Hypervisor Support for Component-Based Operating Systems

Gabriel Parmer and Richard West

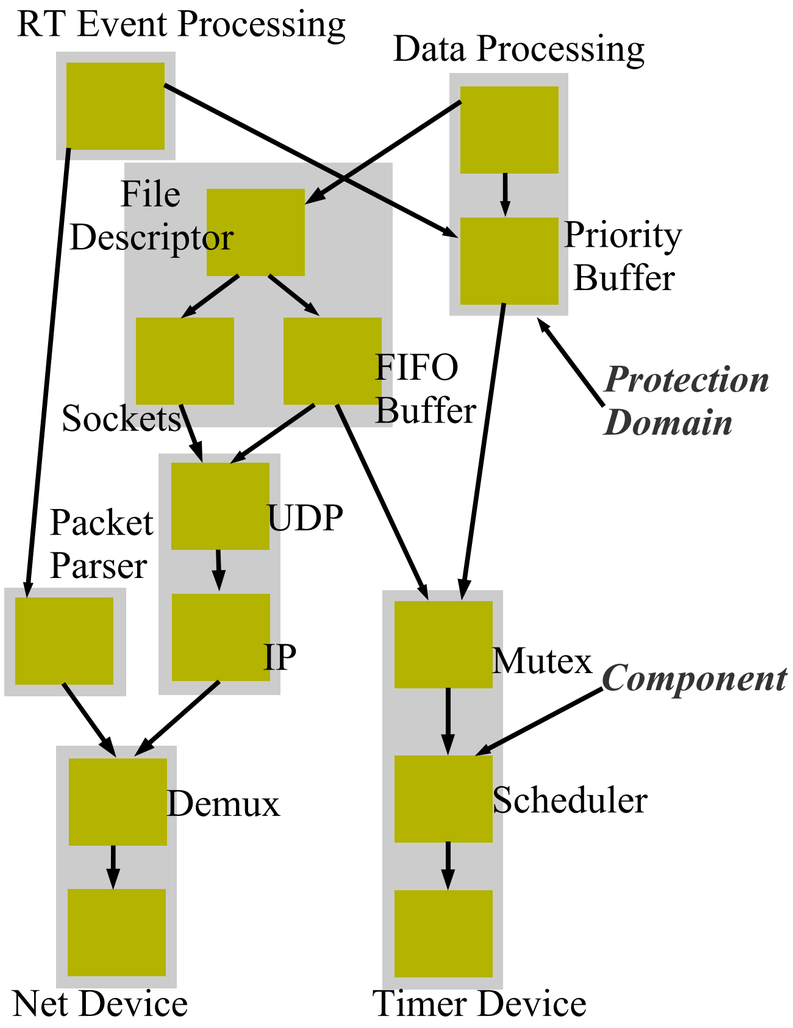
Component-Based Operating Systems

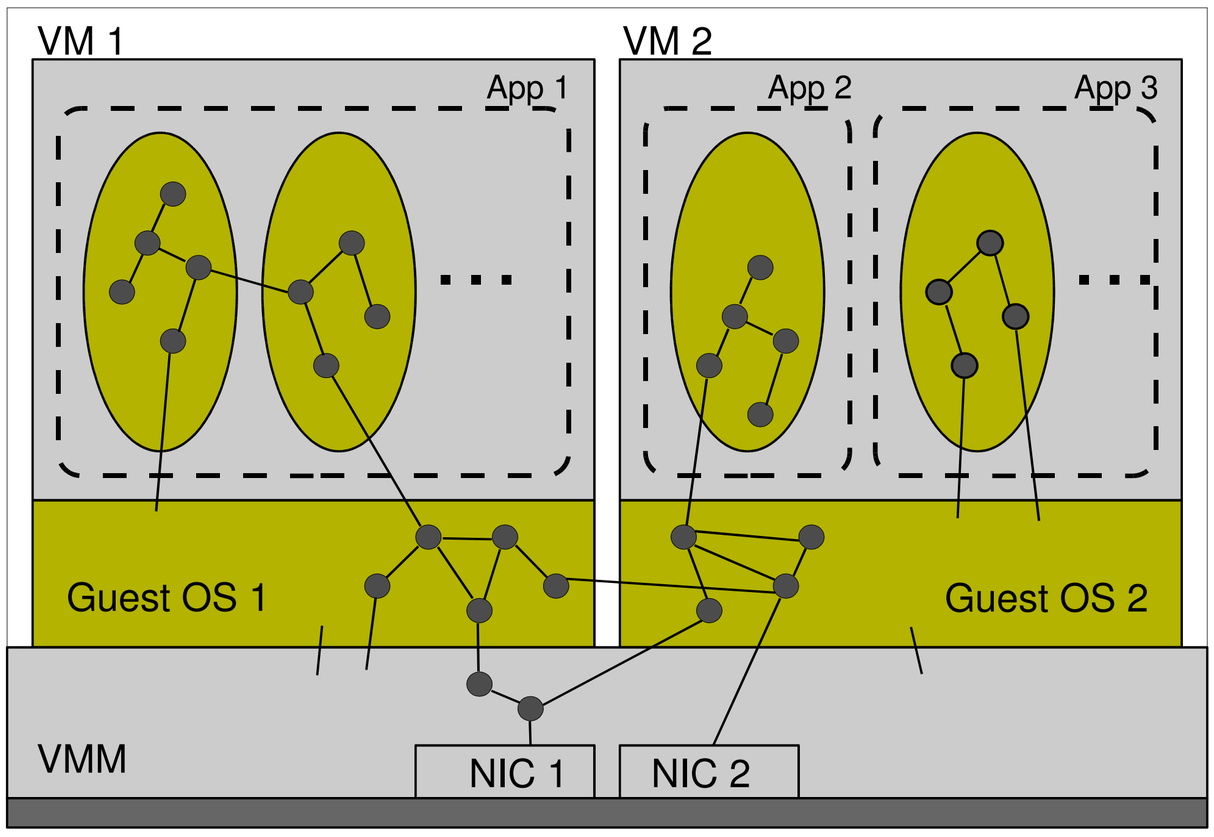
Entire system composed of components that
- provide functionality in the form of policies/mechanisms/abstractions
- advertise an interface and specify dependencies on external interfaces
- software eng. primitive for abstraction and reuse with broad definition

Trust relationship of the system is a graph of components rather than 2 tiered trust hierarchies in monolithic OSs, or 3 tiered in VMs

Component-Based Operating System (COS) are application specific
- minimal Trusted Code Base (TCB) and low memory footprint
- specialized system policies (for performance, security, dependability, ...)
- system creation by composition

Placement of protection domains in a COS?
- dependable systems require predictable recovery from faults
- protection domains limit the scope of errors





Composite OS

A Component-Based OS towards dependable computing
- there is trade-off and tension between
  1) fine-grained protection domains encapsulating every component and
  2) overhead from communication over protection domains that degrades protection (costs of IPC)

Mutable Protection Domains: *dynamically* place protection boundaries between components
- minimize the protection domain granularity (maximize fault isolation)
- while meeting performance goals (latency, throughput)

Application-specific system structure that adapts to run-time behavior to maximize dependability

Isolation Granularity

Scientific case for virtual machines
- application-oriented system structuring
  - offer isolation on the granularity of the application by encapsulating app processes/service dependencies

Problems?
- complete logical isolation between VMs containing apps and their services: *heavyweight*
  - switching between processes more expensive
  - switching between VMs can be even more so
- coarse grained isolation
  - errors within the application, trusted services, or guest OS errors still cause application failure

VMs + COSs?

Virtualize COS?
- communication across protection domains in COS heavily optimized
- optimistically inferred execution times (microsecs)

	Syscall	pg tbl	logic	ICC	Normalized
Native	0.14	0.33	0.16	0.63	1x
H/W	0.14	6.8	0.16	7.1	11.27x
S/W	1.29	1.07	0.16	2.52	4x

VMs as coarse-grained components?
- VMs special case of components
- component is first class VMM abstraction
- Example benefit: light-weight component VMs using controlled, paravirtualized interfaces
  - embedded, fine-grained protection domains, interfaces for sharing