., some distinguished participant). The protocol shown here ensures low communication overhead ($\mathcal{O}(n \log n)$ messages; most naïve protocols have quadratic message overhead).

Leader-election protocol

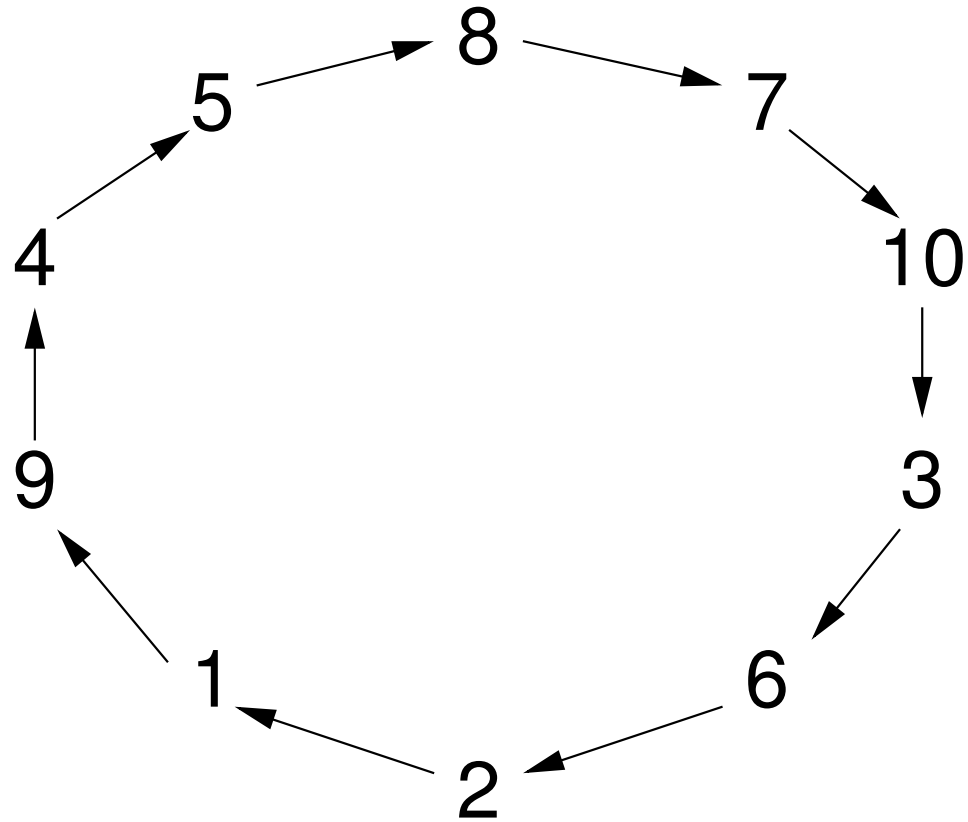
Participants are either **active** or **inactive**. Initially, all participants are *active*.

The protocol proceeds in **rounds**. In each round, at least half of the participants will become inactive. (As a consequence, there are at most $\mathcal{O}(\log n)$ rounds.

