BiiN

Object

Computing

BiiN™

Rev. 1.0
FJP
Foil 1

Object Computing Benefits

- Productivity
  - Reusability
  - Maintainability
  - Extensibility
- Reliability

And

BiiN™

Object Computing Benefits

- To Store Knowledge

But

- OS File Structures and Relational DBMS are Inadequate
- Purely Software-based Persistent Object Support
  - is inefficient, and
  - does not address multi-user access

And

BiiN™

Object Computing Benefits

- To Construct Secure Systems

But

- Requires Run-time Enforcement
- Current Hardware Architectures do not Support Efficient Fine-Grained Protection

BiiN™

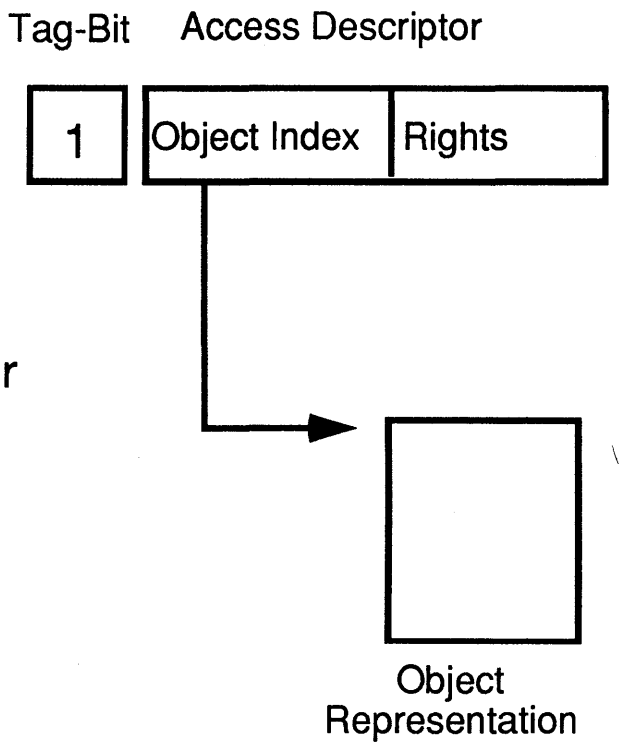
Topics

- Object Addressing and Protection
- Computational Model
- Type Manager Based Protection
- Inheritance
- Object Persistence

BiiN™

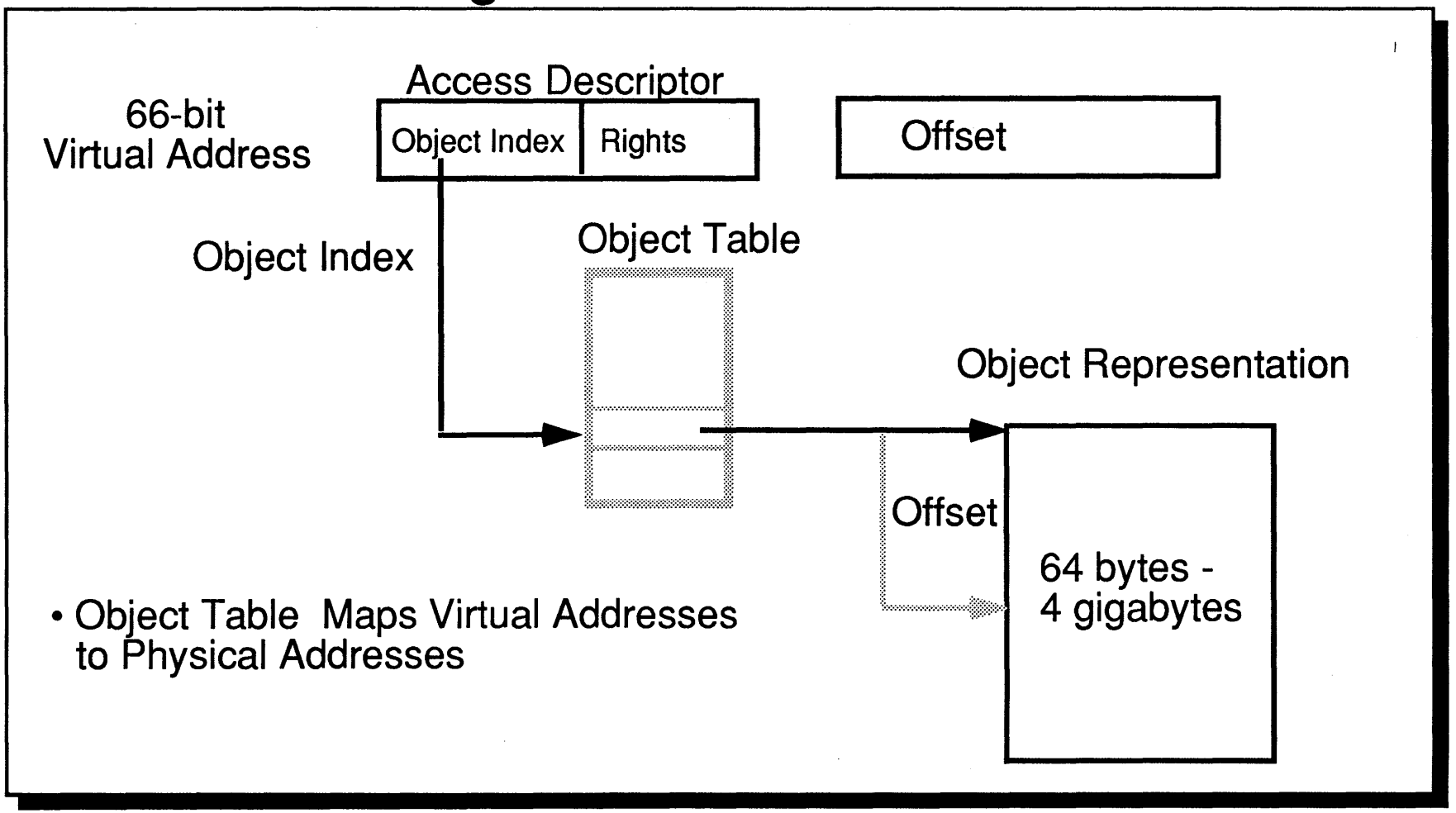
VLSI-Based
Object Addressing and Protection

- Object is a Typed Memory Segment
  - From 64 to 4 gigabytes
  - If >4K bytes, then paged
  - up to 64M objects in virtual address space
- Object Accessed through Access Descriptor
  - AD is unforgeable (33rd bit)
  - AD is used just like a pointer – Access type in Ada
- Rights Determine Allowed Operations



