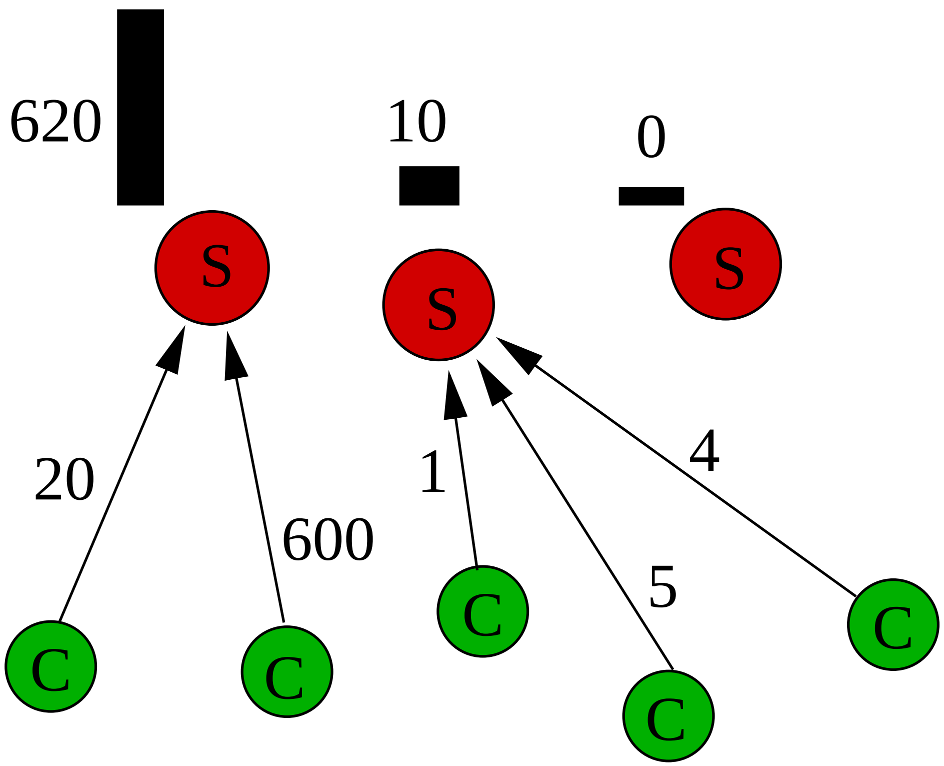
Self-stabilizing Load Distribution for Replicated Servers on a Per-Access Basis

Felix Gärtner *Henning Pagnia*

DARMSTADT UNIVERSITY OF TECHNOLOGY
GERMANY

The Problem

- Level load arising from accesses to replicated data

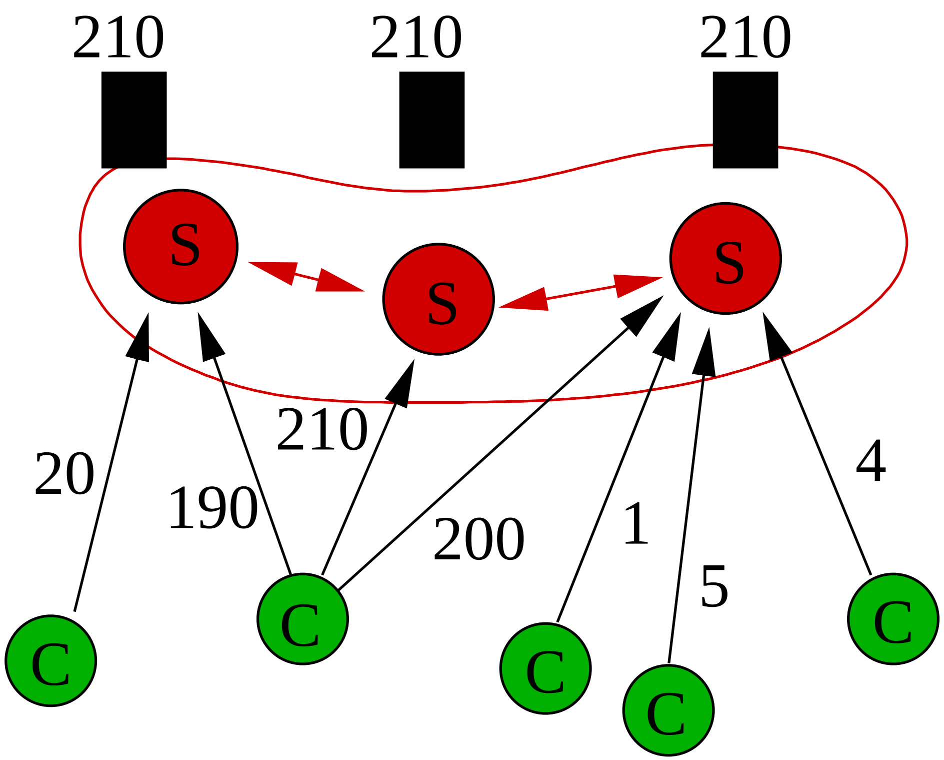


Remainder of This Talk

- The idea of a solution
- The modules of the solution
- The worktogether of the modules
- The self-stabilization property

