אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Self Stabilizing Distributed File System

Shlomi Dolev and Ronen I. Kat

Department of Computer Science, Ben-Gurion University

Research Sponsored by IBM

DFS Motivation

- Performance
  - Communication, placing files closer to users
  - Load, no single bottle-neck
- Fault tolerance
  - No single point of failure
  - Partitions, disconnected operations

Related Work

- File systems
  - NFS – network file system protocol
  - AFS – Andrew File system – CMU (1988)
  - Coda - CMU (1998)
  - Intermezzo – Peter J. Braam, CMU
- Peer to peer (2000)
  - Global storage: OceanStore – Berkeley
  - Server less: Microsoft Farsite.

Talk Overview

- Self-stabilization
- A Distributed File System
- Algorithms
- File system operations
- Future work

Self Stabilization

A self-stabilizing system is a system that can automatically recover following the occurrence of (transient) faults.

The idea is to design system that can be started in an arbitrary state and still converge to a desired behaviour.

Self Stabilization Motivation

- The combination and type of faults cannot be **totally** anticipated in on-going systems
- Any on-going system **must** be self stabilizing (or manually monitored)
- Self-stabilizing algorithm can recover from any arbitrary state reached due to the occurrence of faults

Current Research

- Self healing
- Adaptiveness
- Automatic recovery
- Autonomic computing

Self Stabilization

Dijkstra 1974

Examples

- Token passing (mutual exclusion)
- Spanning tree
- Finding cliques in distributed systems

- **From theory to practice !**

A Distributed File System

- Self-stabilizing spanning tree of the replication servers
- Self-stabilizing extension of the replication backbone tree to include caches
- File operations controlled by a self-stabilizing synchronizer

Algorithms – Self Stabilizing

- **Electing a leader (leader election)**
- Collecting connectivity information
- Optimising communication cost
- Synchronizer for file consistency

