

Protection in the Hydra OS

CMU

Hydra OS

- Hydra microkernel – intended for OS research and experimentation
 - CMU multiprocessor machine
- Separation of policy and mechanism
 - e.g., low-level dispatching but no scheduling policies
- Use of OO model with capability-based protection
 - Everything is an Object. Objects can be accessed through capabilities. Capabilities include unique ID and access rights
 - Protection at granularity of type of operation
 - Capability-based vs. Access-Control Lists based

Hydra Objects

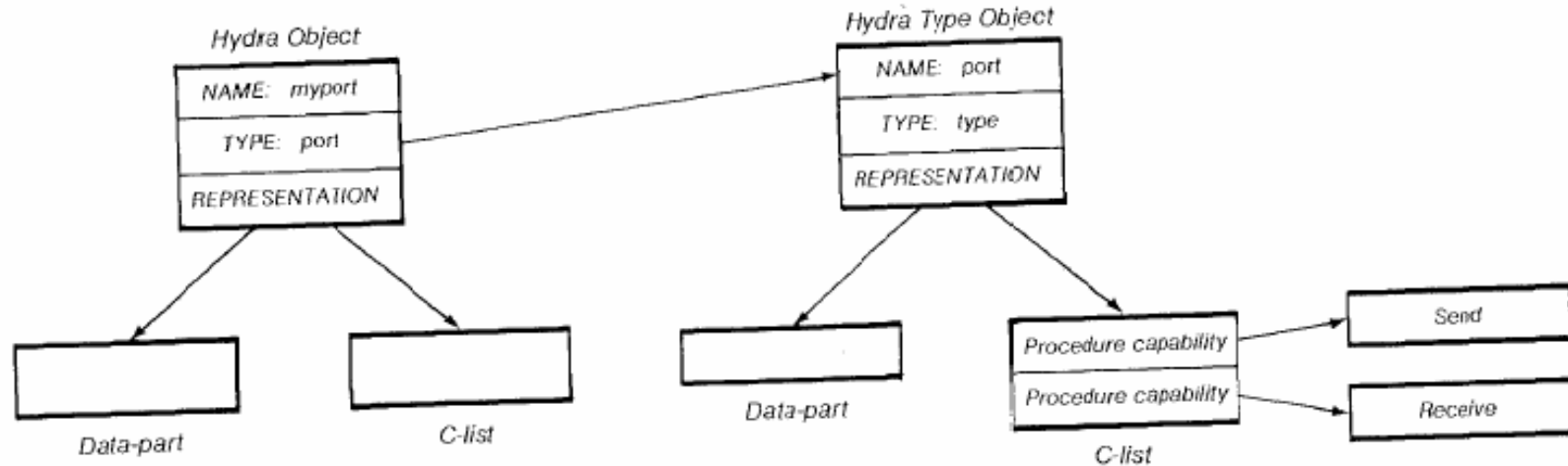


Figure 6-1: Hydra Object and Type Object

PROCESS	The basic unit of scheduling and execution.	PORT	A message transmission and reception facility.
PROCEDURE	The static description of an executable procedure.	DEVICE	A physical I/O device.
LOCAL NAME SPACE (LNS)	The dynamic representation of an executing procedure.	POLICY	A module that can make high-level scheduling policy decisions.
PAGE	A virtual page of C.mmp memory that can be directly accessed.	DATA	An object with a data-part only.
SEMAPHORE	A synchronization primitive.	UNIVERSAL	A basic object with both a C-list and data-part.
		TYPE	The representative for all objects of a given type.

- Protection is at procedure granularity
- Dynamic; LNSs form activation stack

Types

- Default kernel types
- + user instantiated type subsystems (type managers)
- Type manager responsible for creation of objects of given type and provides operations/procedures supported on those objects
- At creation time, capability is created for the object (unique, time + processor ID), and returned to caller
 - Though rights may need to be restricted/reduced before returning to caller