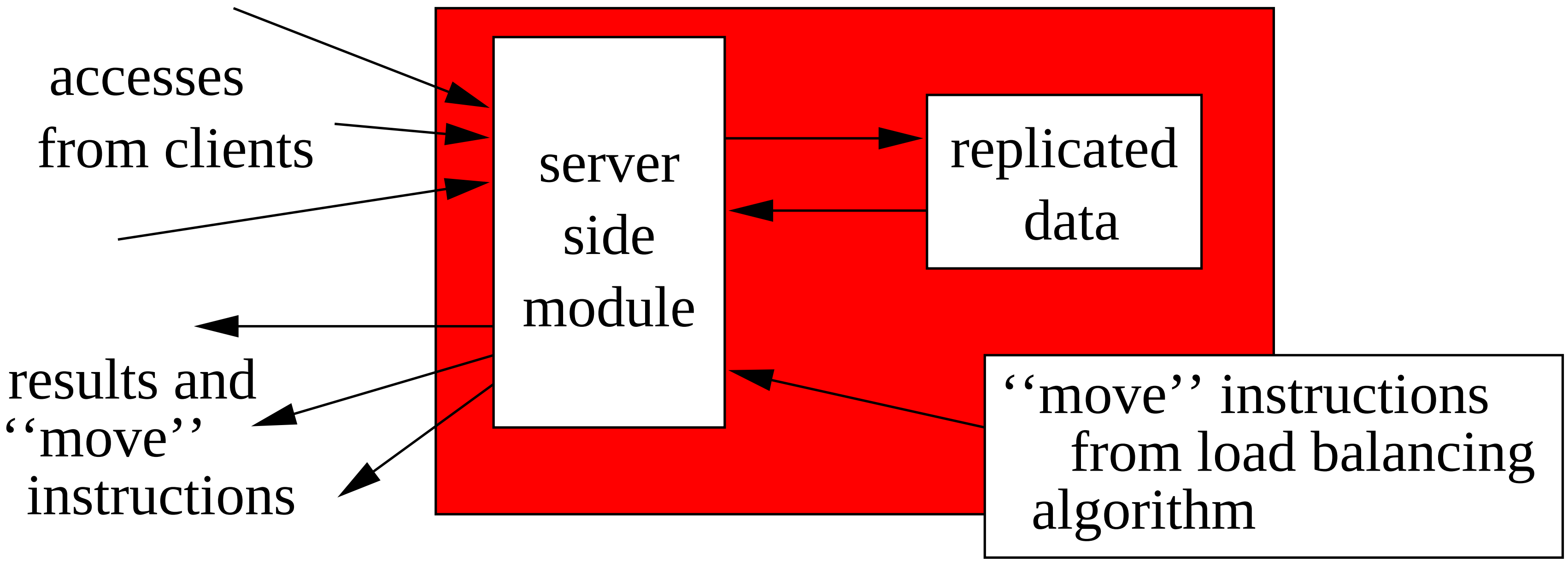
The Idea for a Solution

- Use standard load balancing algorithm on servers
- Use redirection mechanism



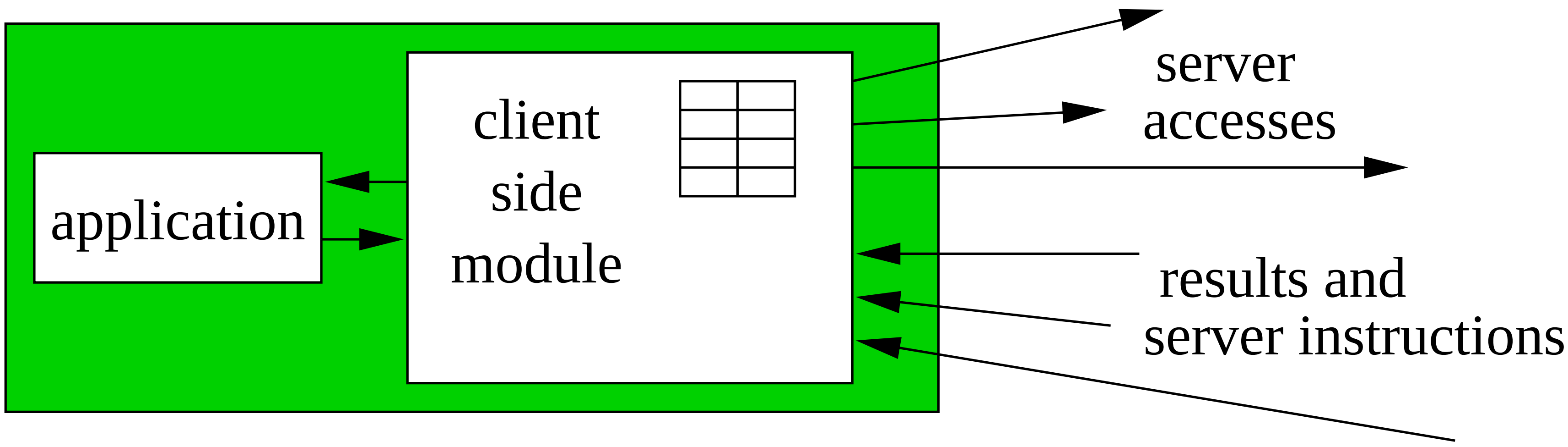
The Solution: Server Side

- Add distribution module
- Collect access statistics by relaying accesses