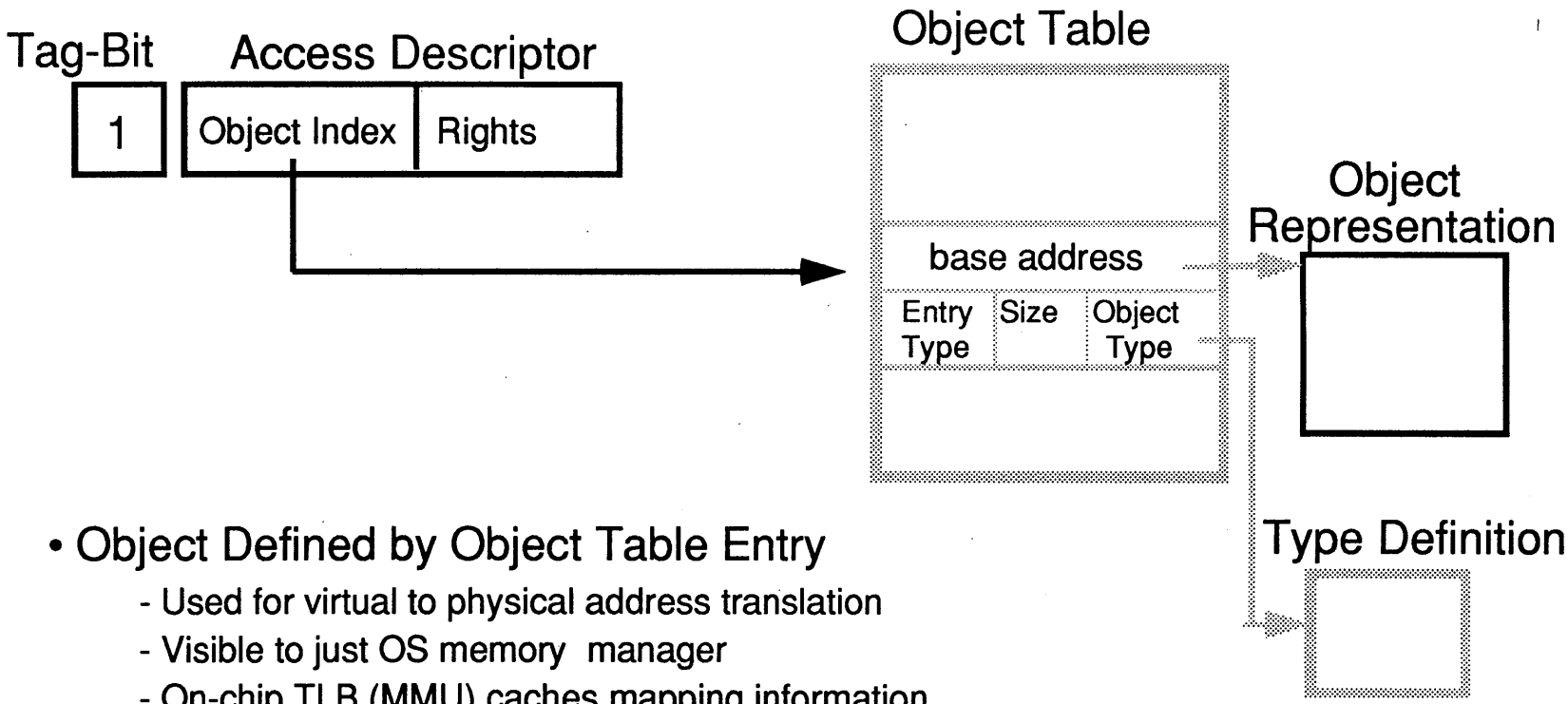
BiiN™

Virtual Addressing



BiiN™

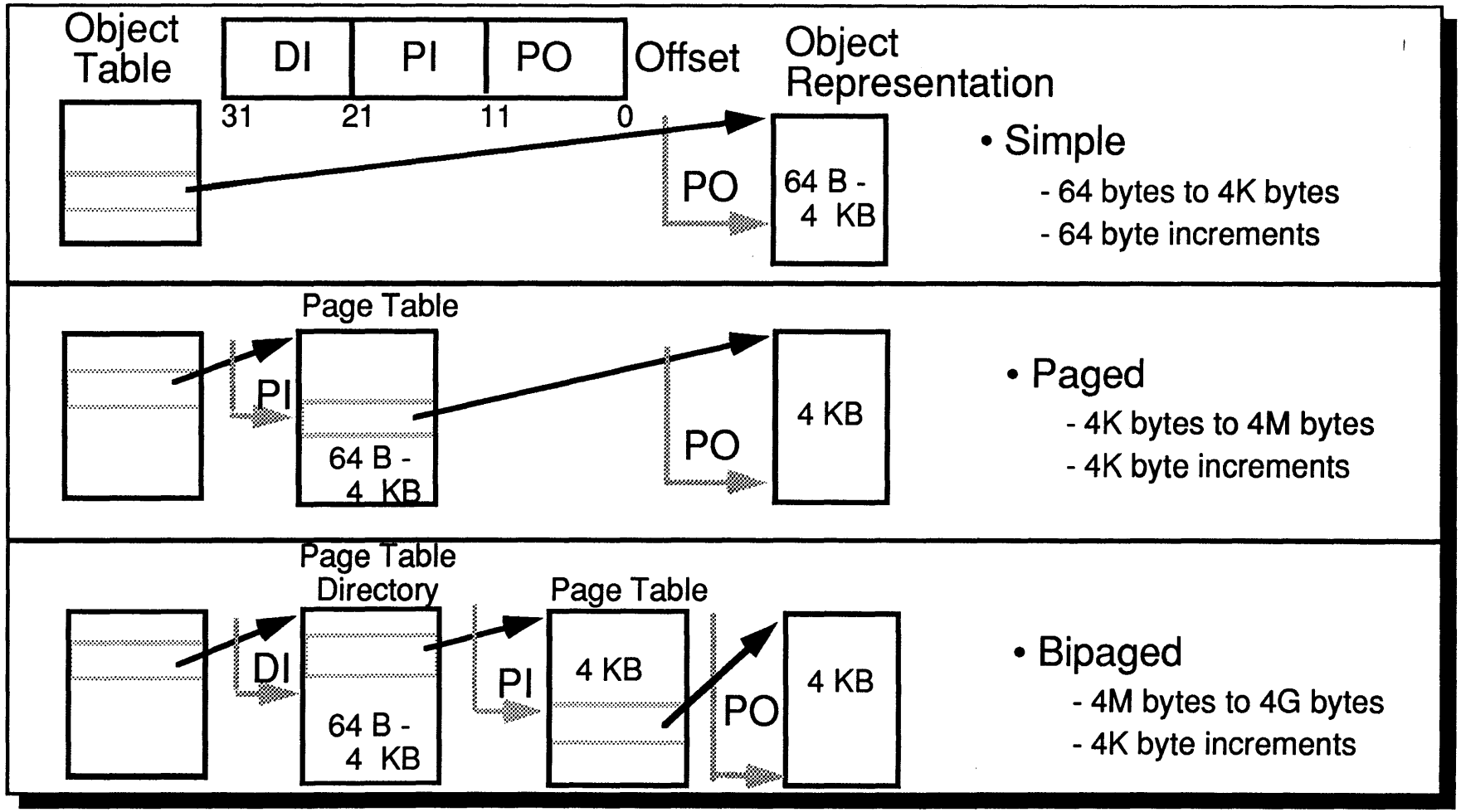
Object Structure



- Object Defined by Object Table Entry
  - Used for virtual to physical address translation
  - Visible to just OS memory manager
  - On-chip TLB (MMU) caches mapping information
- Translation is Independent of Execution Context
  - TLB not flushed on process switch

BiiN™

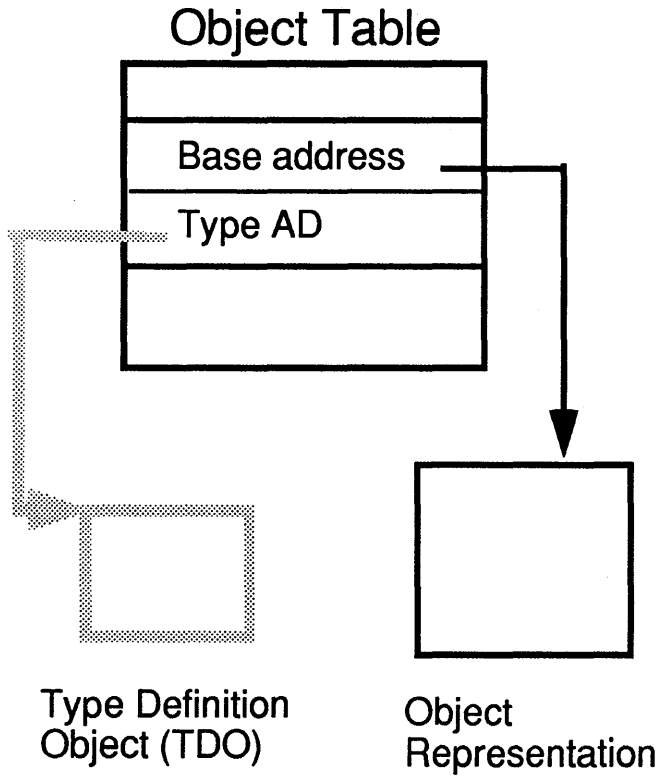
Address Translation



BiiN™

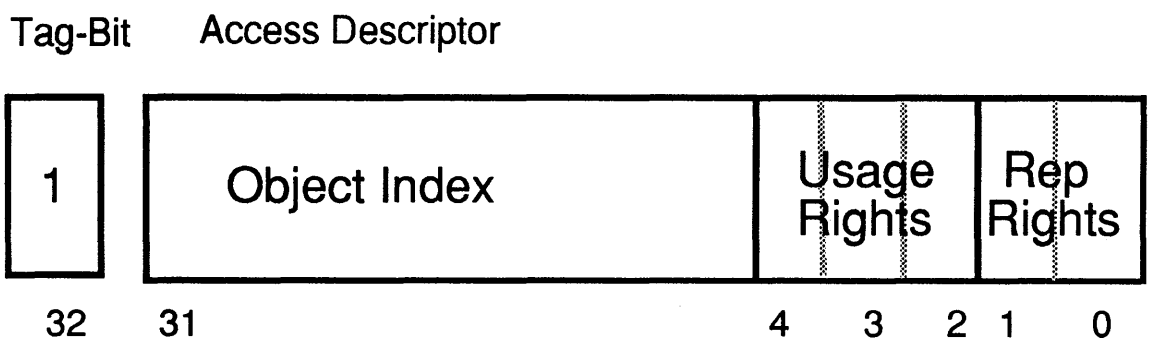
Object Typing

- Object Types are Defined by Type Definition Objects (TDO)
- Each Object Descriptor Contains an AD Pointing to the TDO of itsType
- H/W recognized types:
  - Semaphores
  - Ports
  - Processes
  - Domains
  - TDOs



BiiN™

Representation Rights



- Representation Rights Control Direct Access to Object
  - Read rights checked on Load instructions
  - Write rights checked on Store Instructions
- Rights Checking Occurs in Parallel with Address Translation

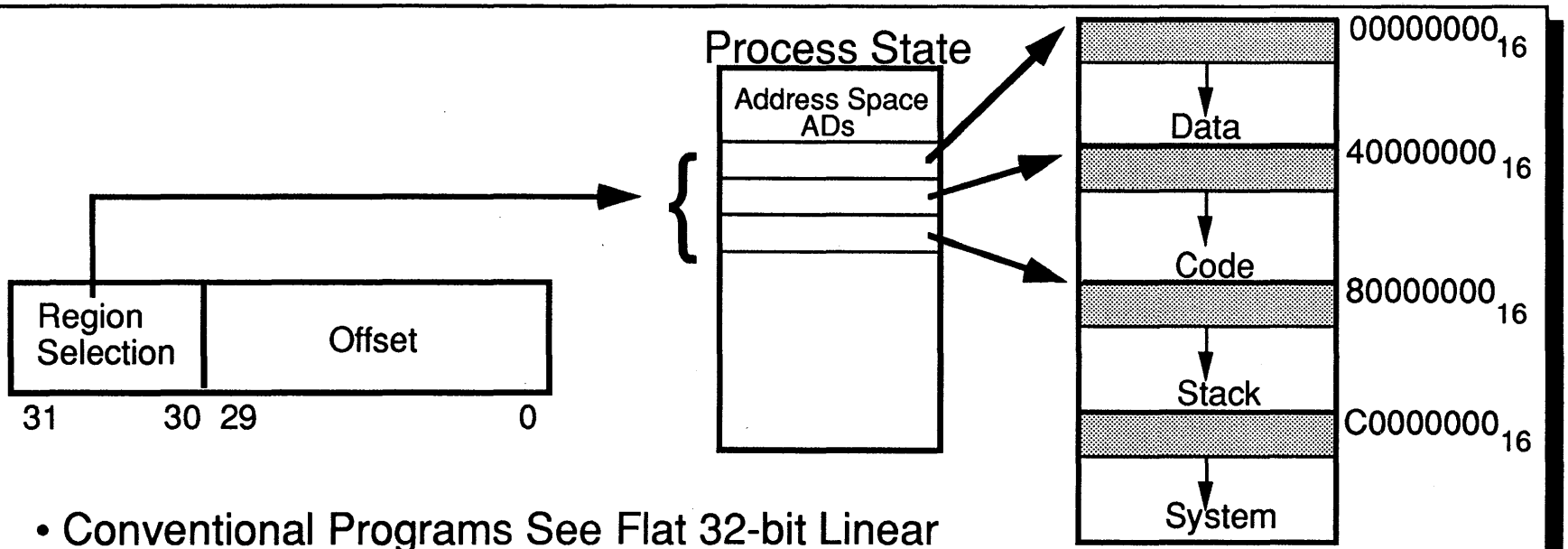
BiiN™

Topics

- Object Addressing and Protection
- **Computational Model**
- Type Manager Based Protection
- Inheritance
- Object Persistence

