

Towards SOS, 11/2003
<http://www.haifa.il.ibm.com/Workshops/SystemsAndStorage2003/abstract.html#10>

Shlomi Dolev¹ and Reuven Yagel^{1,2},
 ·Computer Science Department Ben-Gurion University of the Negev, Beer-Sheva, Israel
 ·Rafael, Haifa, Israel
{dolev|yagel}@cs.bgu.ac.il


06/2007

SOS - Motivation

- Coping with transient faults (e.g. soft-error (SEU) in a remote sensor results unpredictable system states)
- Growing interest in self-* / autonomic computing systems
- Self-stabilizing algorithms/programs assume hardware and operating system are also stabilizing
- Pentium HALTING problem: "... if the ESP or SP register is 1 when the PUSH instruction is executed, the processor shuts down..."

2

Space Vehicle Failure




- ...The Spirit rover has a radiation-hardened R6000 CPU from Lockheed-Martin Federal Systems... The operating system is Wind River Systems' Vx-Works..
- ...attempted to allocate more files than the RAM-based directory structure could accommodate. That caused an exception, which caused the task that had attempted the allocation to be suspended...
- ...Spirit fell silent, alone on the emptiness of Mars...
<http://www.eetimes.com/story/OE620040220S0046>

3

Proposed solution

- To build according to the well defined and understood paradigm of self-stabilization (traditionally used in distributed systems)
- Thereby achieving: trustworthiness, dependability, self-healing, automatic recovery, adaptive systems, ...



4

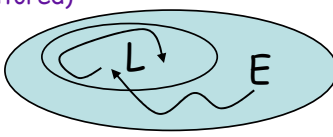
Self-stabilization

- Elegant fault tolerant approach
 - Alternative to the various duplication techniques
 - Started at any state, the system converges to a desired behavior
- Generally used in distributed systems
 - Routing, clock synchronization, leader election, etc.
- Overcome transient faults in the system.
 - "98% of RAM errors are soft errors", wrong CRC during communication etc.

5

Self-stabilization

- 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-Stabilization in Spite of Distributed Control" [Dijkstra '74]
- Self-Stabilization [Dolev '00]

6

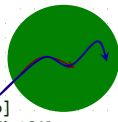
OS Dependability

- Past examples:
 - Dijkstra, "THE" Multiprogramming System '68 (Layered Approach)
 - Denning, Lampson, Saltzer ~'70 (Protection)
 - KeyKOS '85 (Capabilities)
 - EROS '92 (Checkpoints)
 - Micro-kernel ~'90, Exo-kernel '94 (Minimal TCB)
 - SPIN '95/EROS '99 (Capabilities, type safety)
- Recent:
 - IBM: K42 (Autonomic Computing), SUN: Solaris 10 (Predictive Self-Healing)
 - MSR: Singularity (managed code, sw protection, verification)
 - HW e.g. Intel : RAS: Machine Check, Elec. Isolation...

7

Goal: Autonomic Computer

- Following any sequence of transient faults, the (operating) system converges
- Using self stabilization:
 - A system can be started in an arbitrary state and converge to a desired behavior
 - Using fair composition to run hardware + OS layers
- BGU: Self-stabilizing stack [FOFDC07]
 - Microprocessor [DH04]
 - Operating System [DY04, DY05, DY06]
 - Compiler [DH05]
 - Framework: autonomic recovery [BK03, BD06]
 - Middleware: File System [DK02], Group Comm. [DS01]



9

SOS - Directions

- Black-box
 - Take existing (Desktop/Real-time) OS
 - Add stabilization layer
 - Detailed formal specification needed...
- Carefully tailoring a tiny kernel
 - Processor scheduling [SAACS04]
 - Memory management [SSS05, (SOSP05)]
 - Device drivers [SSS06]



10

Assumptions

- Every configuration (processor registers/ memory/IO controllers) is possible
- (Some) program code is hardwired (in ROM) and is correct [e.g. Castro & Liskov, 2000]
- Stabilization of other layers, CPU instruction manual (e.g. Pentium) defines a transition function
- Minimal protection, non-Byzantine programs
- Restrict access to privileged instructions and registers (e.g. HLT, LIDT)

11

Solution 1 (Black Box)

- Defining a legal execution is usually impractical
- Restore original state (variables + code), infinitely often
- Periodic Reset Re-install and Execute
 - Watchdog timer (self-stabilizing)
 - Periodic processor reset
 - During bootstraps OS reinstalled from ROM
- Weak self-stabilization
 - $E = (c_1, a_1, c_{i+1}, \dots, RRE, c_1, a_1, c_2, a_2, \dots, c_j, a_j, c_{j+1}, \dots, RRE, c_1, a_1, c_2, a_2, \dots)$
 - Is it always acceptable?