- Distribute "move" instructions if load needs to be balanced

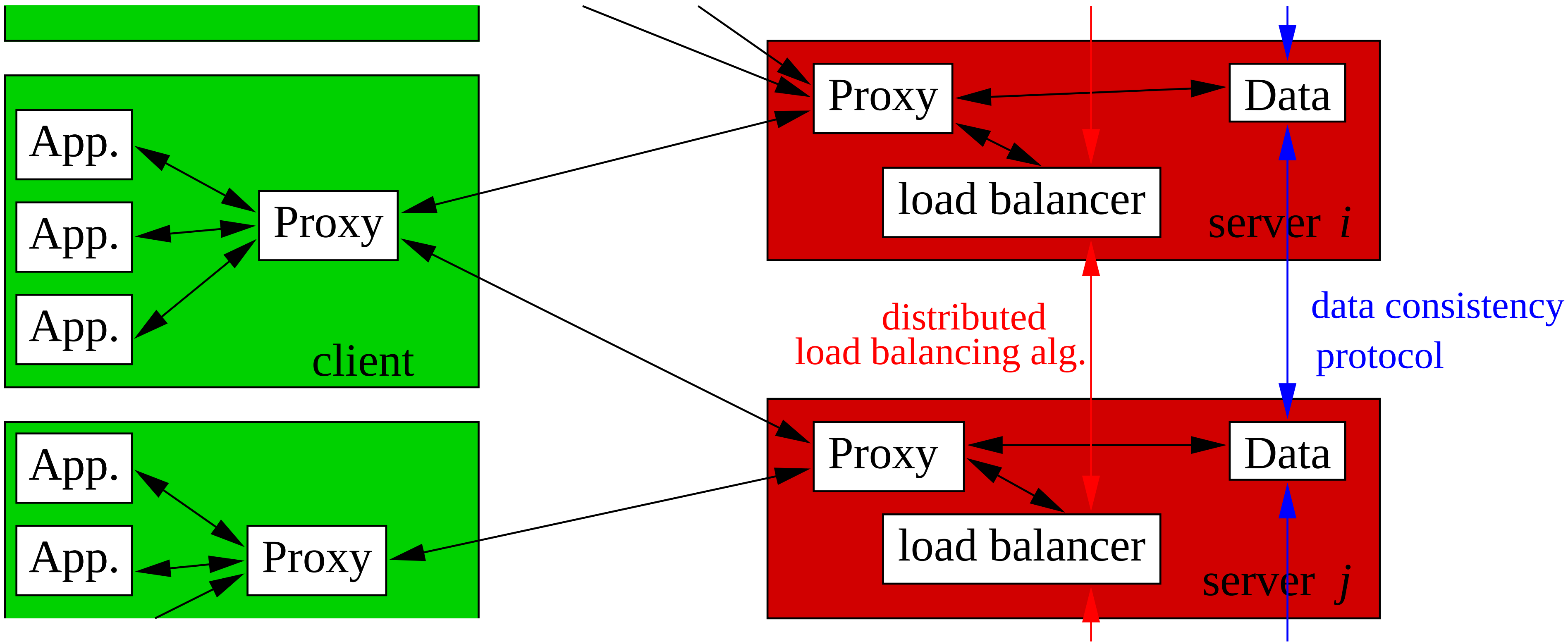


The Solution: Client Side

- Collect access statistics and maintain access pattern
- Relay requests according to access pattern
- Reconfigure access pattern according to “move” instructions