BiiN™

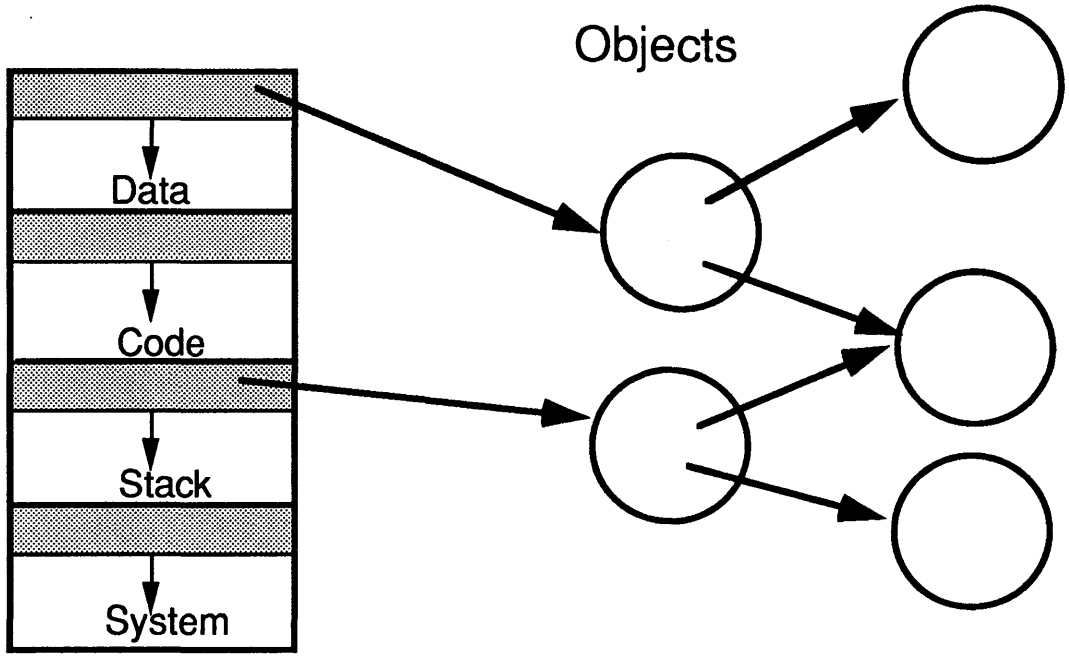
Simple Program Model



- Conventional Programs See Flat 32-bit Linear Address Space
- Each of 3 Regions is an Object
- Regular Call-Return Used for Procedure Calls
- Reserved Region is Processor-specific

BiiN™

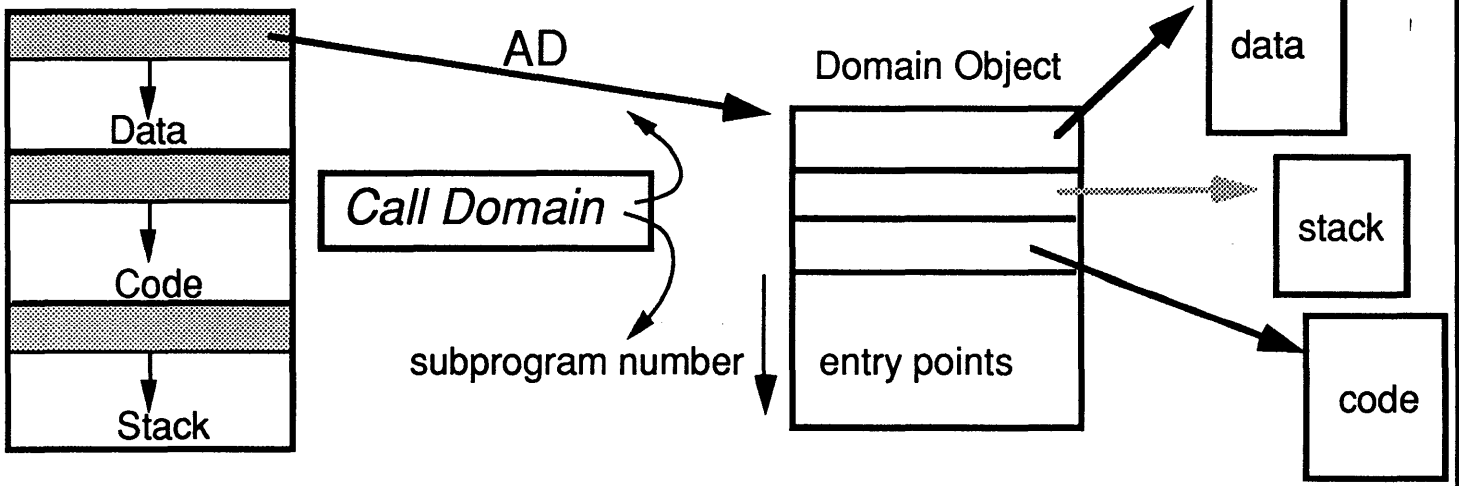
Extended Program Model



- Linear Address Space Can be Root of an Object Network
  - Interconnected by ADs
  - ADs used just like pointers

BiiN™

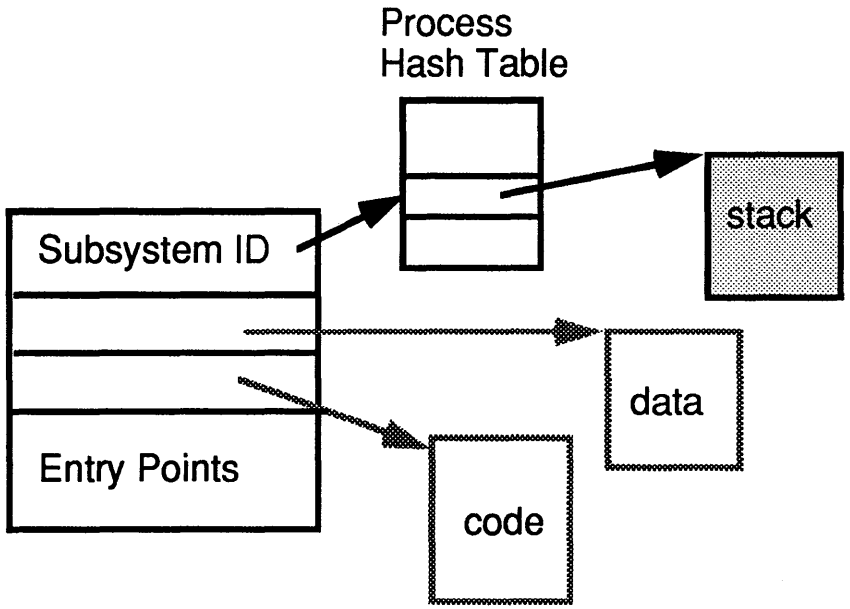
Multiple Address Spaces Per Process



- Destination Address Space Defined by Domain Object
- Interdomain Call/Return Instructions
  - Linkage kept in process object
  - No implicit access between address spaces
  - Parameters may include ADs
  - Performance comparable to other architectures' supervisor calls
- No Limit on Number of Address Spaces per Process

BiiN™

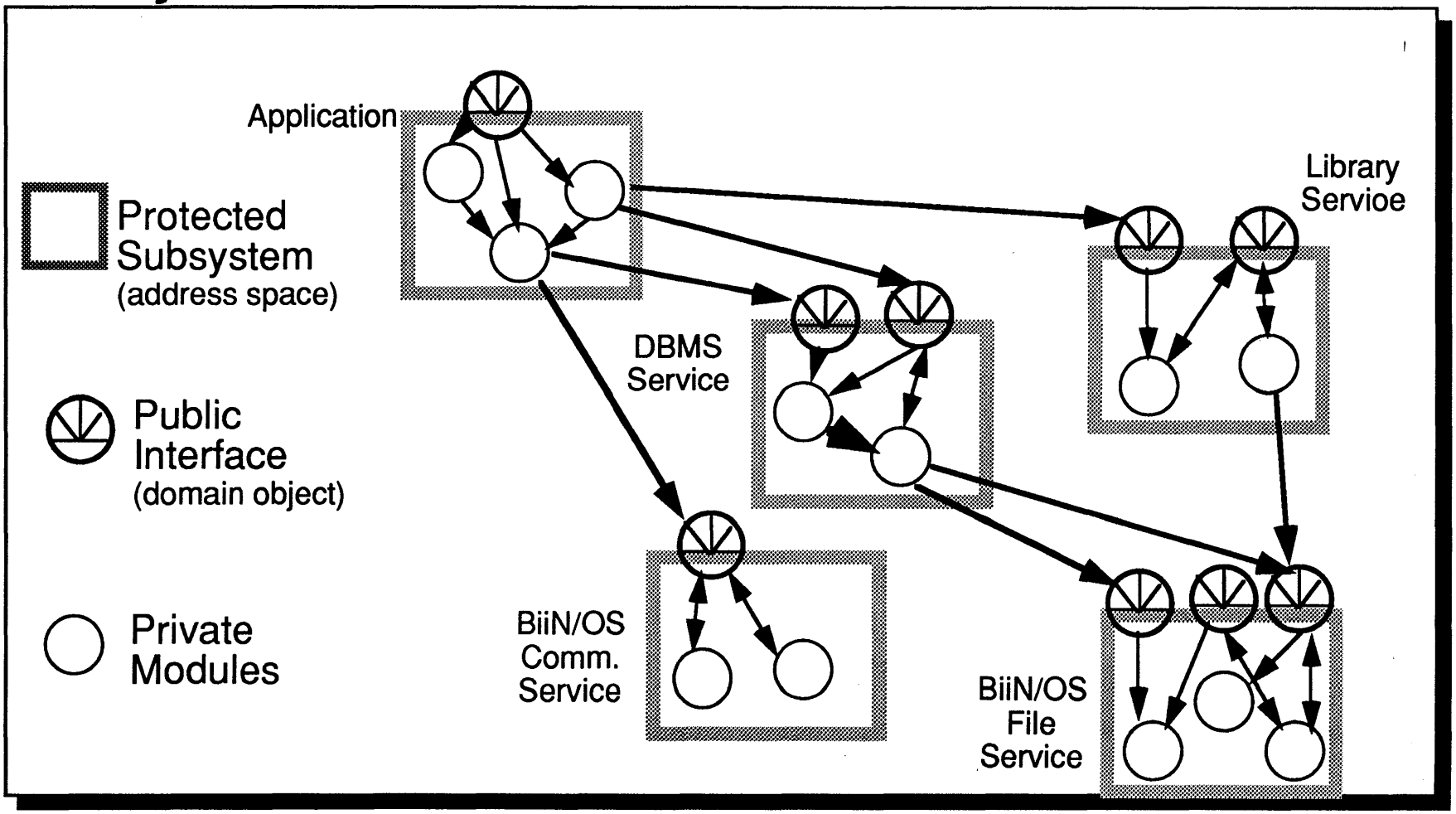
Closer Look At Call Domain



- Subsystem_ID Selects Stack from Process Object
  - Null: use caller's stack
  - World: use the default program stack
  - ID: A unique (named) stack is selected per process

