Experiences with Component Interference on Shared Hardware Resources

Philip Koopman

IFIP WG 10.4 Meeting, March 2004

Overview:

- Memory as a shared resource
  - Ballista testing results on memory integrity
  - New results include real-time Java, VxWorks
- Network as a shared resource
  - Software defect masquerading as an interference source
  - Protection vs. cost tradeoff points for authentication
- Conclusions

Robustness Testing Results

Ballista: Process Isolation

- Ballista robustness testing
  - Run combinational tests on valid and exceptional API parameters
  - Result is considered robust if tasks report recoverable exception
  - Result considered non-robust if crash, hang, or unrecoverable exception (e.g., a Unix signal), or if invalid parameter is accepted
- Experience testing several APIs
  - Unix operating systems
  - Embedded operating systems
  - Windows operating systems
  - HLA RTI (distributed simulation system)
  - CORBA client API
  - Java.lang API
  - Java components; SFIO library; other small case studies

Ballista: Scalable Test Generation

API `write(int filedes, const void *buffer, size_t nbytes)`

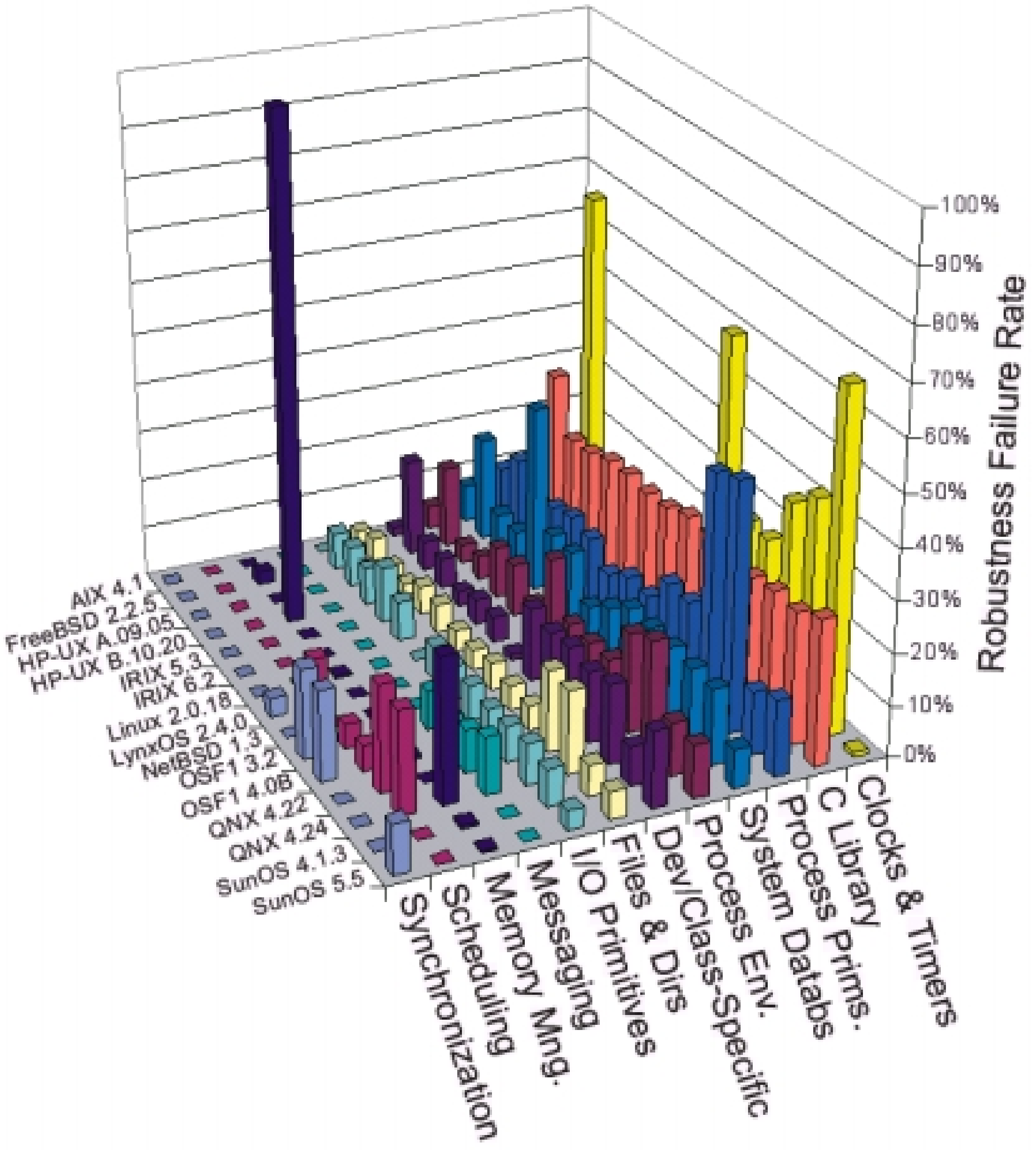
TESTING OBJECTS	FILE DESCRIPTOR TEST OBJECT	MEMORY BUFFER TEST OBJECT	SIZE TEST OBJECT
TEST VALUES	FD_CLOSED	BUF_SMALL_1	SIZE_1
	FD_OPEN_READ	BUF_MED_PAGESIZE	SIZE_16
	FD_OPEN_WRITE	BUF_LARGE_512MB	SIZE_PAGE
	FD_DELETED	BUF_XLARGE_1GB	SIZE_PAGEx16
	FD_NOEXIST	BUF_HUGE_2GB	SIZE_PAGEx16plus1
	FD_EMPTY_FILE	BUF_MAXULONG_SIZE	SIZE_MAXINT
	FD_PAST_END	BUF_64K	SIZE_MININT
	FD_BEFORE_BEG	BUF_END_MED	SIZE_ZERO
	FD_PIPE_IN	BUF_FAR_PAST	SIZE_NEG
	FD_PIPE_OUT	BUF_ODD_ADDR	
	FD_PIPE_IN_BLOCK	BUF_FREED	
	FD_PIPE_OUT_BLOCK	BUF_CODE	
	FD_TERM	BUF_16	
	FD_SHM_READ	BUF_NULL	
	FD_SHM_RW	BUF_NEG_ONE	
	FD_MAXINT		
	FD_NEG_ONE		