12

Solution 2 (Black Box)

- Periodic re-install code and continue execute
- A really non-maskable interrupt architecture
 - Limit the option to mask NMI
 - "During NMI handler, further NMI are discarded, until the IRET instruction is executed", "NMI is normally enabled"*
 - NMI vector hardwired
 - Time guaranty for executing system code
- Consistency check & establishment
- (Hierarchical) boots\reinstalls if needed

13

Solution 3 (Tailored)

- Specify main (additional) requirements for each OS function
- Evolve self-stabilizing solutions that follow computer-architecture/OS progress
- Detailed proof for self-stabilization of algorithms AND implementation
- Using processor instruction set directly
 - Don't rely on existing compilers

14

Processor Scheduling

- Requirements:
 - Fairness
 - Process stabilization preserving
 - Self-stabilizing
- Activation and full execution of scheduler no matter what the system state is
- Process counter increments modulo N
- Monitor next process' validity, e.g., PC in limits and points to some program instruction (of variable length)

15

Sketch of Proof

- In every infinite execution E, the code of the scheduler is started to be executed and is executed from the first instruction to the last instruction infinitely often
 - Based on the NMI architecture

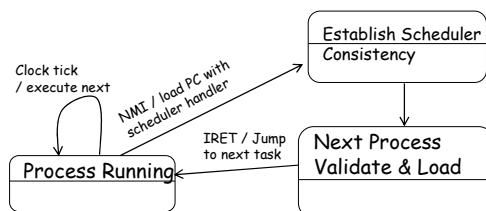
16

Sketch of Proof

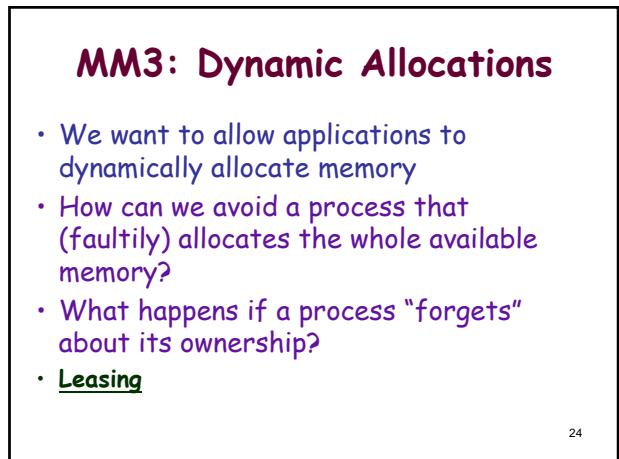
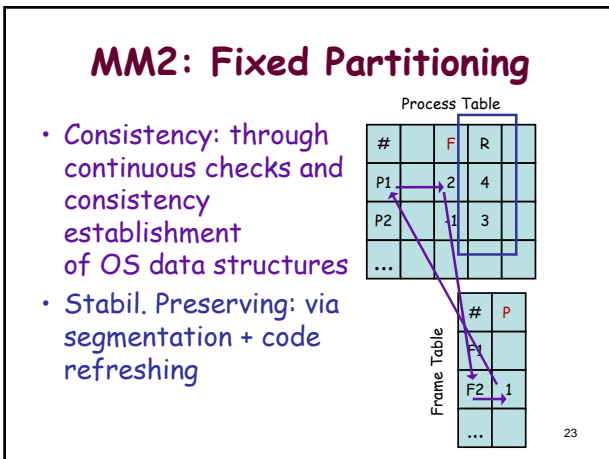
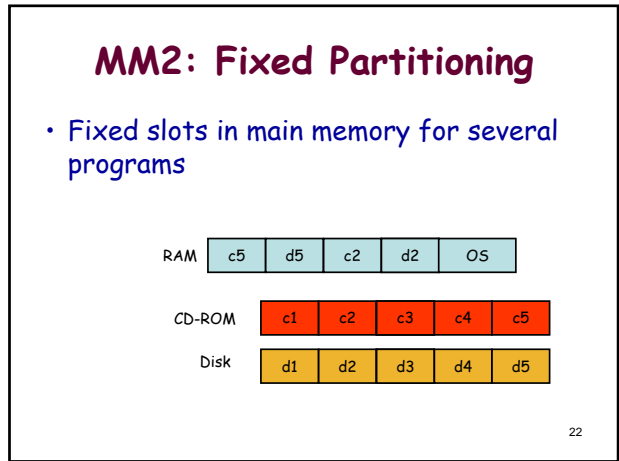
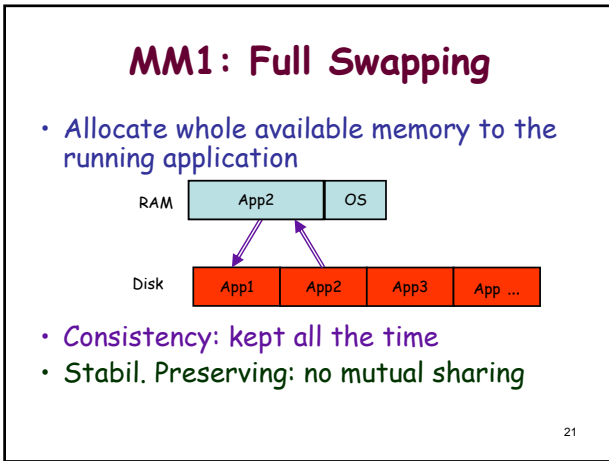
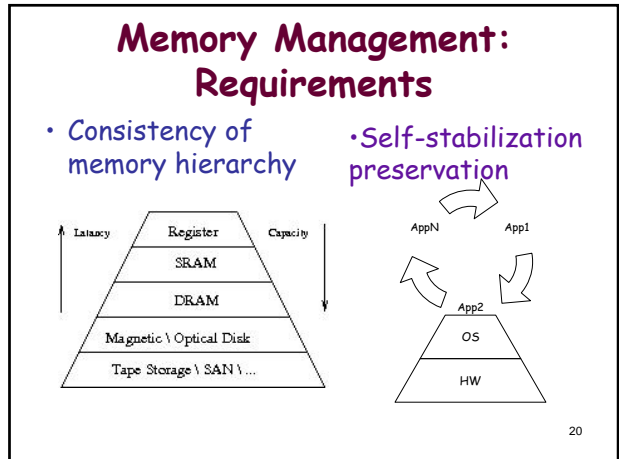
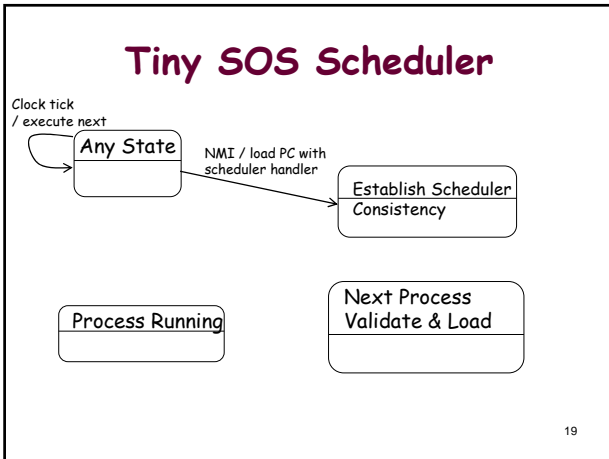
- In every infinite execution E of the scheduler each process is executed infinitely often
 - Correctness of the scheduler code
- The self-stabilizing scheduler preserves stabilization of processes
 - Consistency checks and program restrictions

