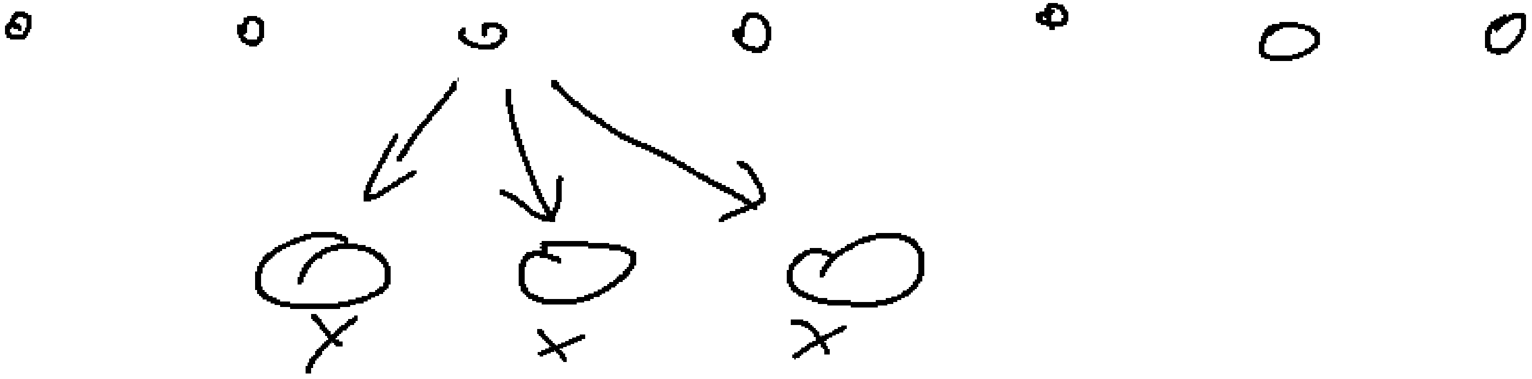
Distributed Computing
Informatyka Algorytmiczna 2021

Prof. Mirosław Kutyłowski

Clients & servers, PAXOS

Client-server model

- Message passing
- Server executes commands of the client
- Single client for a server: commands with sequence numbers solve the problem



Multiple clients, multiple servers

Inconsistency problem: the same order executed on different servers may lead to different results

Client A: x:= x+1

Client B: x:=2*x

Server 1: initially x=0, orders: A (x=1), then B (x=2)

Server 2: initially x=0, orders: B (x=0), then A (x=1)

Serializer

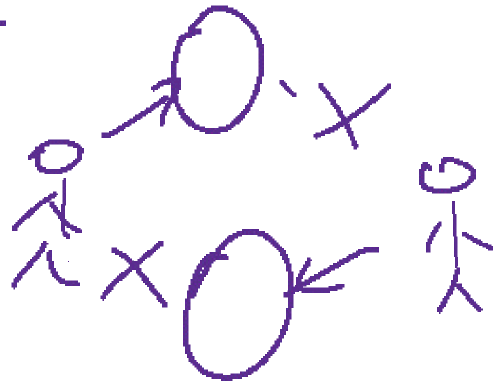
A straightforward but not scalable:

Algorithm 15.9 State Replication with a Serializer

1: Clients send commands one at a time to the serializer
2: Serializer forwards commands one at a time to all other servers
3: Once the serializer received all acknowledgments, it notifies the client about the success

2 Lock protocol

deadlock



Algorithm 15.10 Two-Phase Protocol

Phase 1

```
Client asks all servers for the lock
```

Phase 2

```
if client receives lock from every server then
    Client sends command reliably to each server, and gives the lock back
else
    Clients gives the received locks back
    Client waits, and then starts with Phase 1 again
end if
```

2 Lock protocol

Algorithm 15.10 Two-Phase Protocol

```
  Phase 1
Client asks all servers for the lock
  Phase 2
if client receives lock from every server then
  Client sends command reliably to each server, and gives the lock back
else
  Clients gives the received locks back
  Client waits, and then starts with Phase 1 again
end if
```

2 Lock protocol

Algorithm 15.10 Two-Phase Protocol

Phase 1

```
Client asks all servers for the lock
```

Phase 2

```
if client receives lock from every server then
   Client sends command reliably to each server, and gives the lock back
else
   Clients gives the received locks back
   Client waits, and then starts with Phase 1 again
end if
```

What happens if some servers/clients do not respond?

Serious troubles!

Tickets concept -- PAXOS

Weaker than locks

Reissuable: new tickets can be issued even if old ones not returned

Expiration: a ticket accepted only if is it the most recent one

Ticket protocol --- 1-st trial

Algorithm 15.12 Naïve Ticket Protocol

Phase 1

Client asks all servers for a ticket

Phase 2

if a majority of the servers replied **then**
    Client sends command together with ticket to each server
    Server stores command only if ticket is still valid, and replies to client
else
    Client waits, and then starts with Phase 1 again
end if

Phase 3

if client hears a positive answer from a majority of the servers **then**
    Client tells servers to execute the stored command
else
    Client waits, and then starts with Phase 1 again
end if

Ticket protocol – 1^{st} trial

Problem:

Client A may store the commands on majority, then postpone phase 3

Client B may store some commands

Client A says to execute the stored command

PAXOS

Algorithm 15.13 Paxos

Client (Proposer)		**Server (Acceptor)**	
Initialization			
c	$\triangleleft$ *command to execute*	$T_{\max} = 0$	$\triangleleft$ *largest issued ticket*
$t = 0$	$\triangleleft$ *ticket number to try*		
		$C = \perp$	$\triangleleft$ *stored command*
		$T_{\text{store}} = 0$	$\triangleleft$ *ticket used to store C*

PAXOS

Phase 1 ....

~~$t = t + 1$~~

Ask all servers for ticket t

if $t > T_{max}$ **then**

  ~~$T_{max} = t$~~

  Answer with $\mathsf{ok}(T_{store}, C)$

end if

Phase 2 . . . .

if a majority answers ok **then**
    Pick (T_{store}, C) with largest T_{store}
    if $T_{\text{store}} > 0$ **then**
        $c = C$
    end if
    Send propose(t, c) to same majority
end if

if $t = T_{\max}$ **then**
    $C = c$
    $T_{\text{store}} = t$
    Answer success
end if

Phase 3 ....

```
if a majority answers success then
    Send execute(c) to every server
end if
```

Lemma 15.14. *We call a message* `propose`(t,c) *sent by clients on Line 12 a* proposal for (t,c). *A proposal for (t,c) is* chosen, *if it is stored by a majority of servers (Line 15). For every issued* `propose`(t',c') *with* $t' > t$ *holds that* $c' = c$, *if there was a chosen* `propose`(t,c).

PAXOS properties

- only one propose(t,c) from a user for a given t
  - Indeed, before the next propose phase 1 must be executed with t:=t+1

- Assume there is propose(t',c') with t'≠t and c '≠c
  - let t' be the smallest one with this property
  - nonempty set S of servers that were involved in propose(t',c') and propose(t,c)
  - a server s from S stored (t,c), it must have occurred before accepting ticket t'
  - the client learns from s and is aware of c when issuing propose(t',c'), where c' is the most recent seen by the client
  - There is no more recent as c, as otherwise t' would not be minimal

PAXOS properties

COROLLARY If a command c is executed by some servers then eventually it will be executed by all servers.

Indeed: after the 1st propose(t,c) every proposal will be for c