Toward Self-Stabilizing Operating Systems

Shlomi Dolev and Reuven Yagel

Computer Science Department,
Ben-Gurion University of the Negev and Rafael

24-Nov-2003
Talk Outline

- Self-stabilization and motivation for self-stabilizing operating systems
- Current operating systems do not stabilize!
- Black box solutions
- A self-stabilizing tiny OS
Self-Stabilization

- The combination and type of faults cannot be totally anticipated in an on-going systems
- Any on-going system must be self stabilizing (or manually monitored)
- A self-stabilizing system is a system that can automatically recover following the occurrence of (transient) faults

Windows

A fatal exception 0E has occurred at 0028:C0011E36 in VXD VMM(01) + 00010E36. The current application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue _
Unix-Linux-Windows-DOS do not Stabilize

- Fernando J. Corbató, "It almost goes without saying that ambitious systems never quite work as expected"

- "You must pay extreme attention to detail here. One wrong bit will make things fail. Protected mode errors often triple-fault the CPU, making it reset itself. Be prepared to see this happen again and again."
  [http://my.execpc.com/~geezer/os/pm.htm](http://my.execpc.com/~geezer/os/pm.htm)
Unix-Linux-Windows-DOS do not Stabilize

BSOD
Hardware Support?

From Pentium’s manual:

- “In the real-address mode, if the ESP or SP register is 1 when the PUSH instruction is executed, the processor shuts down due to a lack of stack space. No exception is generated to indicate this condition”
- “During NMI handler, further NMI are discarded, until the IRET instruction is executed”, “NMI is normally enabled”
- Memory management registers: GDTR, LDTR and IDTR
Goal: Autonomic Computer

- Following any sequence of transient faults, soft-errors ("98% of RAM errors are soft errors"), wrong crc during communication etc. the system converges

- Self Stabilization
  - System can be started in an arbitrary state and converge to a desired behaviour
  - Traditionally used in distributed systems (where transient faults are frequent)
  - Cannot run self-stabilizing algorithms unless hardware+OS are stabilizing (Fair composition [Dolev2K,DH03])

- Hence achieved: self-healing, automatic recovery, adaptive systems, …
Related Projects

• Autonomic Computing, IBM
• Recovery Oriented Computing, Berkeley-Stanford

• Self-stabilizing Micro-processor, BGU
• ...
• Self-stabilizing Autonomic Recoverer, BGU
What is an Operating System?

From “Operating System Concepts”, Silbershatz et. al.:

- Resource allocator – manages and allocates resources
- Control program – controls the execution of user programs and operations of I/O devices
- Kernel – the one program running at all times (all else being application programs)
Stabilization Scenario

• Some device driver is not functioning

• Program asking for service again? Some manual reset? Machine restart?

• Define system states \( s \) (variable assignments) which form legal states (e.g., of the driver). The OS will eventually reach such states
Operating System

- Main OS (black-box):
  - UNIX family, Windows, RTOS

- Three main parts of a kernel (tailored):
  - Processor scheduling
  - Memory management
  - Device allocation
Hardware Platform

• IA-32 (Pentium) Architecture
  – CPU modes (real, protected, …)
  – Interrupts: INTR + NMI pins, soft int, faults

• Additional watchdog device
  – Machine restart
  – OS Re-install & Repair

• Harvard model (data - program separated)
Black Box 1: Periodic Reset Re-install and Execute

- Watchdog timer (Self-stabilizing)
- Periodic processor reset
- OS code copy from ROM to RAM during bootstraps

- Is it always acceptable?
Black Box 2: Periodic Re-install and Continue Execute

- OS code (partially) re-install only – “hot boot”
- Watchdog
- Really non-maskable interrupt
  - Limit the option to mask NMI
  - NMI vector hardwired
- What about data consistency?
Black Box 3: Monitor and Establish Consistency

- Consistency check & establishment
- (Hierarchical) boots if needed
Tiny OS Scheduler

- Activation and full execution no matter what the system state is
- Task counter increments mod N
- Monitoring of next task’s validity, e.g., PC in limits
VGA 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 VGA development products, contact Elpin Systems at: (800) 723-9038 or www.elpin.com
VGA 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 VGA development products, contact Elpin Systems at: (800) 723-9038 or www.elpin.com

Bochs BIOS, 1 cpu, $Revision: 1.85.2.1 $ $Date: 2003/01/16 21:58:42 $

Booting from Floppy...

Loading Boot Image
..............
..............
....
Starting to load Self-Stabilizing OS.

Welcome to SIMPLIX.
Booting from Floppy...

Loading Boot Image

Welcome to SIMPLIX.
Loading OS interrupt handlers at address 20000 (ROM!)
Loading task code at address 10000
Installing interrupt handlers at addresses 0x20(=0x80), 0x70(=0x1C0) (ROM!)
Jumping to first task code at address 10000

0) Everything should be made as simple as possible, but no simpler. (Einstein)
Booting from Floppy...

Loading Boot Image

.............

.............

....

Welcome to SIMPLIX.
Loading OS interrupt handlers at address 20000 (ROM!)
Loading task code at address 10000
Installing interrupt handlers at addresses 0x20 (=0x84), 0x70 (=0x1C4) (ROM!)
Jumping to first task code at address 10000

Scheduler switches to next task, counter= 19

0) Everything should be made as simple as possible, but no simpler. (Einstein)
1) Keep It Simple, Stupid. (KISS engineers' principle)
2) The unavoidable price of reliability is simplicity. (Hoare)
3) What can be done with fewer ... is done in vain with more. (Ockham)
Tiny OS Scheduler

Any State
Tiny OS Scheduler

Clock tick / execute next

Any State
Tiny OS Scheduler

Clock tick / execute next

Any State

NMI / load PC with scheduler handler
Tiny OS Scheduler

Clock tick / execute next

Any State

NMI / load PC with scheduler handler

Establish Scheduler Consistency
Tiny OS Scheduler

Clock tick / execute next

Any State

NMI / load PC with scheduler handler

Establish Scheduler Consistency

Next Task Validated & Ready
Tiny OS Scheduler

- Any State
  - Clock tick / execute next
  - NMI / load PC with scheduler handler

- Establish Scheduler Consistency

- Next Task Validated & Ready
  - IRET / Jump to next task

- Doing Task
Tiny OS Scheduler

Clock tick / execute next

Any State

NMI / load PC with scheduler handler

Establish Scheduler Consistency

Next Task Validated & Ready

Clock tick / execute next

Doing Task

IRET / Jump to next task
Tiny OS Scheduler

- Any State
  - Clock tick / execute next
  - NMI / load PC with scheduler handler

- Establish Scheduler Consistency
  - NMI / load PC with scheduler handler

- Next Task Validated & Ready
  - IRET / Jump to next task
  - Clock tick / execute next
  - Doing Task
    - NMI / load PC with scheduler handler
Tiny OS Scheduler

Any State

Clock tick / execute next

Establish Scheduler Consistency

Next Task Validated & Ready

Doing Task

Some Error

Some Error

Some Error
Tiny OS Scheduler

- Any State
  - Clock tick / execute next
  - NMI / load PC with scheduler handler

- Establish Scheduler Consistency

- Next Task Validated & Ready
  - Doing Task
Conclusion

• Self-Stabilizing operating-system is a must and examples are well known

• We make the first steps forward this goal

Thank you