Network Applications

Martin Děcký

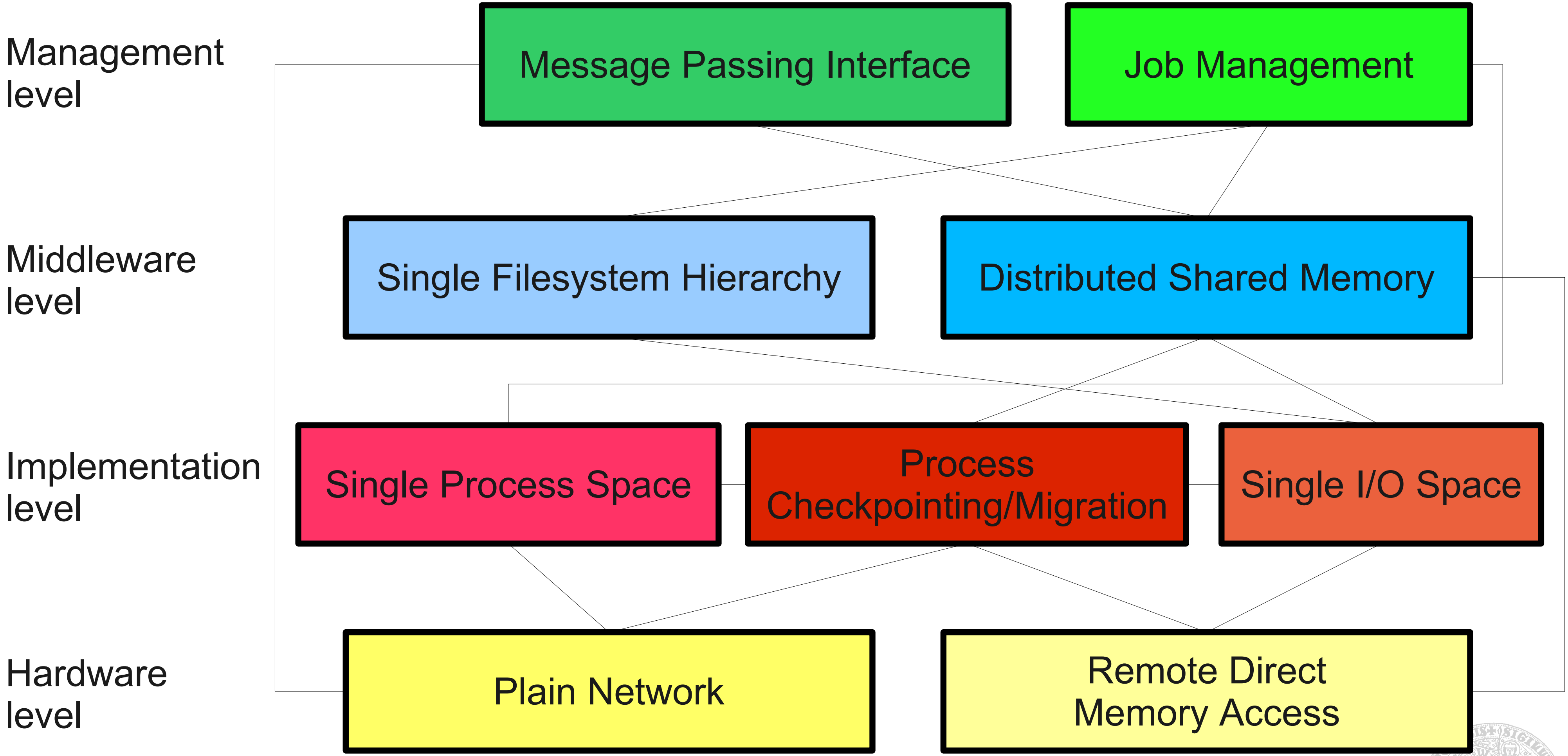
DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz/