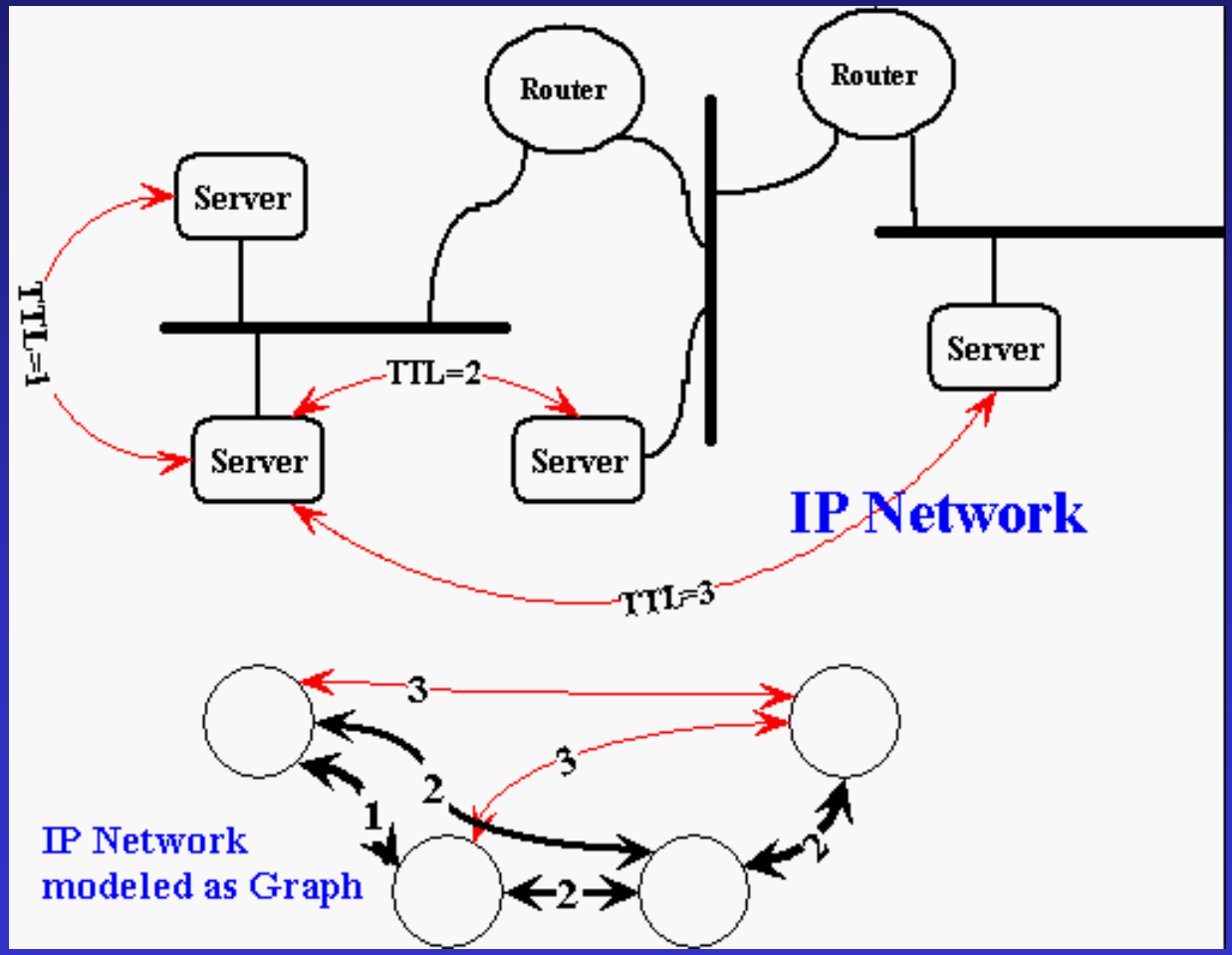
Leader Election

- A single leader coordinates the tree construction
- Leader broadcast heart beats
- If heart beat not received then become leader
- If more than one exists, one survives

Algorithms – Self Stabilizing

- Electing a leader (leader election)
- ➢ **Collecting connectivity information**
- Optimising communication costs
- Synchronizer for file consistency

TTL of Multicast Defines Graph



Update Algorithm

- Repeatedly collect routing tables from all neighbours (in the graph)
- If leader is not in routing table then a manager (local leader) notifies the leader to increase TTL (graph connectivity)
- Routing tables define a distributed BFS spanning tree
- Stabilizes!