BiiN™

Subsystem Based Protection



BiiN™

Subsystem Based Protection Benefits

- Reliability, Maintainability, and Extensibility Without Compromising Performance
- Increased Productivity in Integration and Test
  - Decompose application into protected subsystems
  - Since each subsystem is linked independently, turnaround time (recompile/relink) is faster
  - Since errors confined to subsystem, they are easier to find
- Increased Performance Without Compromising Security
  - Services can safely execute in user's process
  - Other architectures require separate process, which results in:
    - higher invocation overhead
    - potential bottlenecks in symmetric multiprocessors

BiiN™

Topics

- Object Addressing and Protection
- Computational Model
- **Type Manager Based Protection**
- Inheritance
- Object Persistence

BiiN™

Object-Oriented Design

- Define Abstract Data Type
- Define Set of Operations on Type
- Set of Operations form a Module
- Module Hides Implementation
  - Representation of data type
  - Operations (Algorithms) on data types

BiiN™

Mapping to Ada

Object-oriented Design

- Define Abstract Data Type
- Define Set of Operations on Type
- Set of Operations form a Module
- Module Hides Implementation
  - Representation of data type
  - Operations (Algorithms) on data types

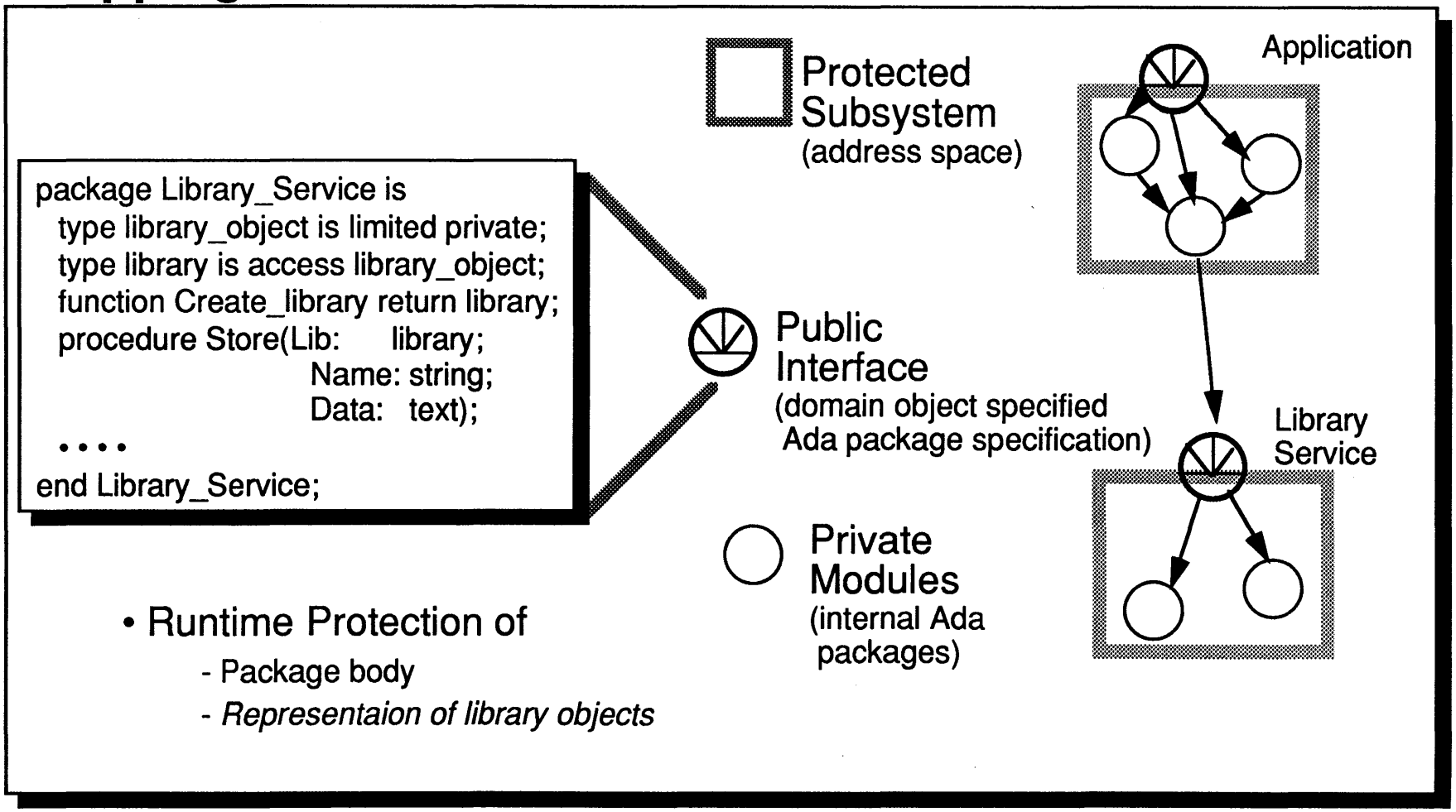
Maps to Ada Package

```
package Library_Service is
   type library_object is limited private;
   type library is access library_object;
   function Create_library return library;
   procedure Store(Lib:    library;
                   Name: string;
                   Data:  text);
   ....
end Library_Service;

package body Library_Service is
   -- Contains Implementation
   -- Hidden from users of package
```

BiiN™

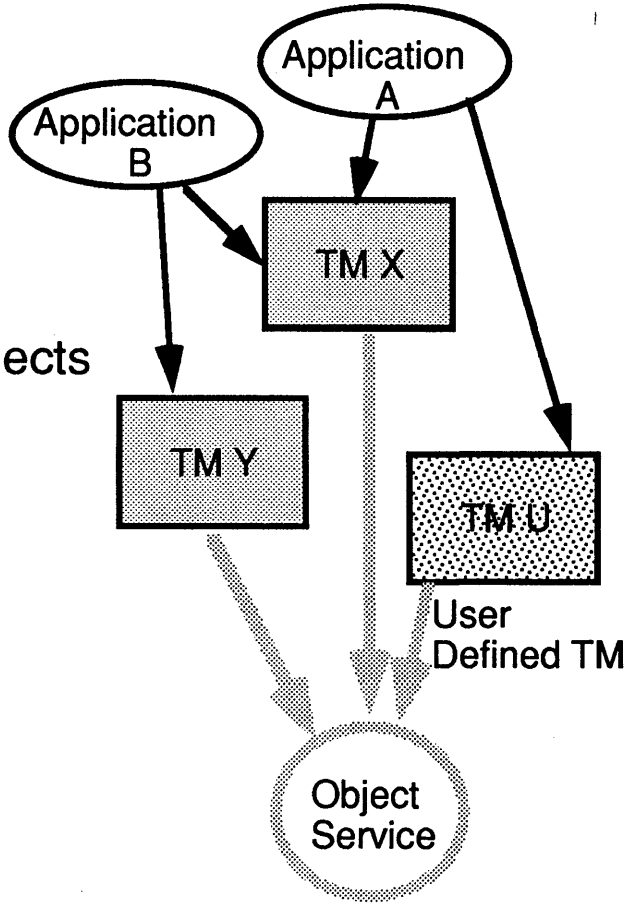
Mapping To Architecture



BiiN™

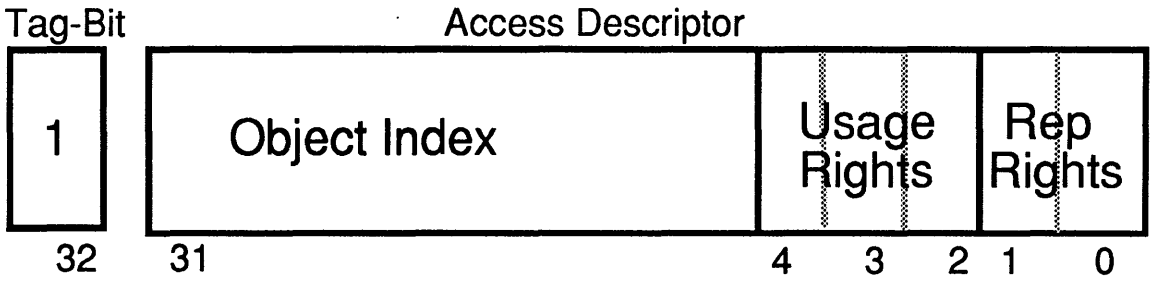
Principles Of
Type Manager Based Protection

- Objects are Typed
- The *X Service* is the Type Manager (TM) for Objects of Type X
- Only TM X Can Access Representation of X Objects
- Applications Can Pass Around ADs (without Representation Rights) for X objects
- Anyone Can Create a New Object Type and TM
- BiiN/OS Provides Object Management Service



BiiN™

Usage Rights



Usage Rights are Interpreted and Checked By Object's TM

Examples

TYPE	USE	MODIFY	CONTROL
FILE	READ	WRITE	DELETE
TDO	CREATE	AMPLIFY	-
DIRECTORY	LIST	STORE	CHANGE A-LIST
PROGRAM	EXECUTE	DEBUG	DESTROY
LIBRARY	LOOKUP	STORE	LOCK