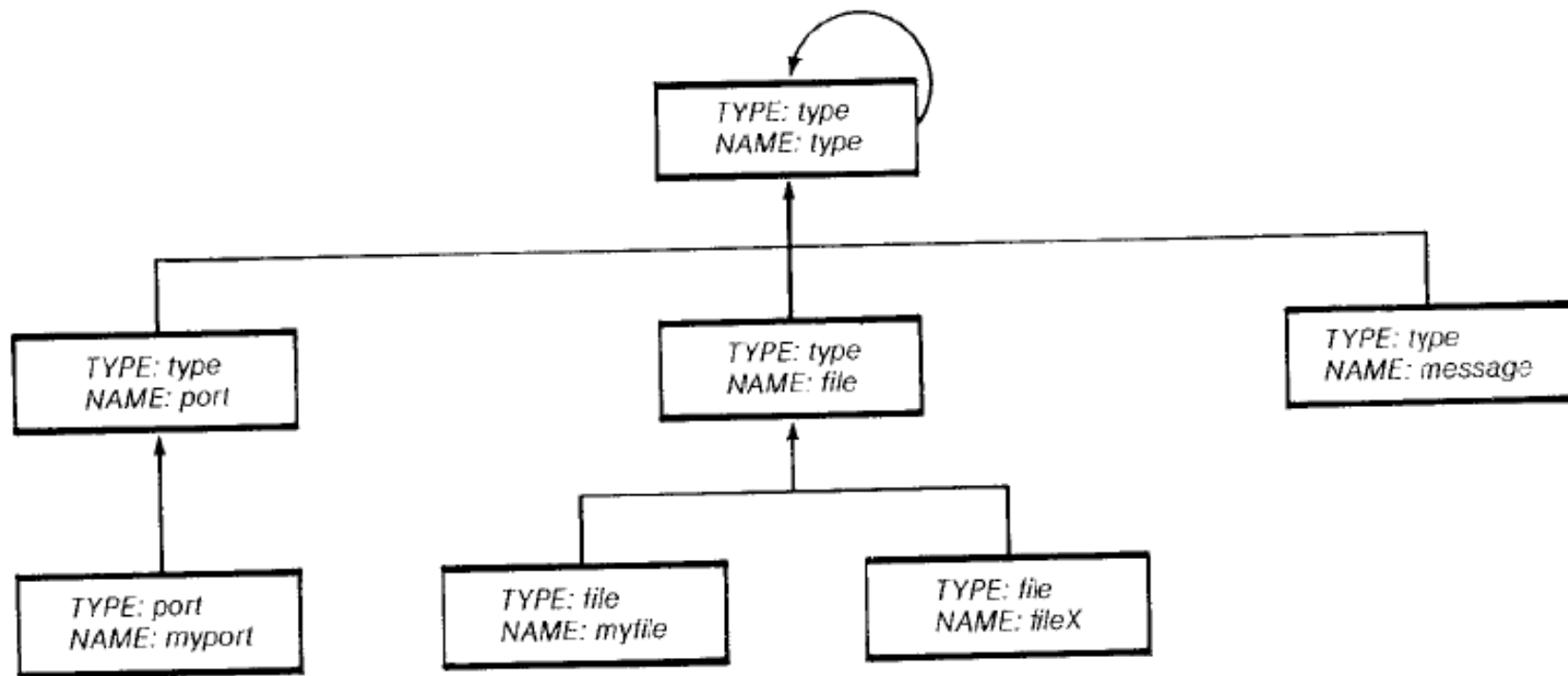


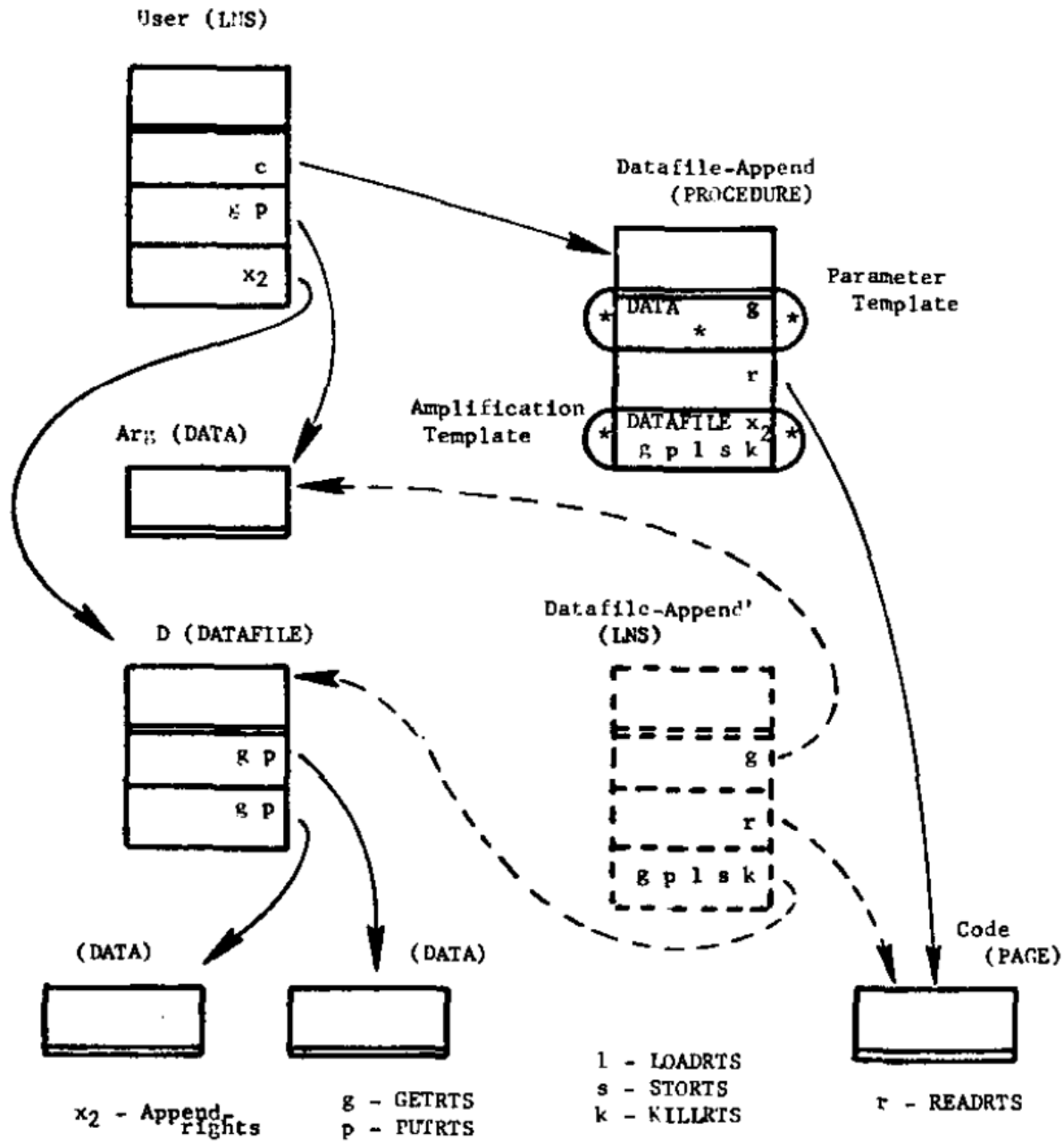
Figure 6-2: Hydra Type Hierarchy

Capabilities

- Unique ID + access rights
- 16 bits for generic rights – determine rights for executing some of generic operations supported on all objects: Get/Put/AppendData/Cap; DeleteRts, KillRts, CopyRts, ...
- 8 bits for auxiliary rights – interpreted in a type-dependent manner
- LNS specifies all objects and associated access rights which can be accessed at any given point in execution
- Single procedure may have different active LNSs with different sets of capabilities