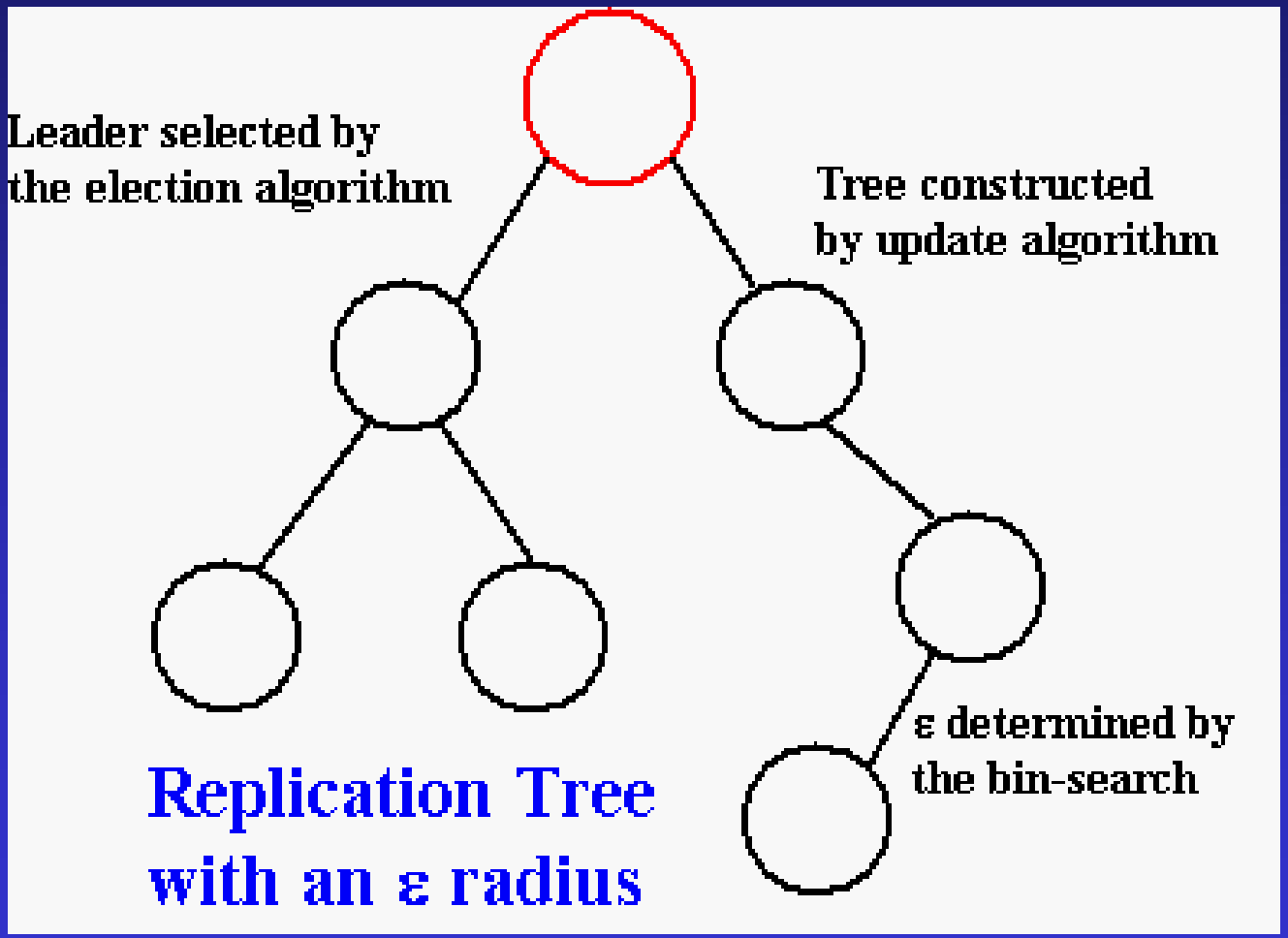
Algorithms – Self Stabilizing

- Electing a leader (leader election)
- Collecting connectivity information
- ➢ **Optimising communication costs**
- Synchronizer for file consistency

Optimising Communication Costs

- Goal: find the minimal ε radius that keeps connectivity
- Too small, increase ε by a factor of 2
- Run a 2nd instance of update with $\gamma < \varepsilon$
- Finding the smallest ε by binary search