CHARLES UNIVERSITY IN PRAGUE
FACULTY OF MATHEMATICS AND PHYSICS

Distributed Computing

- Goal
  - Multicomputer system transparently as a single system (similar to multiprocessor system)
- Motivation
  - Scalability
    - Clusters, grids
  - Better use of resources
    - CPUs and memory of idle machines
  - High availability
    - Fail-over

Distributed Computing (2)



Distributed Computing (3)

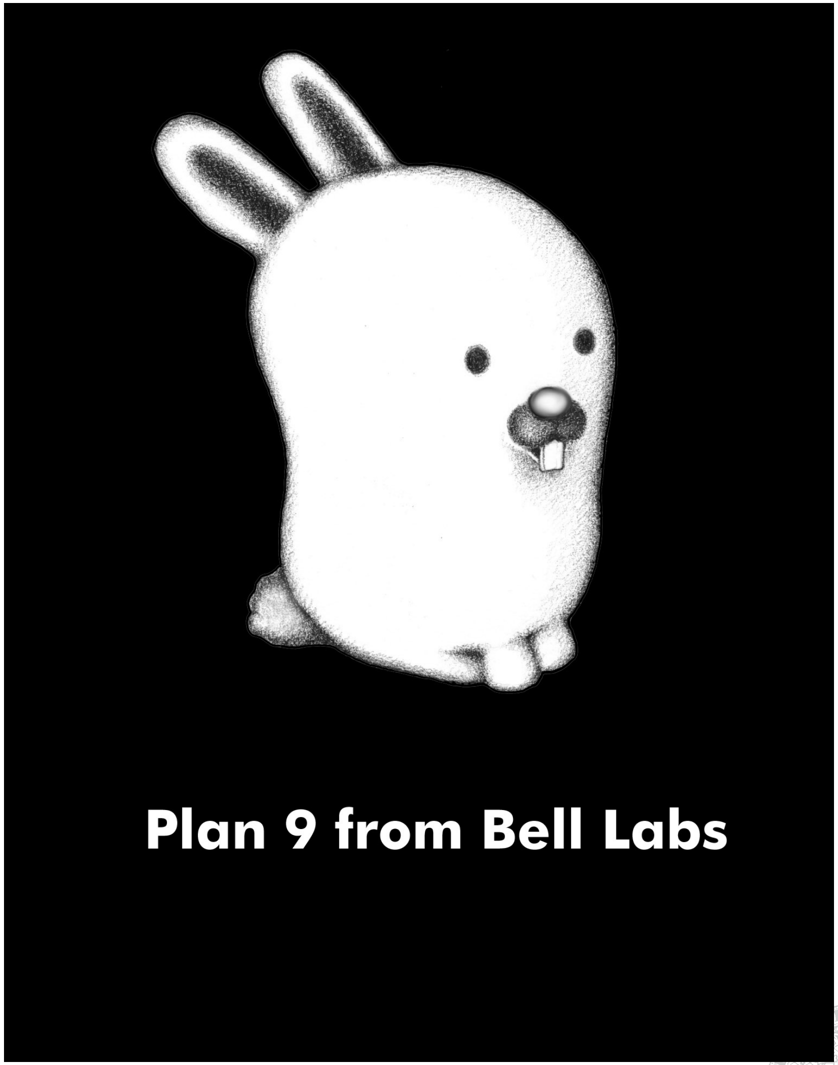
- Illusion degree vs. heterogenity
  - Middleware (OpenMP)
    - Pure client programs level
      - Manual deployment and management
    - Heterogenous environments
  - Global resource naming (Plan 9)
    - Transparent to client programs
      - Manual management
      - All resource operations reduced to a few ones (overhead)
    - No process migration

Distributed Computing (4)