Templates

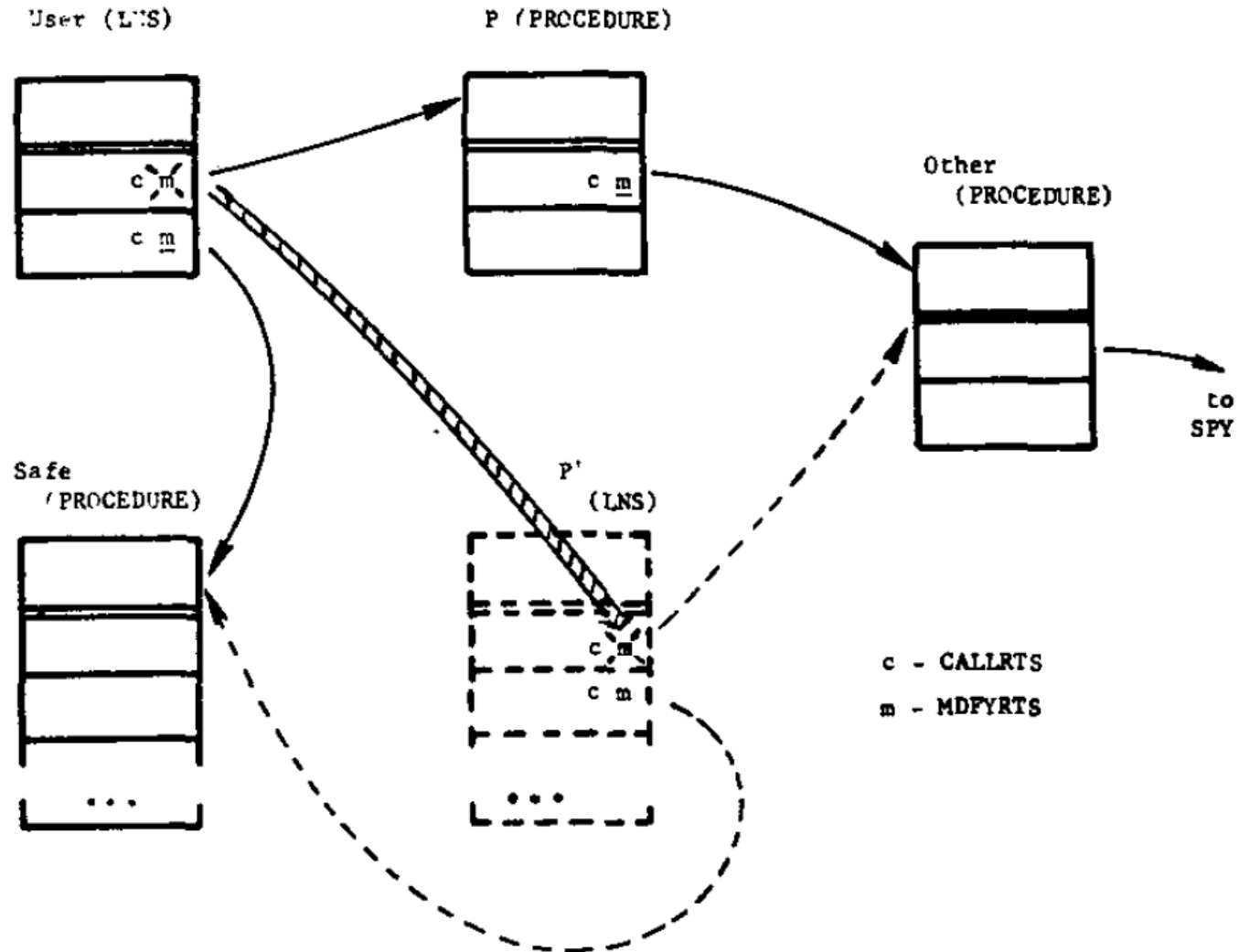
- Procedure has private data/Hydra objects it “owns”, but also parameters
- Parameter templates – specify type of capability and minimum access rights for parameter to procedure
 - If parameter passed has more rights, minimum will be used
- Hydra does capability type/rights checking on \$CALL
- Amplification templates – augment rights of passed parameter so as to permit certain data access/manipulation within the procedure
 - Allowed only for trusted subsystems/type managers
- Creation templates – Hydra creates objects based on template on call \$CREATE, then type manager performs initialization and may restrict rights as necessary