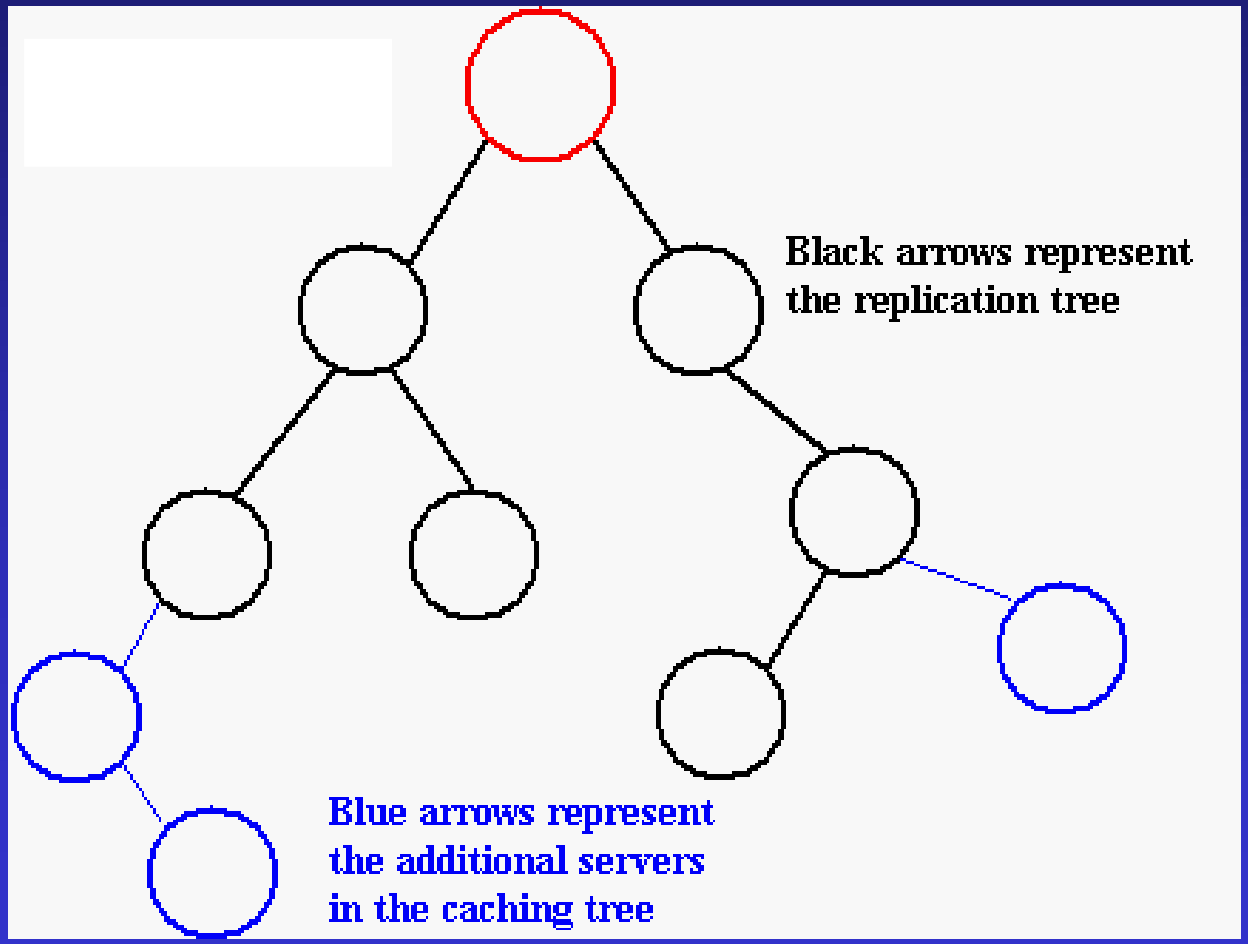
Tree Structure



Caching Tree

- Extends the replication tree
- The update algorithm constructs both
- Servers execute two instances
- Caches execute one instance

Combined Spanning Tree



Algorithms – Self Stabilizing

- Electing a leader (leader election)
- Collecting connectivity information
- Optimising communication costs
- **Synchronizer for file consistency**

Synchronization Mechanism

- Provide reliable command and timing
- Propagate commands between servers
- Collect and distribute information

Synchronizer

- Leader repeatedly chooses a colour in a round robin fashion (Dijkstra)
- The colour is **propagated** to the leaves
- **Convergcast** of the colour arrives to leader before the leader chooses the next colour
- **Distributes** and **collects** information!