17

Tiny SOS Scheduler

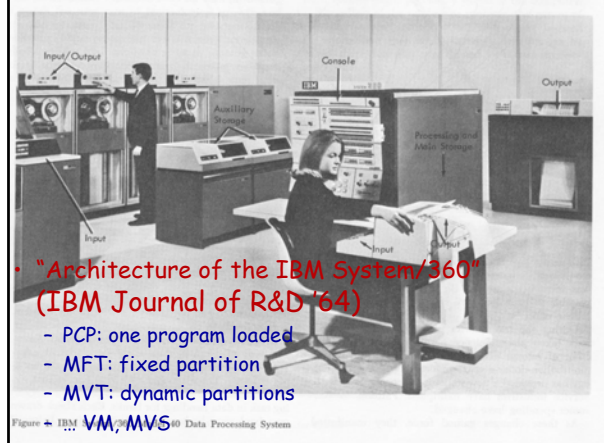
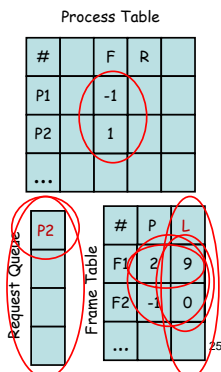


18



MM3: Dynamic Allocations

- Consistency: dynamic memory is temporarily leased & garbage collected, verification of PCB & queue
- Stabil. Preserving: access only through special segment selector

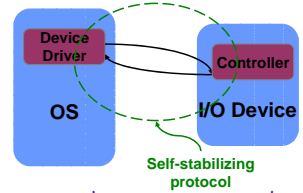


- "Architecture of the IBM System/360" (IBM Journal of R&D '64)
 - PCP: one program loaded
 - MFT: fixed partition
 - MVT: dynamic partitions

