Tagged/Capability Architectures

Shamik Das

28 March 2000

Overview

- What is a capability?
  What is a capability architecture?

- Examples of capabilities in use:
  - EROS
  - Symbolics 3600
  - guarded pointers

- Other capability architectures

- Questions

What is a Capability?

- *capability* – token, ticket or key that gives possessor permissions on an entity or object

- Implementation:
  – unique object identifier plus access rights for the object

- Properties of capabilities:
  – protected (unforgeable, unmodifiable)
  – context-independent
  – persistent after process exit
  – provide uniform access to shared resources

Access Models

- System Object Access Matrix

	File 1	File 2	File 3	Process X	Port Y	...
TK	read write	read write	read write	delete suspend revive	send receive	
Jeremy	read write	read	write		send	
bunnie					send receive	
JP	read		read		send	
⋮						

Access Models (cont.)

- Access Control Lists vs. Capabilities

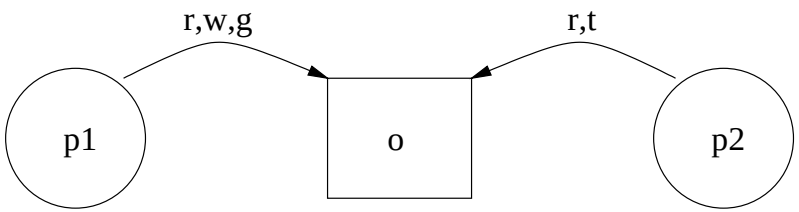
  Access Control Lists
  - system must maintain list for each object
  - lists must be kept secure
  - system must be able to verify users' identities
  - access rights are not transferable
  - access rights are easily revokable

  Capabilities
  - system only needs to verify capability
  - capabilities must be unforgeable
  - access rights are easily transferred
  - access rights are not easily revoked

Access Models (cont.) – *Take-Grant*

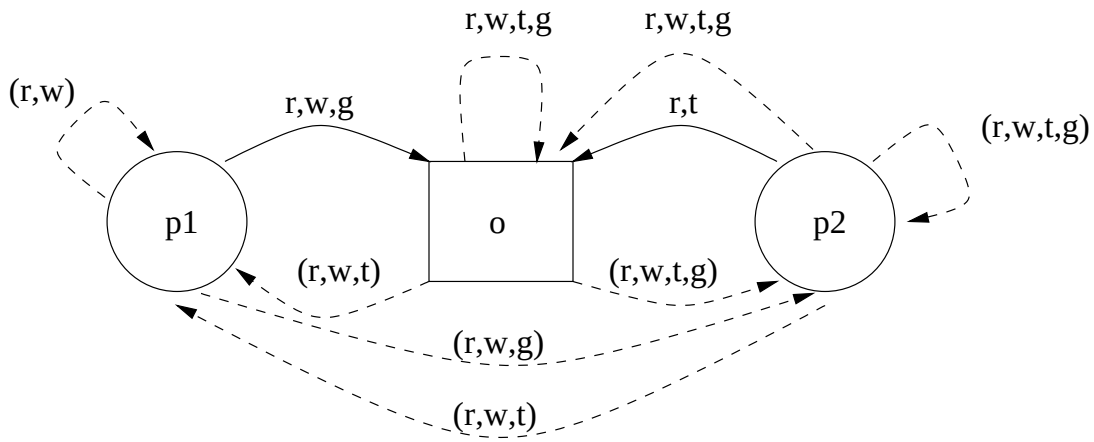
- Formal Framework: *Take-Grant* model
  - **read, write** capabilities on data
  - **take, grant** capabilities on capabilities

- Access Graph:



Take-Grant (cont.)

- Implied Permissions:
  - *de jure* access
  - *de facto* access

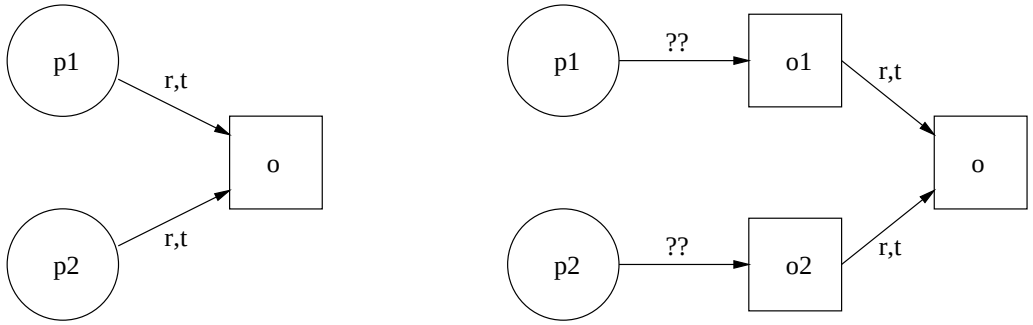


- Assumptions:
  - presumed collusion

Issues with *Take-Grant*

- Security Verification
  - run-time verification requires reference monitor

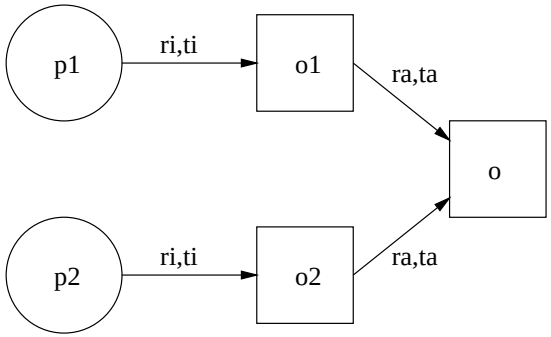
- Selective Access Revocation



- Possible Remedies
  - capability versioning
  - indirection (*Diminished-take* model)