TEST CASE `write(FD_OPEN_RD, BUFF_NULL, SIZE_16)`

- Ballista combines test values to generate test cases

Failure Rates By POSIX Fn/Call Category

- Anecdotally, system killers lurk where there are high robustness failure rates
- New HP-UX 10 system killer was in memory management



Inter-Task Isolation Results

- Tests run as a long series
  - Each test is spawned as a separate task with clean state, BUT
  - How do we know that one test doesn't affect the next?
- Approach: permute order of tests
  - Run entire test suite in different order
  - If all tests results are identical, then probably no carry-over
  - Also, re-run tests and check for consistency
- Results on workstation Unix variants:
  - No carryover with order permuted on Digital Unix
  - Identical results when re-running tests on Digital Unix
  - Other workstation Unix variants performed similarly

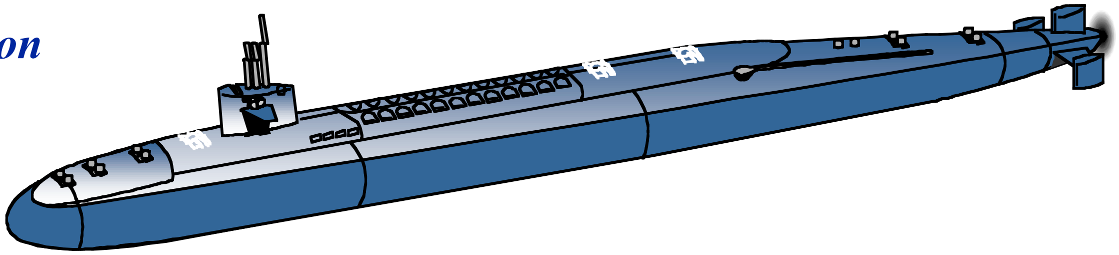
Isolation of Kernel from Tasks

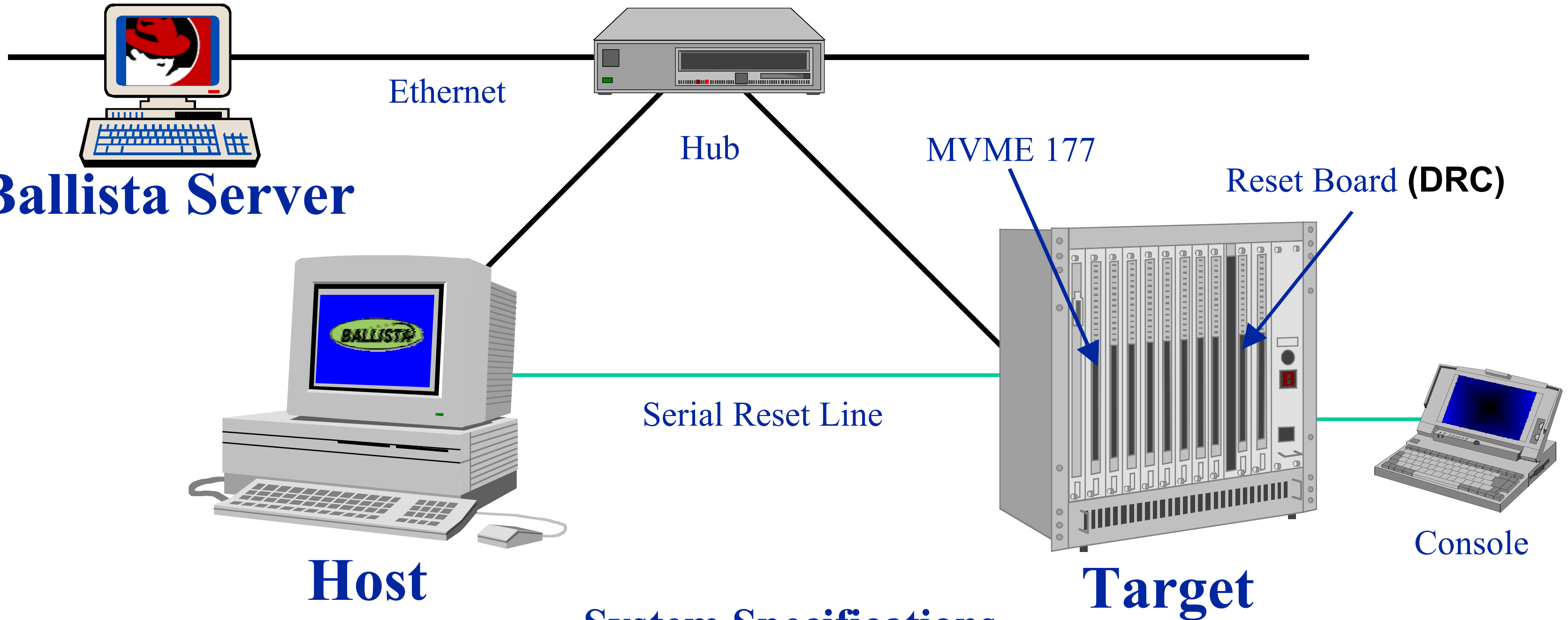
- Kernel can’t be 100% isolated from tasks
  - OS API calls give opportunity to attack kernel
  - Measure kernel corruption via repeatability & system crashes

- System crashes observed:
  - 10 Unix variants had no system crashes
  - 5 Unix variants had one or two functions that could crash system:
    - HPUX 10.20; Irix 6.2; LynxOS 2.4.0; Digital Unix 3.2; QNX 4.22
  - Windows NT & Windows 2000 – no observed crashes
  - Windows 95; 98; 98 SE – 7 or 8 functions could crash system
  - WinCE 2.11 – 28 functions could crash system
  - VxWorks 5.3.1 – 2 functions with *repeatable* system crashes
    - This is a surprising result … more shortly