Replication Consistency

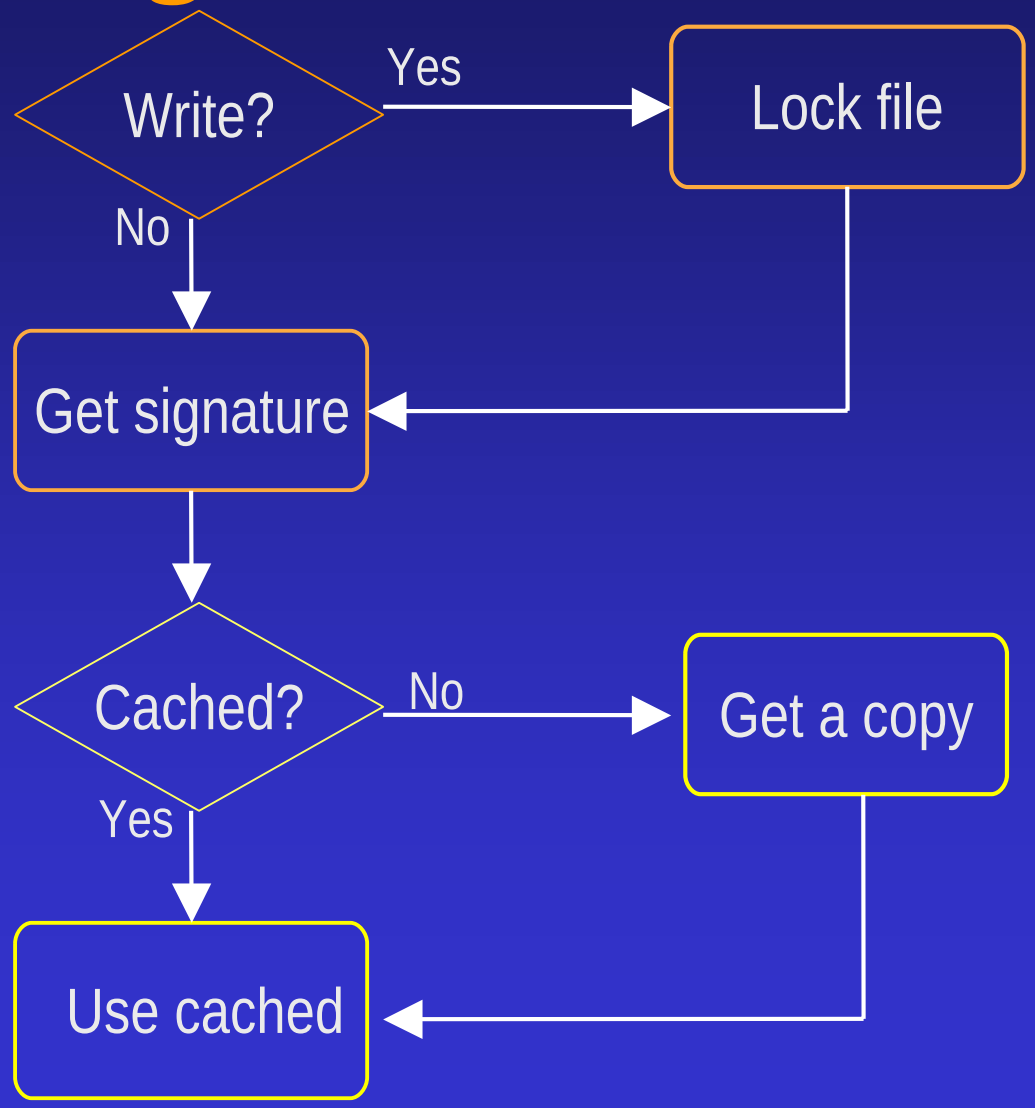
- Verifies file signatures (e.g., crc)
- Different signatures – a conflict
- Conflict resolution (e.g., majority)
- Broadcast resolved signature and sources list