Diminished-Take model

- Indirection
  - **ra, wa, ta, ga** direct rights
  - **ri, wi, ti, gi** indirect rights

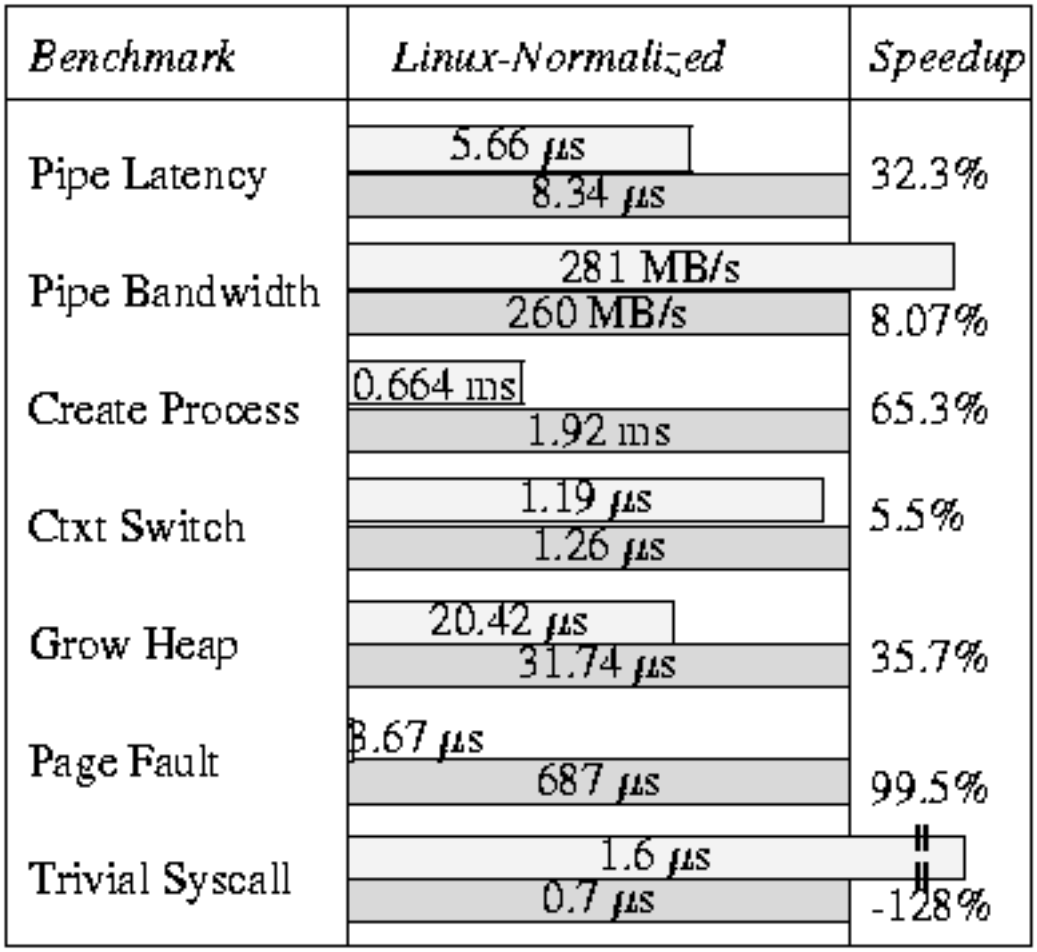


- Diminish Operation
  - **dta, dti, dga, dgi** diminished rights
  - $diminish(r) = r \cap \{\mathbf{ri}, \mathbf{ra}, \mathbf{dti}, \mathbf{dta}\}$
  - reduces need for reference monitor

The Extremely Reliable Operating System (EROS)

- Capability-Based OS
  - runs on commodity processors
  - *diminished-take* capability model
  - universal persistence through checkpointing

EROS Benchmarks



Symbolics 3600 Lisp Machine

- Lisp-based instruction set
  - most Lisp operations run in one cycle
- tagged data and run-time type checking
- generic instructions
  - one instruction covers all applicable data types
- hardware-assisted garbage collection

Garbage Collection

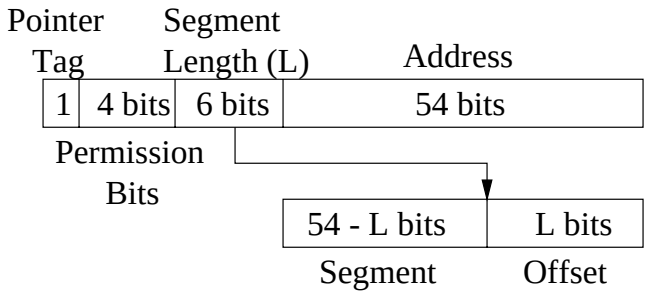
- Hardware support
  - type fields
  - page tags
  - multiword read instructions

- rapid identification of pointers

- rapid page scan

Guarded Pointers

- issue: sharing across threads vs. sharing across processes
- pointer architecture



- requires ISA modifications/extensions
- example: M-machine memory system
  - allows for zero-cost context switching

Other Capability-Based Systems

Burroughs B5000	segmented memory, segment descriptors
MIT PDP 1	supervisor instructions, C-lists
IBM System 38	augmented ISA and tagged memory
Intel i432	two-part (data+capability) segmentation
Mach	microkernel O/S with capabilities for ports
Amoeba	object-based distributed O/S