BiiN™

A TM Example Using Libraries

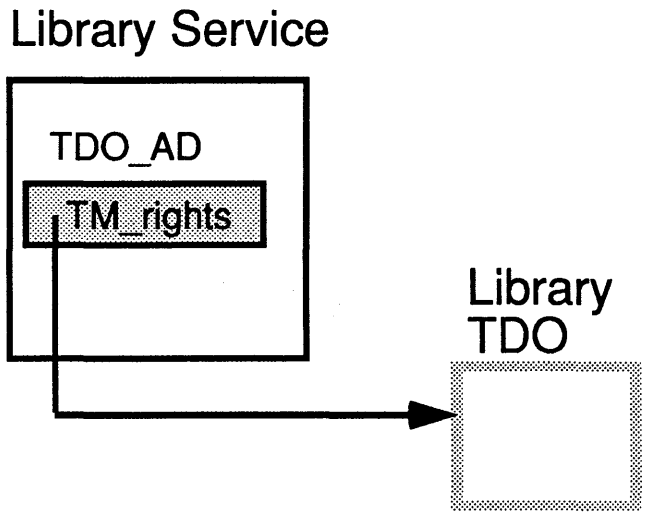
Outline

- Creation of the Library Type
- Creating an Object of Type Library
- User-Level Protection on Library Objects
- Invoking the Lookup Procedure on a Library Object

BiiN™

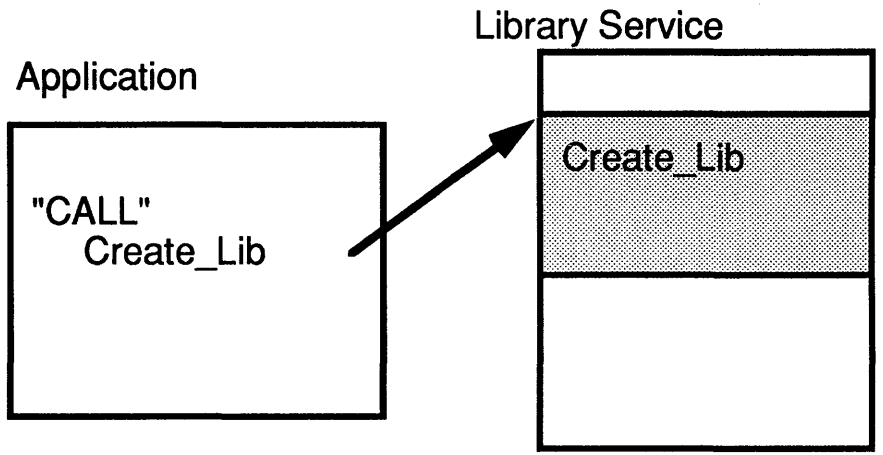
Type Creation

- To Define a New Data Type, TM Creates a Type Definition Object (TDO)
  - AD to TDO is never given out
  - Usage Rights of that AD have TM_rights semantics
    - Create (use) right
    - Amplify (modify) right



BiiN™

Object Creation

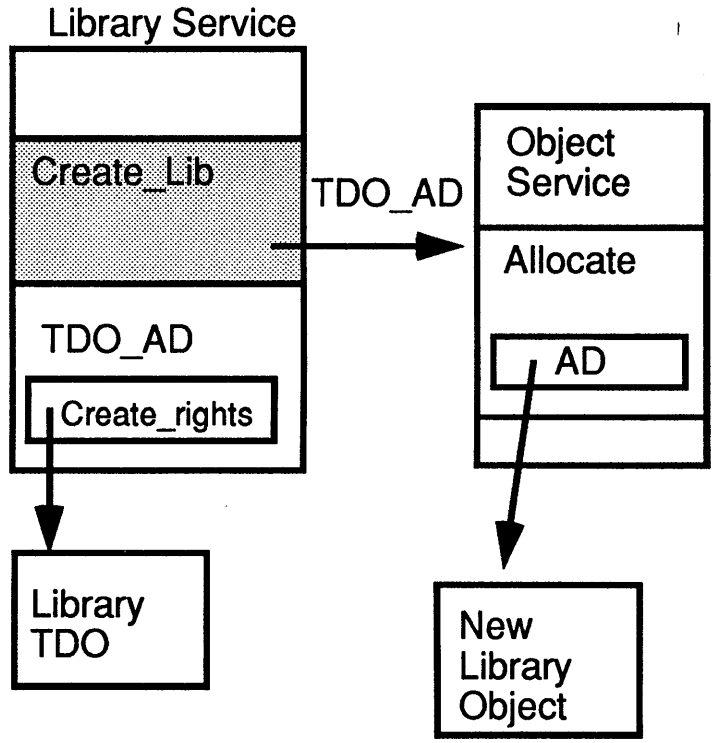


- Application Cannot Create Library Object Directly
  - It cannot get AD with create rights for Library TDO
- Application Must Call Create_Library Subprogram
- Call Requires AD for Library Service Domain Object

BiiN™

Object Creation (cont'd)

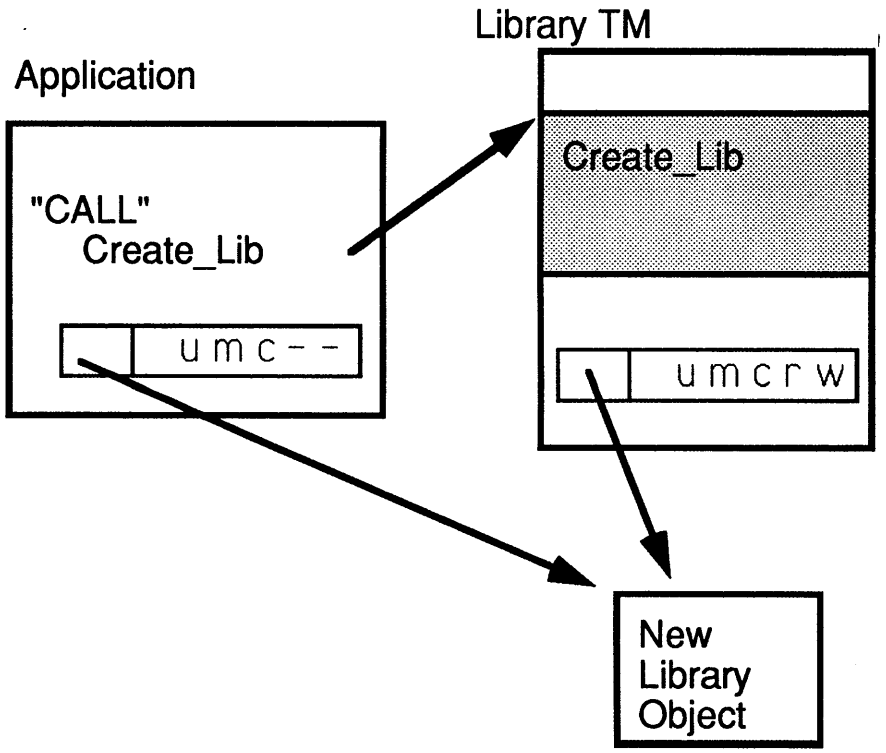
- Library Service Calls BiiN/OS Object Service
  - AD for Library TDO is passed
  - Object service checks for create_rights in AD
- Object Service Allocates Library Object
  - Returns AD with all rights to Library TM



BiiN™

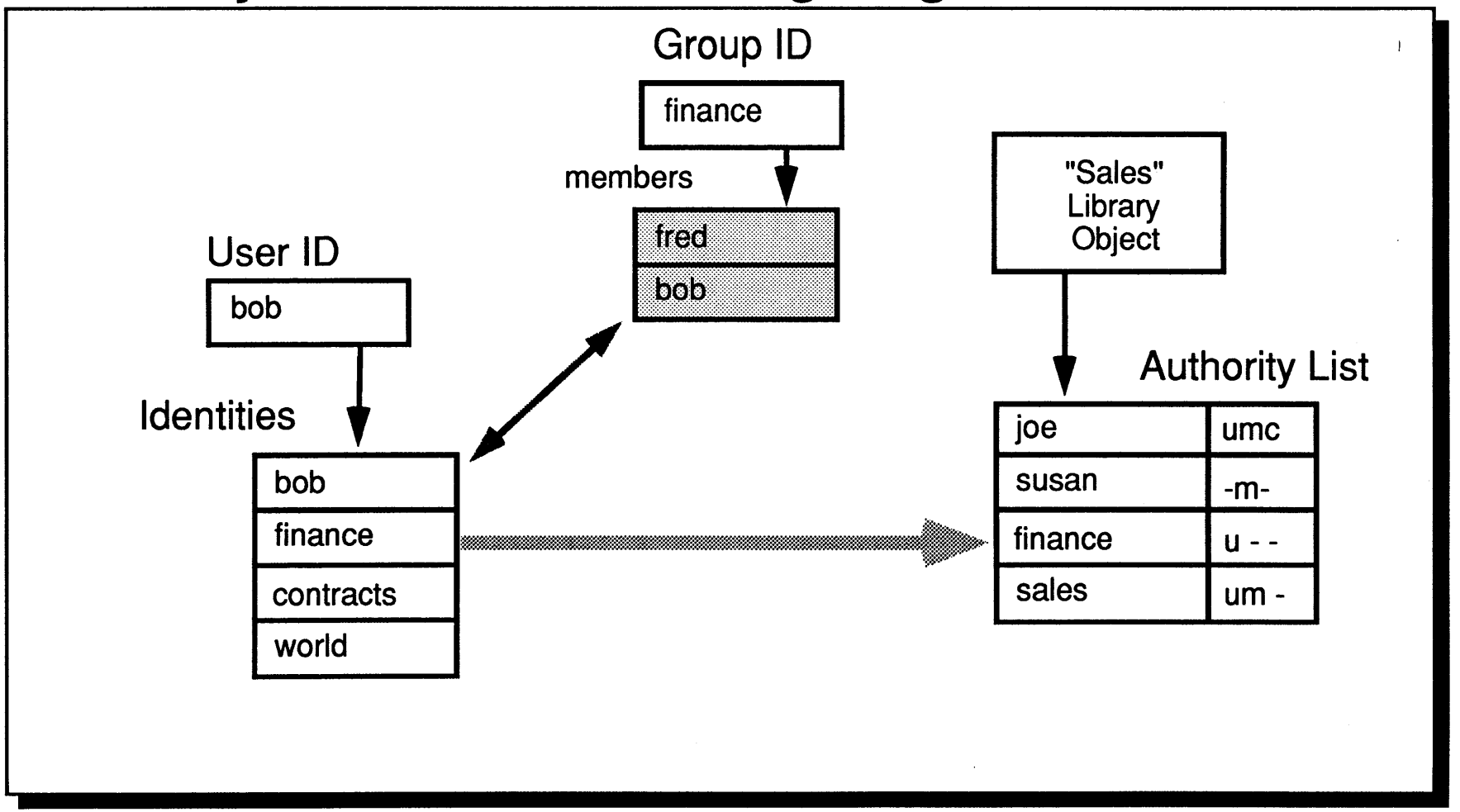
Object Creation (cont'd)