Submarine System Robustness Tests

Dynamics Research Corporation





System Specifications

•Solaris 2.6	•VxWorks 5.3.1
•Gnu C Compiler	•Tornado 1.01
•Red Hat Linux 5.2	•Apache Web Server

Submarine Robustness Test Results

Operating System	Tester	Number of Modules Tested	Number of Tests Run	Actual Results			Normalized Results		
				STOP 5	STOP 4		STOP 5	STOP 4	
				Catastrophic	Abort	Restart	Catastrophic	Abort	Restart
VxWorks 5.3.1	DRC	38	13071	439	1913	1248	3%	15%	10%
	CMU	37	9944	360	1078	1428	2%	13%	9%
	Total	**75**	**23015**	**799**	**2991**	**2676**	**3%**	**13%**	**11%**
Solaris 2.5	CMU	233	92658	**0**	15374	28	0%	17%	0%
All Other*	CMU	4097	3186701	52**	674595	7387	0% - 1%	9% - 26%	0% - 3%

* 24 Other Operating Systems Tested
** Module Catastrophic Failures vice Test Failures

- TYPES OF STOPS
  - STOP 5 - Catastrophic; Tests Crashed the System, Requiring Hard Reboot
  - STOP 4 - Abort; Suffered Abnormal Termination
  - STOP 4 - Restart; Tests Hung in the OS Call, Requiring Task Restart
- VxWORKS HAS POTENTIAL FOR STOP 5’s
  - Not so for Solaris

OS With No Memory Protection

- VxWorks version did not have memory protection
  - Any task can overwrite OS memory
  - Expected lots of big crashes – but that's not what we saw
- Lots of carryover seen in testing
  - Changing order of test runs showed dramatic differences
  - Needed to do hardware reboot after every test over many weeks
  - Many difficult to reproduce crashes; difficult to analyze
- BUT, relatively few hard crashes
  - System would keep running long after OS state was corrupted
  - Crashes often required long series of tests to manifest
  - In general, system corruption not as dramatic as expected
    - (Still it was bad, but outward symptoms were sometimes subtle)