Device Drivers

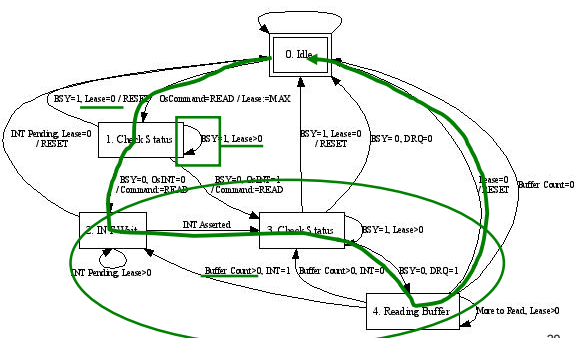
- "a modern Seagate drive contains roughly 400,000 lines of code" [IRON FS]
- "modern disk controllers often have many megabytes of memory inside the controller" [Minix]
- "In Windows XP, for example, device drivers cause 85% of reported failures" [Nooks]
- "Since drivers account for the majority of the code (over 70% in this release)... the error rate for drivers is almost seven times higher than the error rate for the rest of the kernel" [Engler et. al.]

DD: Requirements



- Ping-pong: exchange requests and replies infinitely often
- Progress: eventually every I/O request is executed completely and correctly according to, e.g., the ATA spec.

DD 1: Leasing



DD1: Leasing

- The driver reaches its idle state infinitely often
- From any configuration a safe (idle) configuration is eventually reached - device reset
- From there ping-pong holds
- Eventually progress holds

DD2: Consistency Checks

- Device driver contains consistency checker
- Consistency of driver current state with a snapshot of the controller

OS Driver PCS	Device Controller Snapshots
0,1	PCS=0 PCS=3
2	PCS=1, INT=1 PCS=2, INT=1
3	PCS=1 PCS=2, INT=1 PCS=3, INT=0
4	PCS=3

31

DD2: Consistency Checks

- Eventually requirements hold
- Combining the 2 solutions:
 - varying snapshot sizes according to degree of device stabilization
 - fallback to leases and restarts

32

Implementation

- Pentium in real-mode, single address space
 - Simple
 - common for sensors/microcontrollers
 - Protected mode & VM mechanisms can be handled accordingly
- Kernel size: ~1-4K
 - TinyOS ~1K, VxWorks ~10²K, Linux (default xconfig) ~4M
- Fault injection with the Bochs simulator
- Efficiency & compatibility?

33

Implementation

```

1 ) MM_FindFrame:  ;(PT, FT, i)
; al contains current frame suggestion
; nf <- (frames[PT[i]] + 1) modulo M
2 )   and     byte [bx+FRAME_COL], FRAME_MASK
3 )   inc     al
4 )   and     al, FRAME_MASK

;Check all slots for an empty one.
;while nf != frame[PT[i]] and FT[nf] != nil
5 ) while:
6 )   cmp     al, [bx+FRAME_COL]
7 )   jz     endwhile
8 )   lea    si, [frames]
9 )   add    si, ax
10 )  mov    di, [si]
11 )  cmp    di, NULL_PROCESS
12 )  jz     endwhile
; do nf <- (nf + 1) modulo M
13 )  inc    al
14 )  and    al, FRAME_MASK
15 )  jmp    while
16 ) endwhile:

; return found frame number in register 'al'
17 )  ret
    
```

34

Some Impl. Experience

- Minimal protection mechanisms
- Hardwired code
- Exception avoiding/handling, e.g. stack overflows, changes to interrupt mechanisms (NMI, fixed)
- Hidden processor states, reduced instructions
- Variable instruction length
- "You must pay extreme attention to detail here. One wrong bit will make things fail..."
<http://my.execpc.com/~geezer/os/pm.htm>

35

```

Bochs BIOS - Version 2.40
Copyright (C) 1990-2000 Elpin Systems, Inc.
All rights reserved.

Licensed for use with bochs, courtesy of MandrakeSoft.

For information on this or other UGA development products, contact
Elpin Systems at: (800) 723-9838 or www.elpin.com

Bochs BIOS, 1 cpu, $Revision: 1.103.2.2 $ $Date: 2004/02/02 22:39:22 $

Booting from Floppy...
Welcome to SOS
Loading 2nd stage to address 1000h
Loading interrupt handlers at address 80000h (UROM!)
Installing interrupt table at addresses 0 (UROM!)
    
```

Space Vehicle Failure



...the rover was in fact listening and rebooting, the team commanded Spirit to reboot without mounting the flash file system

...But just in case, the team is working on an exception-handler routine that will more gracefully recover from an allocation failure...

37

Conclusion

- The work shows theoretical and practical ways to achieve the goal of a self-stabilizing OS
- The (system) research community & industry can benefit from the foundation of self-stabilization

- <http://www.cs.bgu.ac.il/~yagel/sos>

Thank you

38