Locking Table

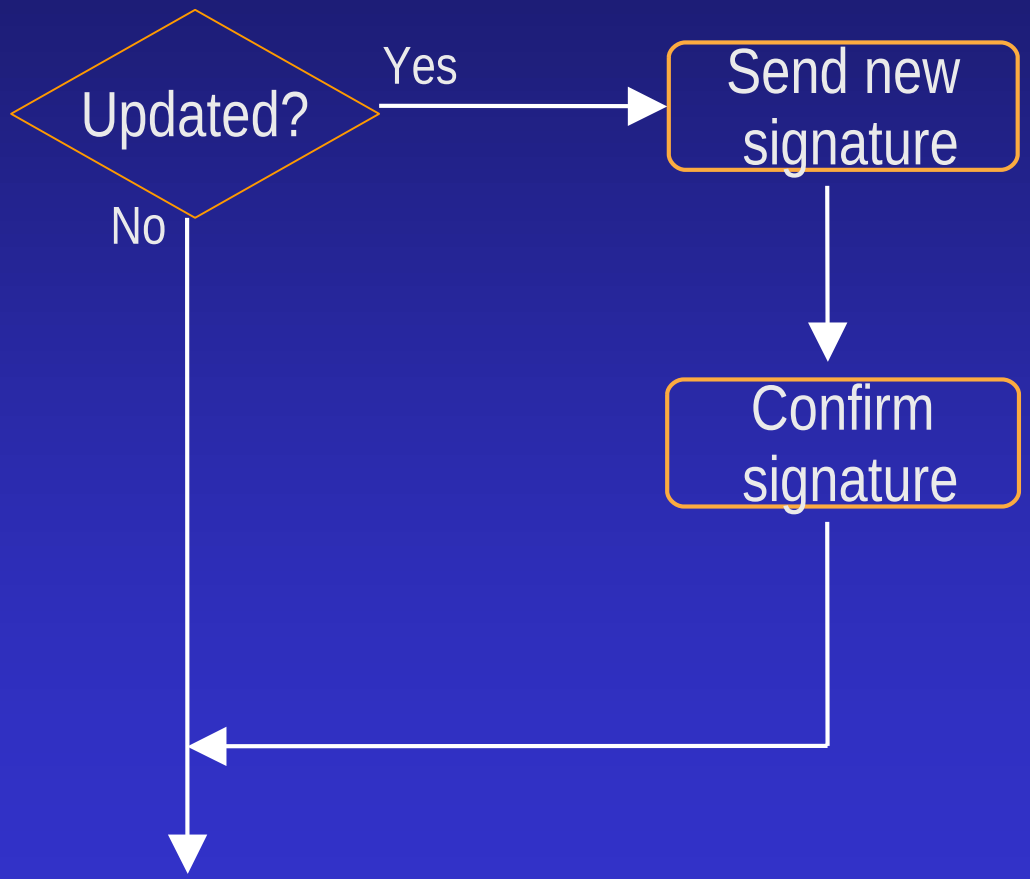
- A single global lock table
- Servers request a lock
- Leader resolves multiple requests
- Lock are removed by requesting server (or disappearance of server from tree)