Java & Real-Time Linux Testing

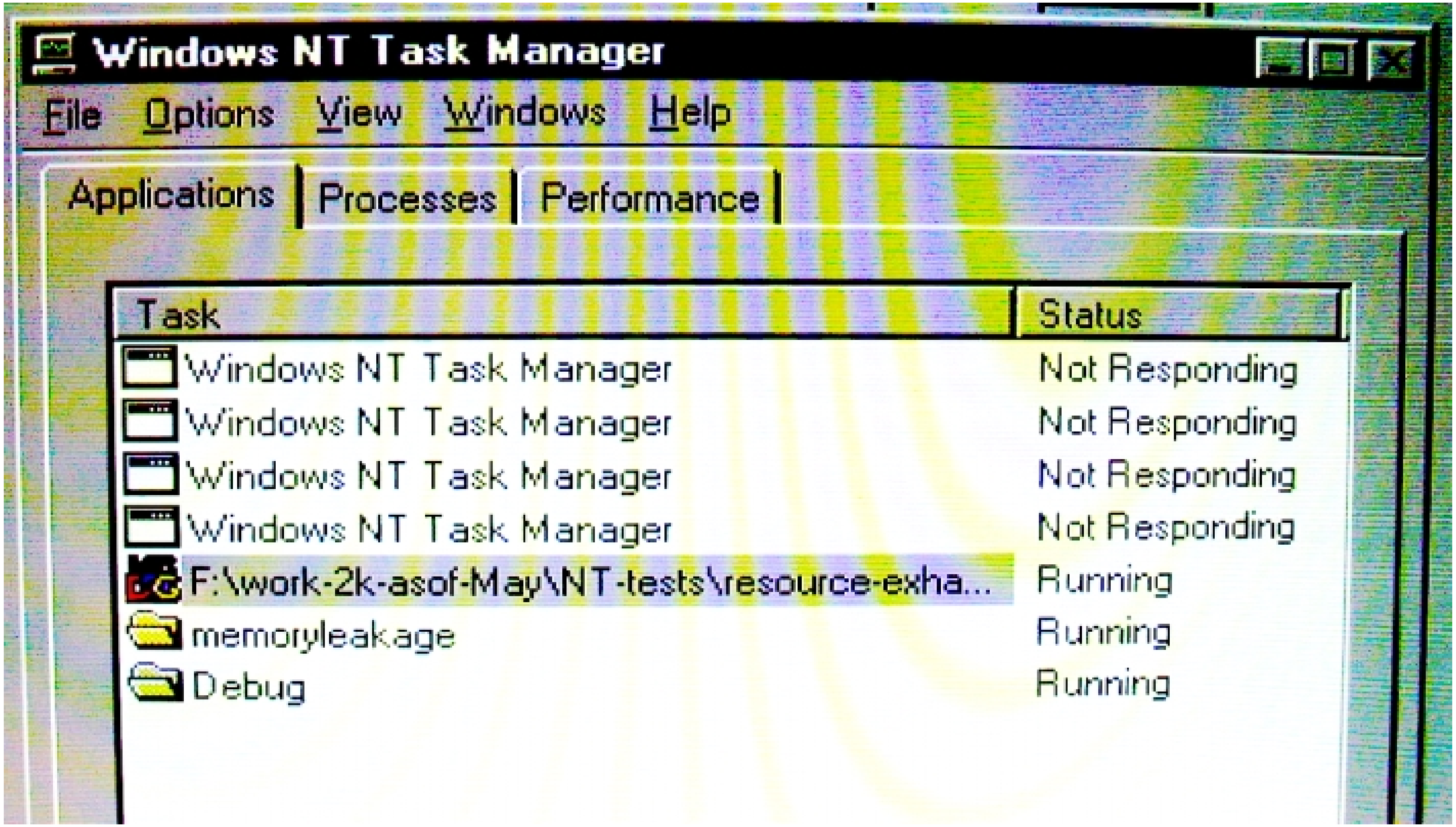
- Real-time Linux & Java as a candidate for spacecraft use:
  - Tested 266 methods; 232,570 tests per environment; java.lang
  - "Robustness failure" when exceptional inputs lead to unrecoverable Java task state
- Generic Baseline (Red Hat Linux+SUN JVM)
  - 4.7 % Robustness Failure Rate / No JVM crashes
  - Reasonably robust compared to:
    - POSIX Operating Systems 10-20 %
    - HLA-RTI (High Level Architecture Run Time Infrastructure) 10 %
- Proposed config. (Timesys Linux-RT GPL+RT-Java)
  - Some segmentation faults (impossible to handle in Java) – resulted in ***JVM crashes***
  - Other robustness failure rates comparable to generic version

Preliminary Wear-Out Testing

- Ran several concurrent copies of Ballista on Linux
  - Found little in way of races, wear-out
  - The one problem found was tracked down to non-reentrant exception handler that leaked memory buffers

- Windows wear-out testing found detection is improving
  - Win2K detected resource leaks much more quickly than Win NT
    - But, could be made to leak memory and even resource managers
  - WinXP looked even better on some very quick tests

  - (But, this work was just a preliminary investigation)


Windows NT Task Manager
File Options View Windows Help
Applications Processes Performance

Task	Status
Windows NT Task Manager	Not Responding
Windows NT Task Manager	Not Responding
Windows NT Task Manager	Not Responding
Windows NT Task Manager	Not Responding
F:\work-2k-asof-May\NT-tests\resource-exha...	Running
memoryleakage	Running
Debug	Running

Lessons Learned

- Memory protection really works
  - Inter-task memory protection provided excellent results
    - Problems we found were almost always easy to repeat and isolate
  - Task-to-kernel memory protection was good
    - But, API provided vulnerable spot (of course)
  - Operating systems with weaker or non-existent memory protection did poorly
  - No free lunch – triggering memory protection can make offending task unrecoverable

- Java isn't a silver bullet
  - JVM testing managed to crash JVM on Timesys RT-Java
  - Null pointers caused unrecoverable exceptions in commercial code [DeVale02]

BALLISTA

http://ballista.org