- Multiple-system image (LinuxPMI)
  - Transparent process migration
    - Systems can be relatively heterogenous (except CPU type)
    - Automatic management and load ballancing
  - Almost no resource sharing and IPC
    - Except CPU, physical memory, pipes and trivial cases
- Single-system image (MOSIX, Amoeba)
  - Transparent process migration
    - Nodes are almost fully homogenous
  - Full resource sharing and IPC
    - Single filesystem hierarchy, global resource naming and access by design
      - Sometimes with hardware support (RDMA)

Plan 9 from Bell Labs

- Unix successor
  - *Unix 4th edition*
- Hybrid kernel design
- Single basic paradigm
  - *Everything is a file*
    - Filesystem name spaces
- Unified resources
  - Local and remote resources treated equal



9P

- 9P2000
  - Network communication protocol
  - Connection-based
    - TCP
    - IL (IP protocol 40)
      - Reliable datagram sending, in-sequence delivery, adaptive timeouts, low complexity
      - Suitable for local area networks
  - Client-server approach
    - Serving filesystem trees (resource naming)
    - Running a constant set of methods on files

9P messages

- Version
  - Define a *session*
    - Abort outstanding I/O
- Attach
  - Get a filesystem tree
- Auth
- Walk
- Open, New
- Clunk, Delete
- Stat, Wstat
- Read, Write
  - Identpotent
- Flush

Plan 9 Files

- Supplied by kernel drivers
  - `dev` driver
    - `cons`, `consctl`, `cmd`, `cputime`, `kmesg`, `null`, `zero`
  - `proc` driver
    - Similar to Linux `/proc`
      - Live processes and their properties (`note`)
      - `trace` (kernel trace)
  - `env` driver
  - `mnt` driver
    - Serves files using 9P protocol from servers

Plan 9 Files (2)

- Supplied by remote binding
  - `import hostname /proc /mnt/remote/proc`
- Supplied by user space servers
  - `net`
    - `/net/tcp/clone`, `/net/tcp/0/ctl`, `/net/tcp/0/data`, `/net/tcp/0/local`

Name spaces

- Each process can have a different view (name space) of the filesystem tree
  - Name space group inherited by `fork()`
    - `int bind(char *name, char *old, int flag)`
      - File `old` becomes alias for `name`
      - Original files are not hidden (union)
    - `int mount(int fd, int afd, char *old, int flag, char *aname)`
      - Replace a subtree with a tree `aname` served by `fd` (open connection to server)

Name spaces (2)

- Flags
  - `MREPL` (add files to the end of the union)
  - `MBEFORE` (add files to the beginning of the union)
  - `MCREATE` (newly created files are stored in the given directory)
  - `MCACHE` (cache files content locally)
- Usage
  - `$PATH` replacement
    - Every process (and user) can enhance `/bin` via bind

Plan 9 Filesystem – Fossil

- User space server
  - Snapshots (copy-on-write)
    - Archives (removable), snapshots (permanent)
    - Available to all users
  - Implements filesystem hierarchy
  - Relies on backend server for storing data and metadata blocks

9P Persistent Storage – Venti

- Storing blocks (512 B – 56 KB)
  - Write-once
    - Originally designed for optical jukeboxes
  - Addressing using SHA-1 hash of the data block
    - Verification of the correctness of the server
    - Hypothetical collisions not solved
  - Index storage (hash table with constant buckets)
  - Data log storage
  - Fossil builds hash trees above Venti

Inferno

- Fork of Plan 9
  - Derived from 2nd edition
  - Monolithic kernel
    - The whole system runs in privileged mode or inside another host environment (web browser)
  - No standard user-space
    - Virtual machine approach (Limbo language)
    - Platform independent byte code, JIT (Dis)
  - Styx
    - Variant of 9P (9P2000)

MOSIX

- Fork-and-forget Unix (Linux) cluster
  - Single-system image
  - Transparent load ballancing
    - Sharing of CPU (same type) and physical memory
    - Unmodified Unix/Linux API
      - Except management extensions
    - Process migration between nodes
      - Whole process images and state
      - Multiple migration criteria (to avoid trashing, ping-pong, etc.)
        - **Memory requirements**
        - Communication cost
        - CPU usage vs. local resources I/O frequency