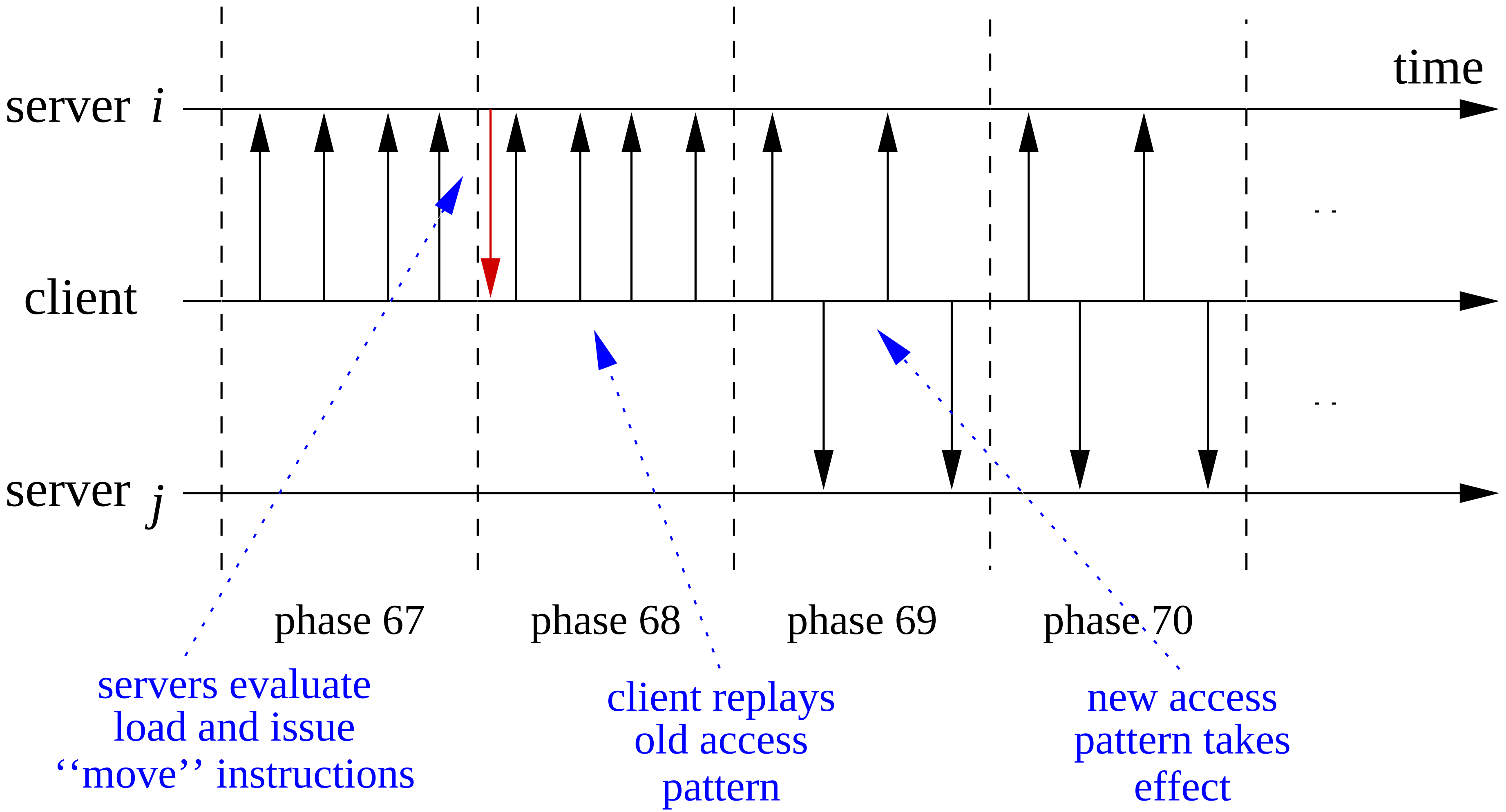
The Modules



App.
App.
App.
Proxy
client
App.
App.
Proxy
Proxy
Data
load balancer
server i
distributed
load balancing alg.
data consistency
protocol
Proxy
Data
load balancer
server j

The Worktogether

- Necessary: synchronized clocks



The Self-stabilization Property

- Composition of two self-stabilizing algorithms: (a) load balancing and (b) client/server redirection modules
- Proof obligations for (b): set of stable states P (nothing changes)
  - Closure: If P holds, and the access pattern is regular, then P will hold again after one phase.
  - Convergence: If the access pattern is regular, the system will reach a state in P after at most two phases.
- Assumptions: constant message delivery delay, perfectly synchronized clocks.
- Overall proof obligation: show interference freedom between (a) and (b)

Conclusions

- Problem statement
- Solution: introduce a redirection mechanism
- Make it self-stabilizing

Further questions:

- Practical effects of non-perfectly synchronized clocks on stabilization time?
- Effects of different consistency control protocols?