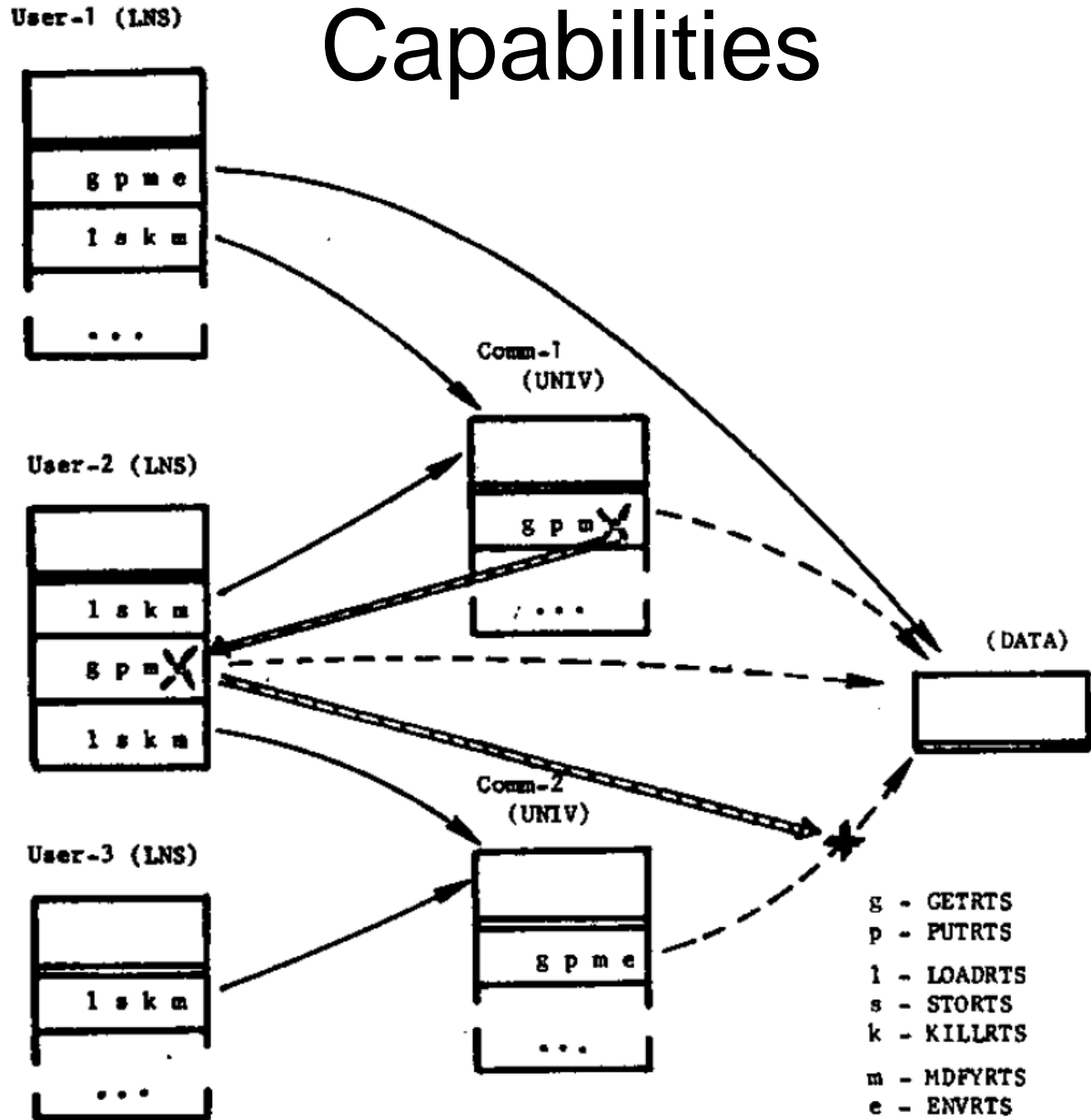
Some special rights

- Separation of policy and mechanism – Hydra attempts to ensure there are sufficient mechanisms for all sorts of policies, useful, or not so much...
- -> Lots of access rights
- MDFYRTS – permission to modify rights in a capability
- ENVRTS – permission to copy capability to another object's C-list
- UCNFRTS – if removed for procedure capability, that capability will not be able to modify any of its own objects, -> will not be able to copy information from its parameters
- Use of these is useful for range of protection policies

e.g., Confinement



e.g., Limiting Propagation of Capabilities



Revocation

- Permanent, temporal, selective, partial, revocation of rights to revoke...
- Solution required a new mechanism (beyond kernel interpreting rights) - Aliases