File System Operations

Accessing a File



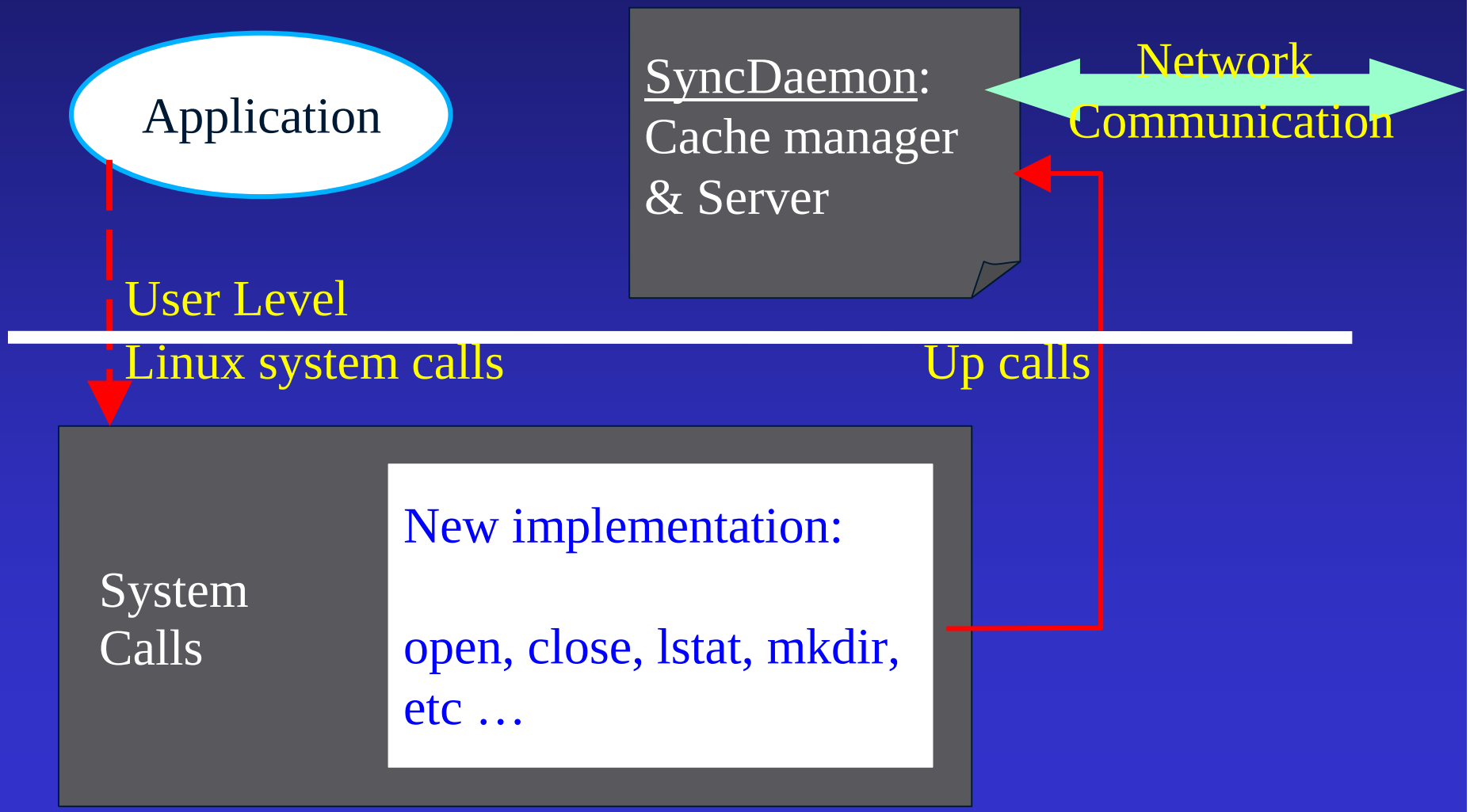
Closing a File



Meta Access (e.g., ls)

- Blocked until a lock is obtained
- Globally processed
- Verify signatures on results

SRFS Linux Interface



Future Work

- Kernel VFS module.
- Communication improvements:
  - Reducing update messages
  - Using timers with β-synchronizer
- Integrating disconnected operations
- Conflict resolution algorithms