- Library Service Then
  - Initializes library object
  - Removes representation rights in AD (using restrict rights instruction)
  - Returns AD to application
  - Does NOT keep AD to library object
- Application Cannot Access Representation
- Application Controls Usage Rights to Just its Library Object



BiiN™

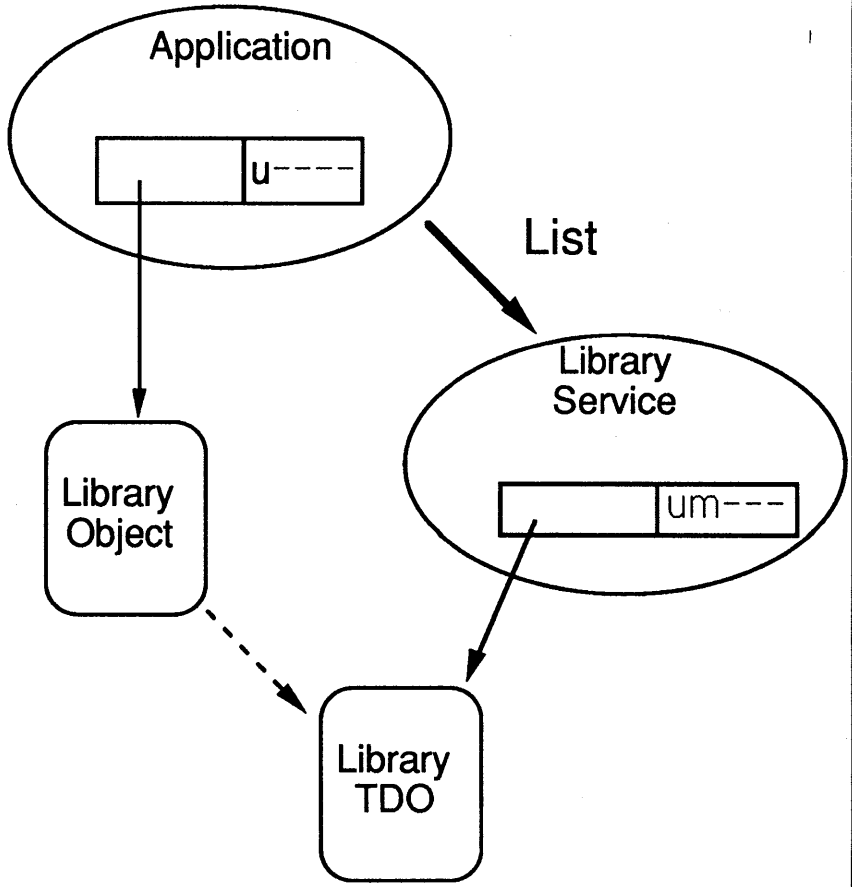
Authority-List Determines Usage Rights



BiiN™

Type Specific Operations

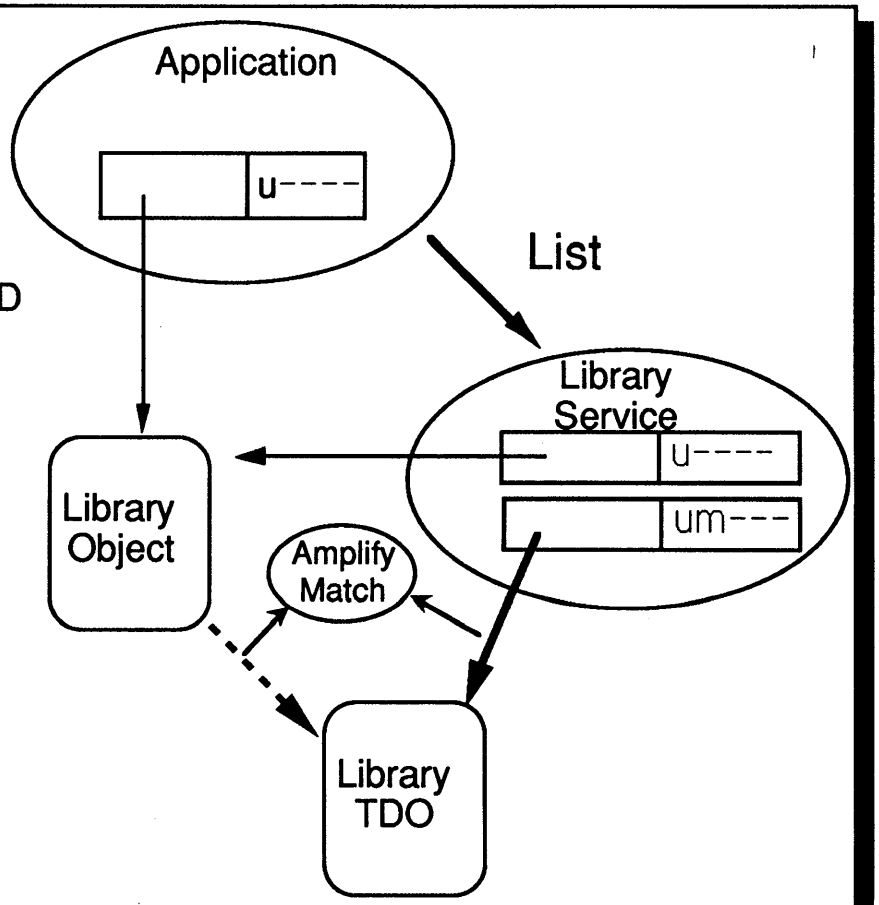
- Application has AD for Library Object with Just Use (Lookup) Rights
- Application Invokes List Subprogram Passing AD



BiiN™

Type Specific Operations (cont'd)

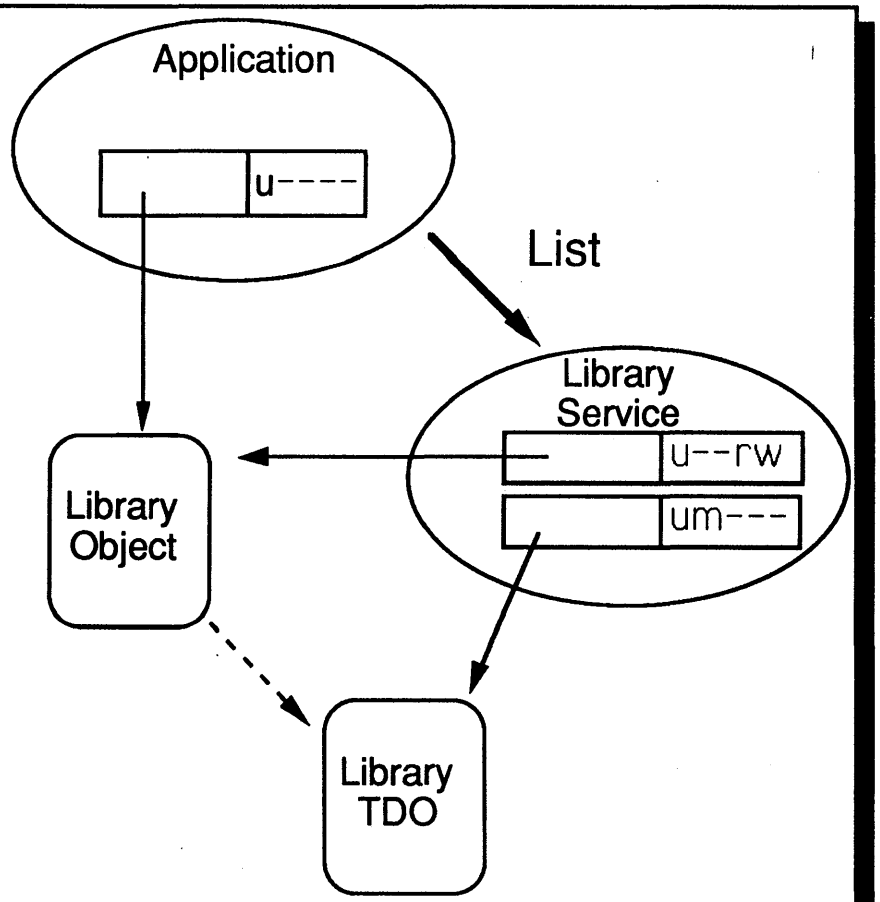
- Application has AD for Library Object with Just use (lookup) Rights
- Application Invokes List Subprogram Passing AD
- Library Service Executes Amplify Instruction
  - Takes AD to-be-amplified and AD with Amplify (modify) rights for a TDO
  - Verifies type match
  - Adds representation rights



BiiN™

Type Specific Operations (cont'd)

- Application has AD for Library Object with Just use (lookup) Rights
- Application Invokes List Subprogram Passing AD
- Library Service Executes Amplify Instruction
  - Takes AD to-be-amplified and AD with Amplify (modify) rights for a TDO
  - Verifies type match
  - Adds representation rights
- Library Service Can Now Access Representation of Library Object



BiiN™

Relationship To Security

"The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structuring of the TCB and the system. The TCB shall incorporate significant use of layering, abstraction and data hiding. "

From the Orange Book (DOD 5200.28-STD) section 3.3.3.1.1, System Architecture for B3 level security

BiiN™

Relationship to Security (cont'd)

Corresponds to SAT Type Enforcement

- **S**ecure **A**da **T**arget
- Extends Bell and LaPadula Model Beyond A1
- NCSC/Honeywell Research

"Domains are essentially a mechanism for encapsulating managers for different data types and transformations between data types. This provides a way to decompose the proof of security for the system into manageable pieces and to tailor the security policy for a system in an application dependent fashion. ... Thus, type enforcement is more than a mere convenience. It provides a way to unify the treatment of trusted subjects with that of generic untrusted subjects."

From "Extending the Noninterference Version of MLS for SAT", IEEE Transactions on Soft. Eng. Feb. 1987.