MOSIX (2)

- Resource management
  - Global resources
    - Accessible and coherent on all nodes
      - Cluster filesystems (Direct File System Access)
      - Network filesystems mounted on all nodes
      - Special hacks (`/dev/null`, etc.)
  - Local resources
    - Accessible only on the home node
      - Local filesystem access
      - Device drivers
      - Pipes, shared memory
      - Syscalls changing local machine state
    - Migrated processes communicate with process deputies (proxies) or are migrated back to the home node

MOSIX (3)

- History
  - Since 1977: Prof. Amnon Barak (Hebrew University of Jerusalem)
    - MOS (based on *Unix 7th edition*) on PDP-11
  - Since 1981: Various Unix variants
    - Notably *Unix System V* on VAX
  - Since 1991: BSD/OS on x86
  - Since 1999: Linux on x86
  - Since 2001: closed source
    - *openMosix* fork (by Moshe Bar)

openMosix

- Based on last open MOSIX source code
  - Targeted at Linux 2.4 on x86
  - Various optimizations
    - Support DFSA on plain NFS (mounted on all nodes)
    - Smaller migration overhead
      - On-demand migrating of the individual pages of the process
  - Development ended in 2007
    - Because of low-cost multiprocessor computers

LinuxPMI

- Multiple-systems image
  - Based on openMosix for Linux 2.6
    - Originally never released beyond alpha stage
    - Many deviations from the original MOSIX concepts
  - MSI is like SSI, but from the perspective of each node
    - Targeted mostly on CPU-intesive tasks
    - Some I/O operations proxied transparently to the home node, communication using pipes is also transparent
      - Not supported: Writable memory mapped files, memory mapped devices, direct I/O operations, shared memory

Amoeba

- Distributed OS
  - Andrew Tanenbaum
  - Microkernel design
  - No process migration, but multicomputer transparency
  - First appearance of multikernel approach
    - Each core in a multiprocessor system runs its own copy of the microkernel
  - Designed with consern and server separation
    - Inter-process communication uses generated RPC

Amoeba (2)

- Basic concepts
  - Naming separation
    - File names managed by a dedicated *directory server*
      - Operations: create, delete, append (cap.), replace (cap.), lookup, getmasks, chmod
    - Maps file names to capabilities
  - Immutable files
    - Stored on dedicated *bullet servers*
    - Committed files
      - Operations: create, read (originally as a whole), delete, size
      - Simple replication
      - No coherency issues
    - Uncommited files
      - Operations: create, modify, insert, delete, read, size, touch

Amoeba (3)

- Capabilities
  - Users have a set of capabilities
  - Directory server maps files to capabilities
    - This allow permission checks
  - Problem: There is no global storage of all capabilities owned by the users
    - Capabilities in the directory server have a timeout
      - After a capability timeouts, it is removed
      - File names with no capabilities (or all capabilities expired) are automatically removed
    - Uncommited files
      - Timeout: 10 minutes
      - Timeout is restarted with each user operation
    - Commited files
      - Timeout: 24 hours
      - Timeout is restarted by the bullet server storing the data

Network Global Memory

- Extending the physical memory of a node
  - Various implementations
    - Network paging
      - Similar to the usual disk paging (swapping)
        - On memory pressure the page is sent over the network to a different node (where it is stored in physical memory)
          - In parallel, the page is also stored on the disk (as a backup)
        - Page-in handling similar to
      - Global cluster memory management
        - Local and global frames
        - Page-out
          - Local LRU from local to global frames
          - Global LRU for global frames (distributed coordinator)

Network Global Memory (2)

- Page-in
  - Location of the global frame
  - Global Cache Directory
    - Maps from frame ID to node
    - Each node has a piece of the directory
      - Broadcast request
  - Page Ownership Directory
    - Replicated on each node