Credits

Faculty

Prof Shlomi Dolev dolev@cs.bgu.ac.il

Graduate Students

Ronen I. Kat kat@cs.bgu.ac.il

System Engeenier

Albina Budker albinabu@cs.bgu.ac.il

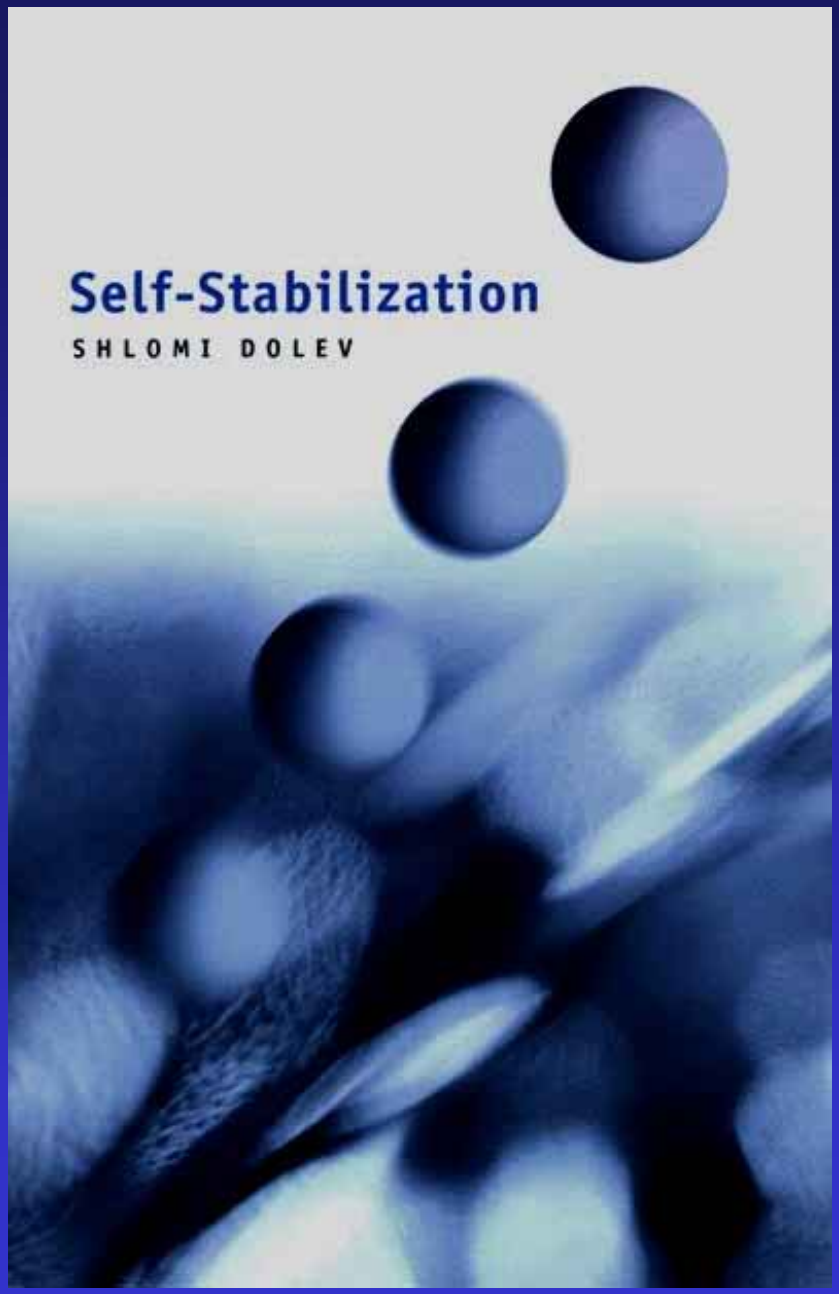
Undergraduate Students:

Amir Livneh livneha@cs.bgu.ac.il
Itay Granik granik@cs.bgu.ac.il
Boris Lansky lanskyb@cs.bgu.ac.il
Naama Shmuel shmueln@cs.bgu.ac.il
Moshe Shish shishm@cs.bgu.ac.il
Guy Erlich erlichg@cs.bgu.ac.il
Avital Chohen avitalco@cs.bgu.ac.il
Yael Biran birany@cs.bgu.ac.il
Tamir Fridman tamirf@cs.bgu.ac.il
Shiraz Bernard shirazb@cs.bgu.ac.il
Zvika Ferents ferents@cs.bgu.ac.il
Roy Feintuch feintuch@cs.bgu.ac.il
Chen Shalev shalevc@cs.bgu.ac.il
Shay Kraim kraim@cs.bgu.ac.il
Alex Hayuit

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Visit us at

www.cs.bgu.ac.il/~srfs



MIT Press, 2000