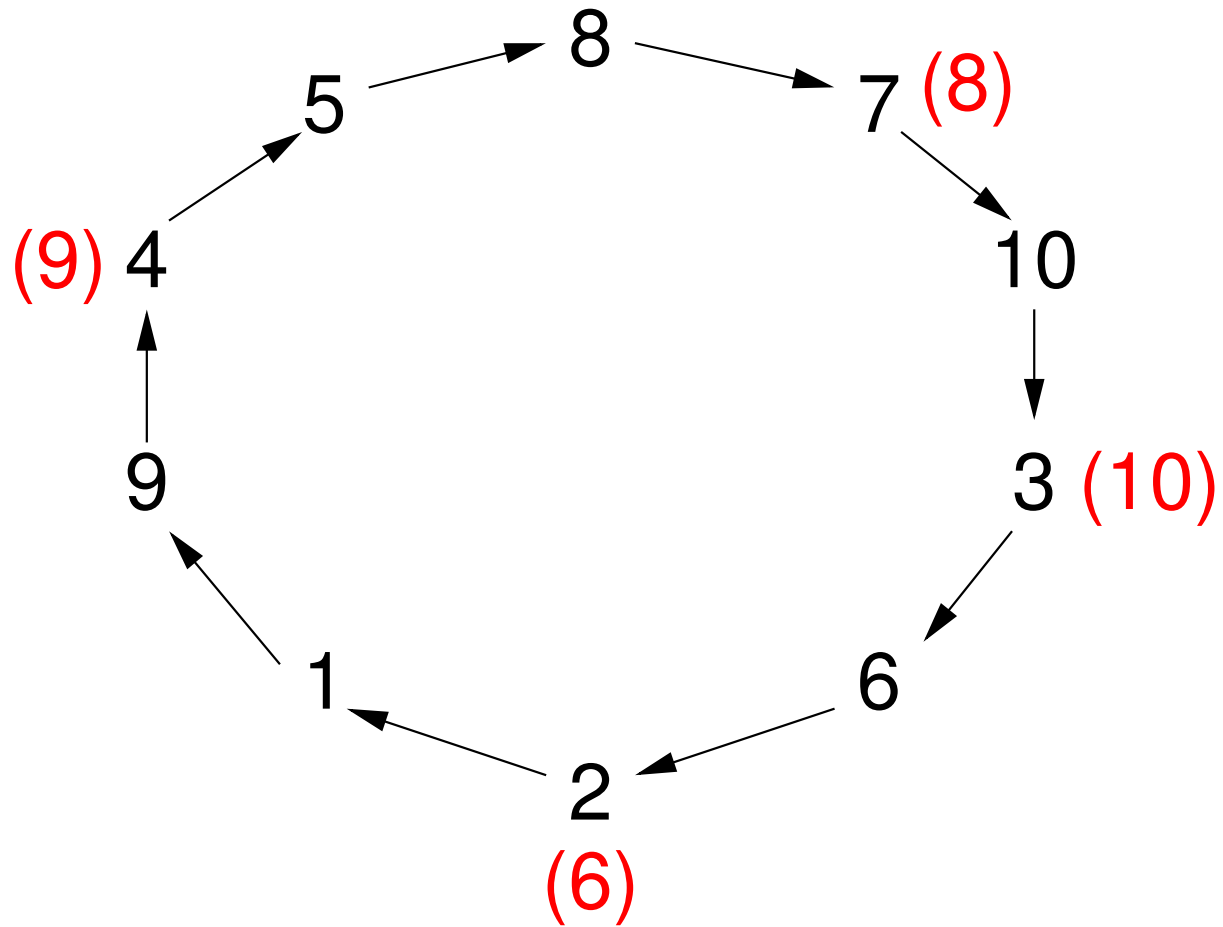
In each round every active participant receives the numbers of the two nearest active participants (in incoming direction). A participant remains active only if the value of the nearest neighbour is the largest of the three. In this case, the participant adopts this largest number as its own.

The last remaining active participant is declared the leader.