BiiN™

Topics

- Object Addressing and Protection
- Computational Model
- Type Manager Based Protection
- **Inheritance**
- Object Persistence

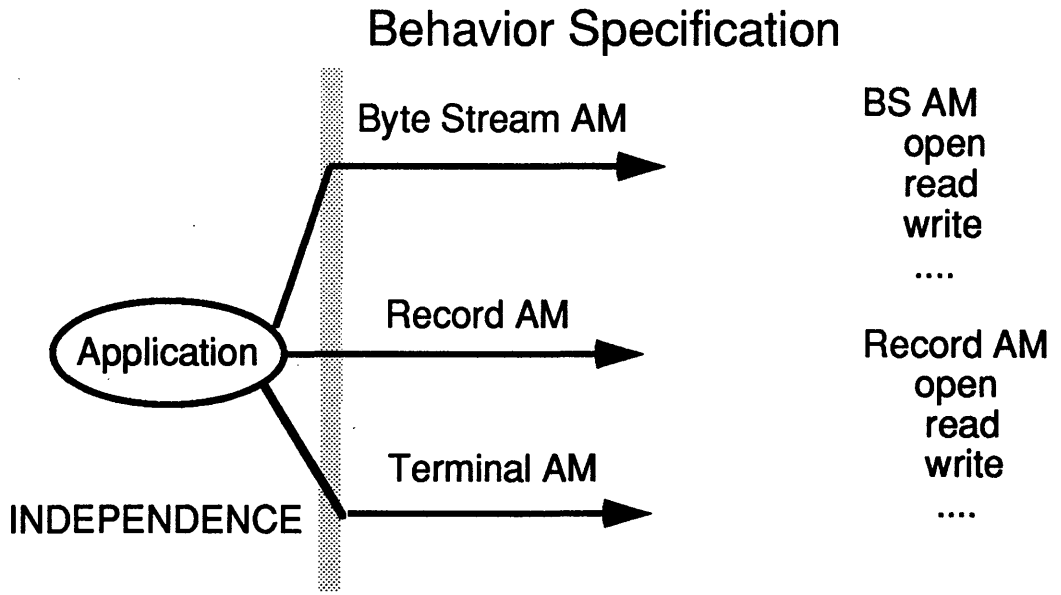
BiiN™

Behavior Inheritance

- Multiple Implementations for Same Behavior
- Example
  - *Behavior*
    - Byte-Stream Access Method
  - *Implementations*
    - File, Pipe, Magtape, Terminal
- Implementation is Selected in Call Instruction Based on Type of Object

BiiN™

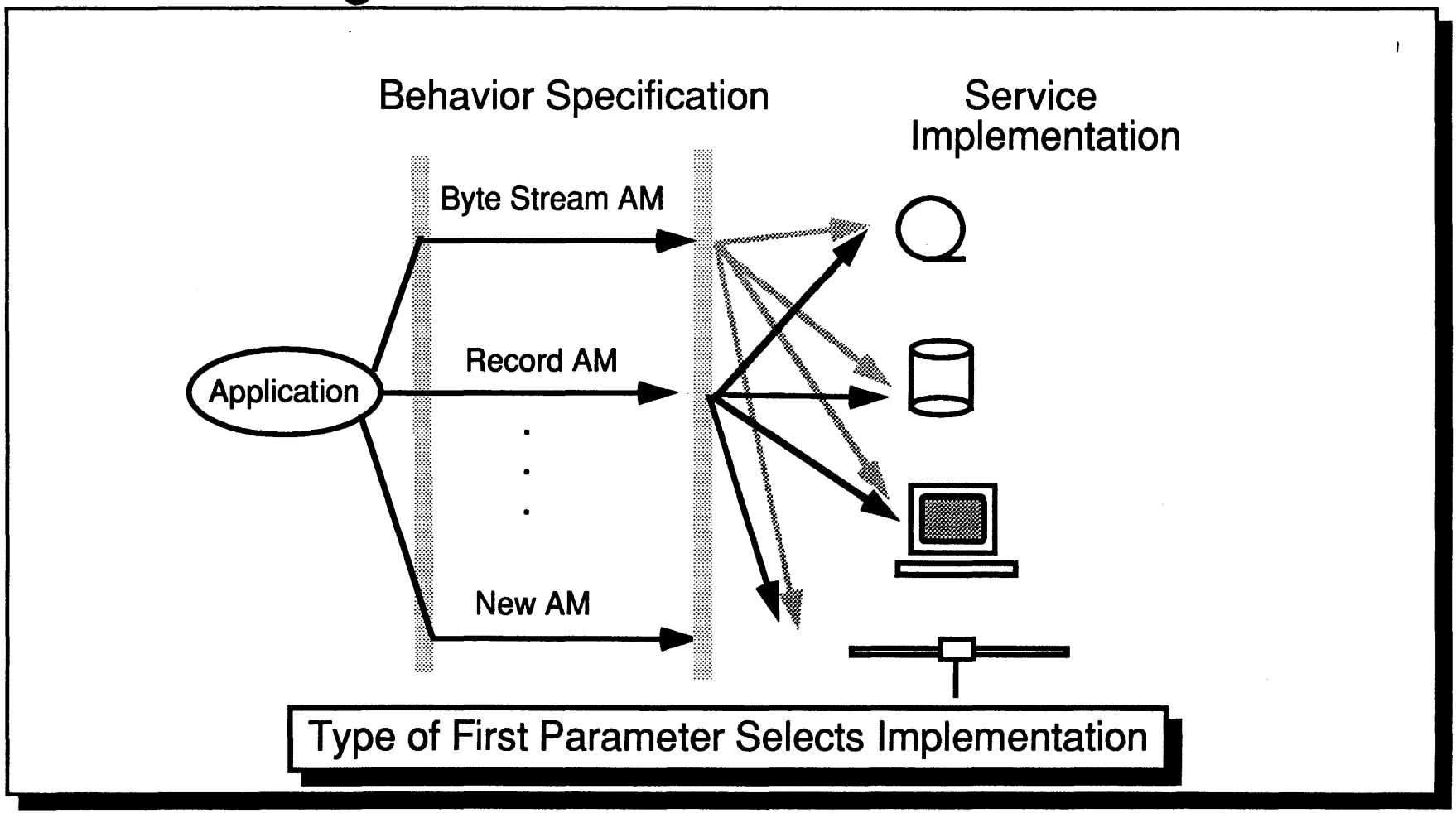
Application Independent of Implementation



- Ada Package Specification Specifies Behavior
- Provides Semantic Definition to Interface
- Services Provide Implementation (package body)
- New Behaviors can be Added

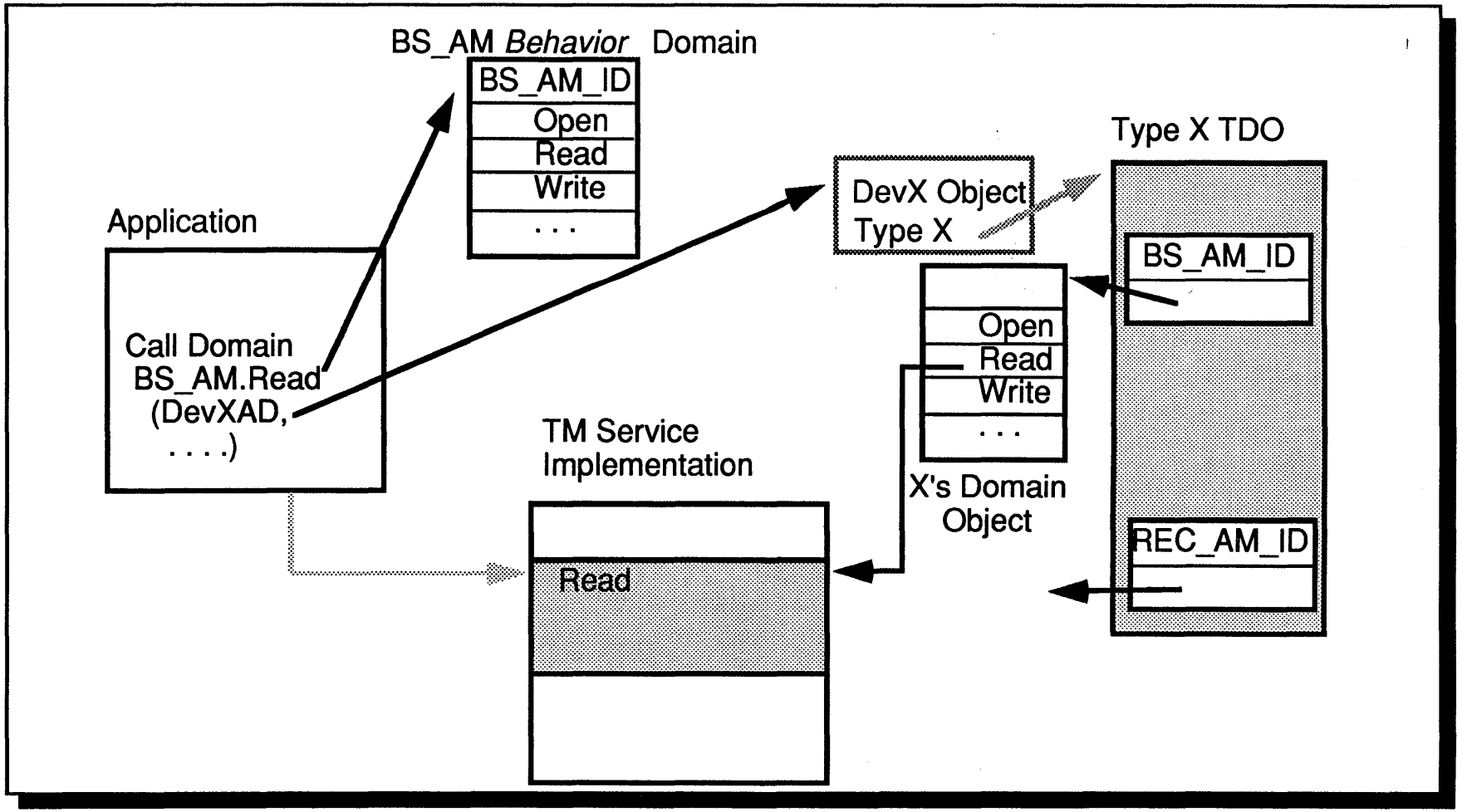
BiiN™

Call Vectoring



BiiN™

Implementation Selected by Object's Type



BiiN™

Behavior Inheritance Summary

- Dynamic Binding on Every Call Based on Type
- Old Program Binaries Work with New Implementations Without Even Relinking
- *Call Behavior* Instruction Same as Call Domain
  - Different effect due to difference in domain objects
  - Thus, invokable from any language
- Service Can Dynamically Add Implementations for New Behaviors