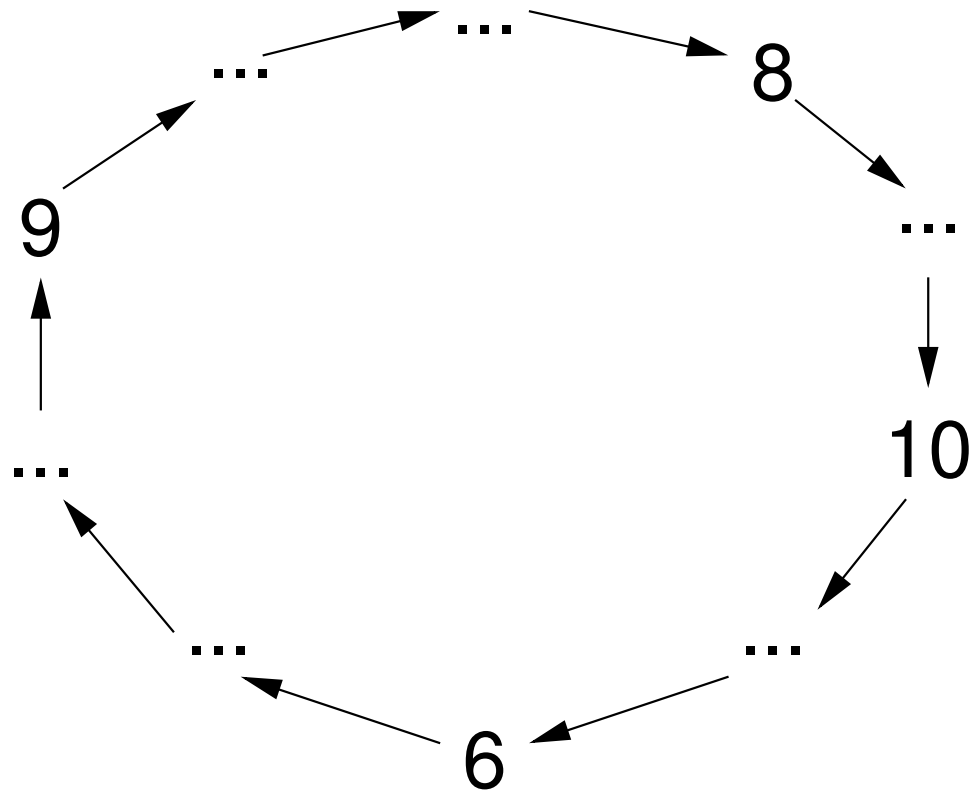
Leader Election: Example



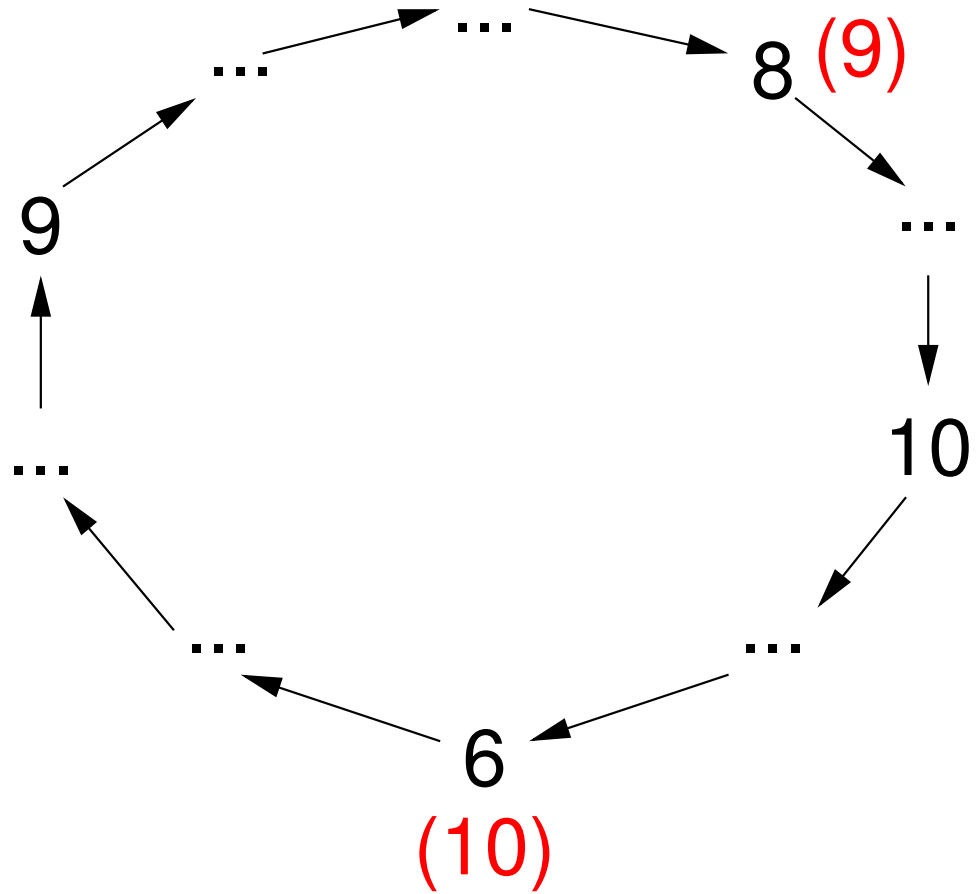
Leader Election: First round



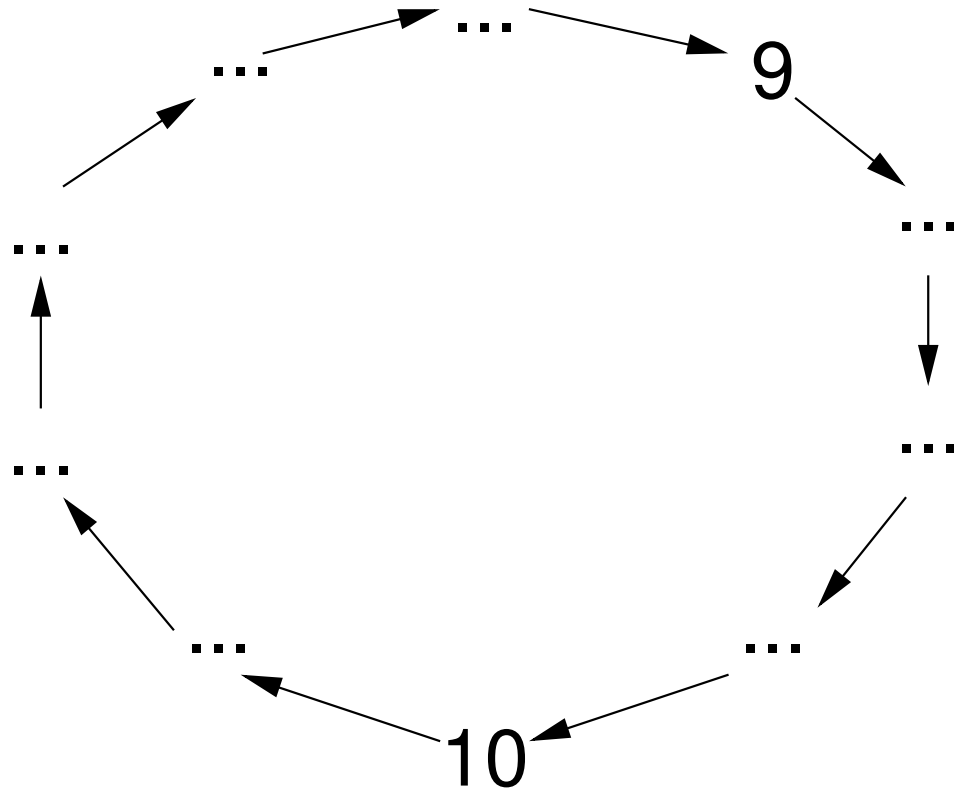
Leader Election: Result of the first round



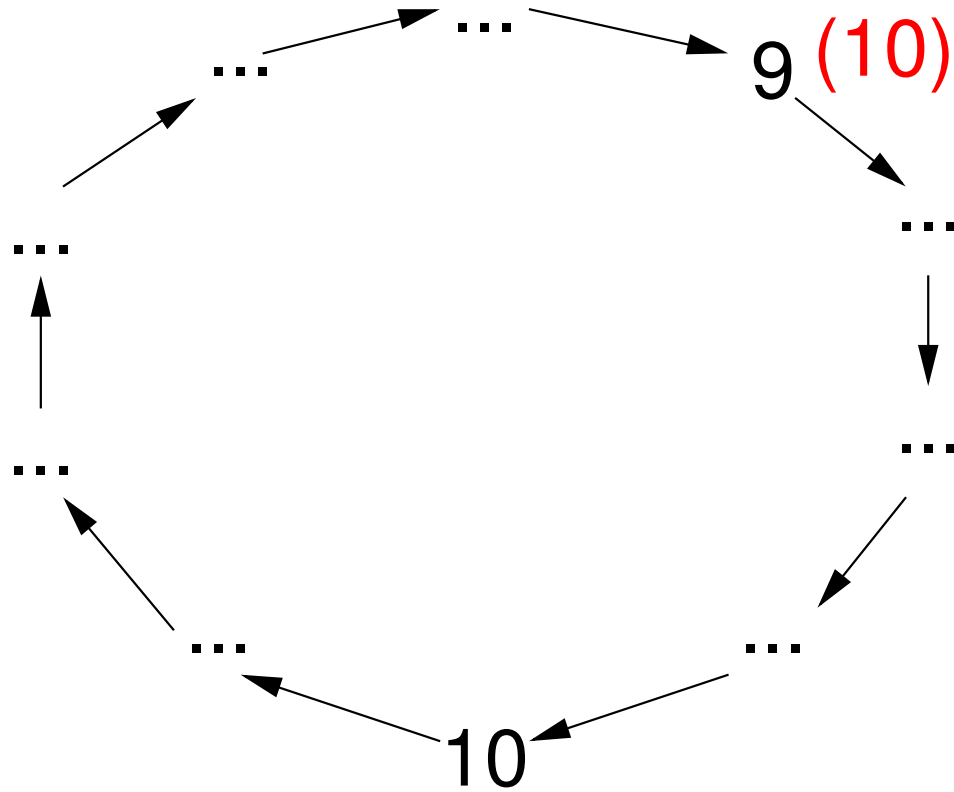
Leader Election: Second round



Leader Election: Result of the second round



Leader Election: Third round



Leader Election: Final result