BiiN™

Topics

- Object Addressing and Protection
- Computational Model
- Type Manager Based Protection
- Inheritance
- **Object Persistence**

BiiN™

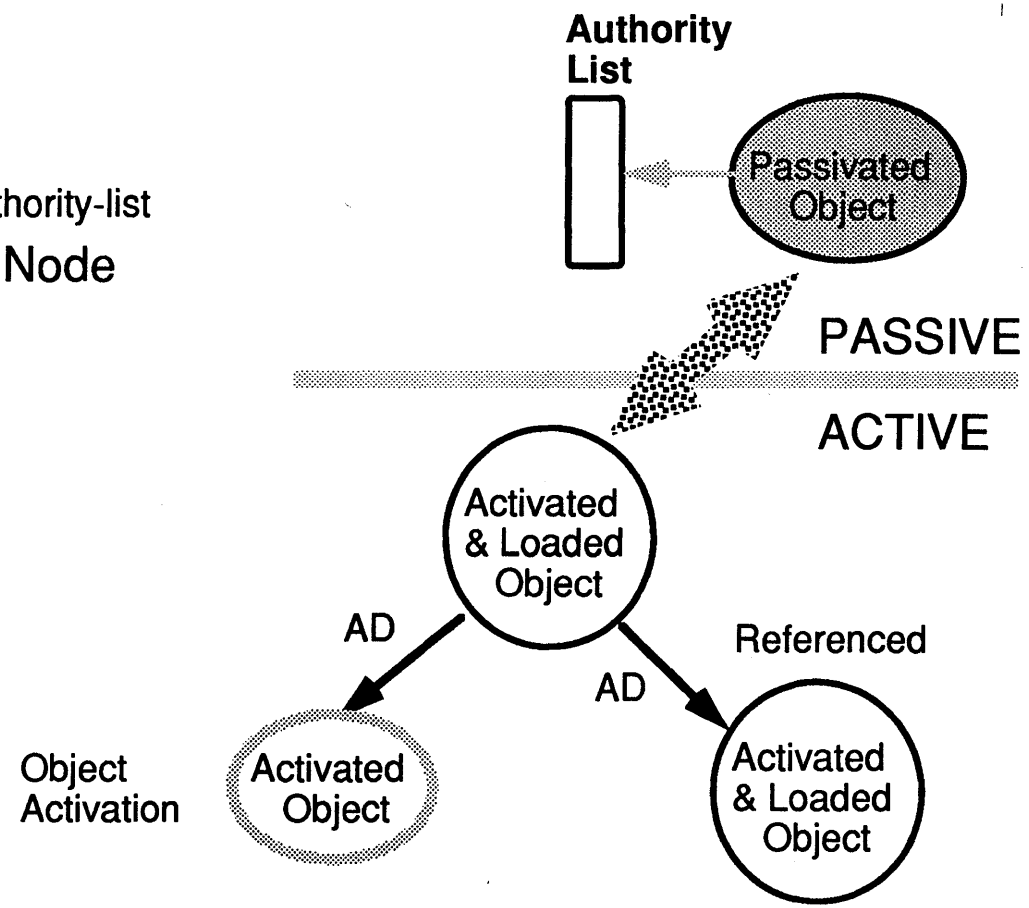
Object States

- Two object states: *Passive* and *Active*
- A Passivated Object is
  - Stored in permanent storage (disk)
  - Managed by Passive Store Management
  - Protected by Authority List
- An Activated Object is
  - Stored in virtual memory
  - Managed by object service
  - Protected by VLSI-based object addressing

BiiN™

Activation

- Implicit
  - Similar to VM fault
  - Rights in AD determined by authority-list
- Object Networks Can Cross Node Boundaries

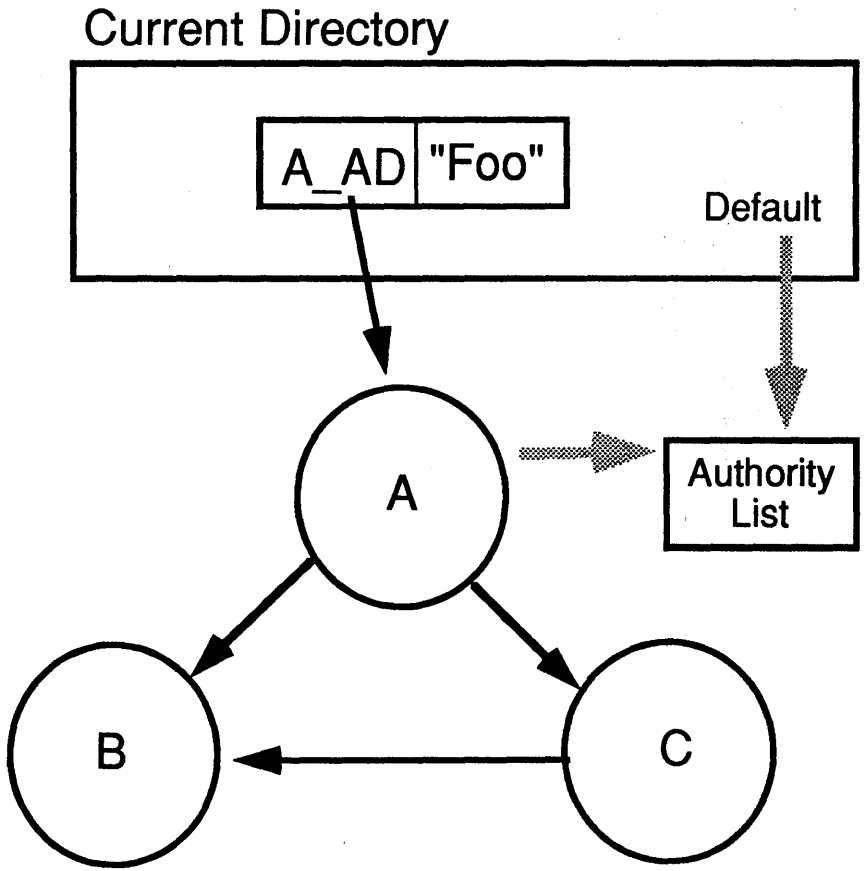


BiiN™

Passivation

- Explicit
  - Transaction-based for synchronization and recovery
  - Controlled by type manager to assure consistency

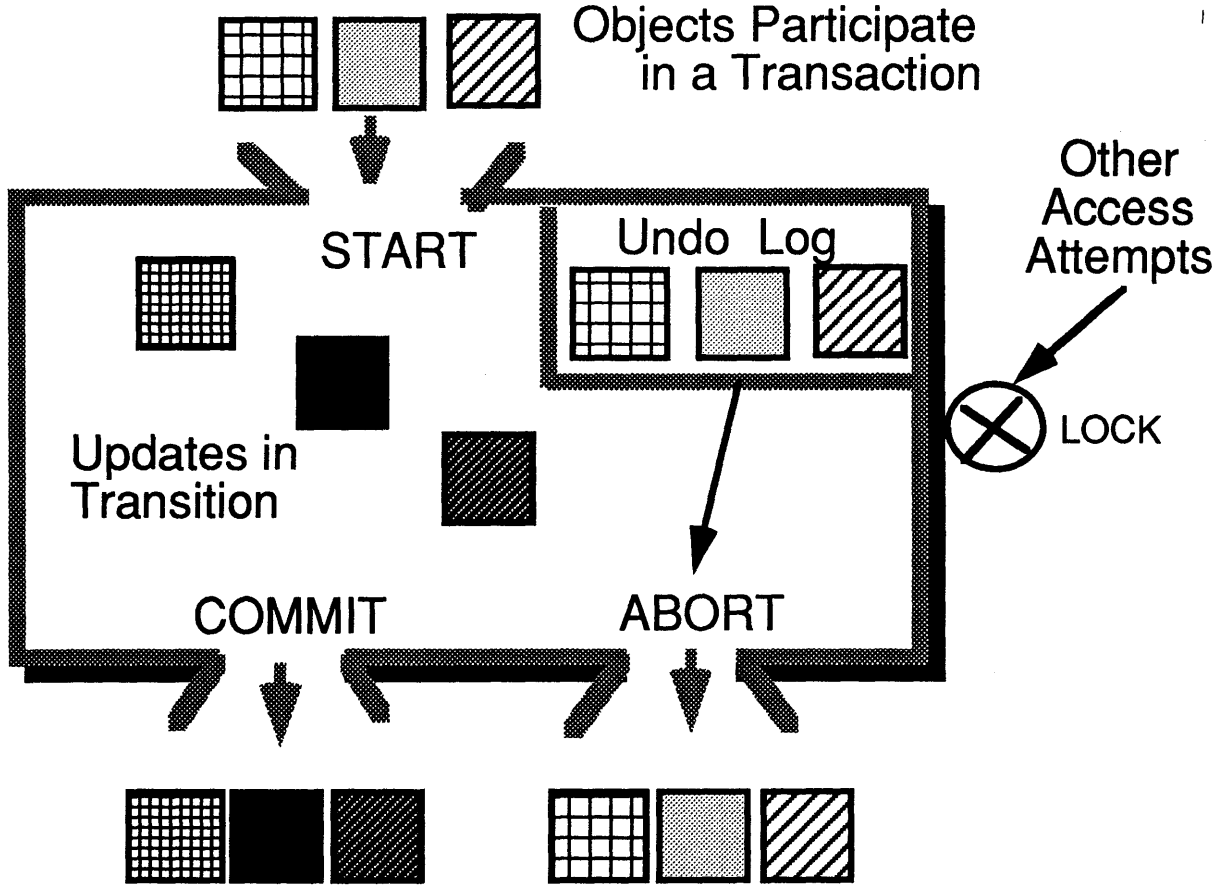
```
Start_Transaction;
        Store("Foo", A_AD);
        Update(A_AD );
        Update(B_AD );
        Update(C_AD );
Commit_Transaction;
```



BiiN™

Transaction Concepts

Properties
- Atomicity
- Synchronization
- Integrity
- Recoverability



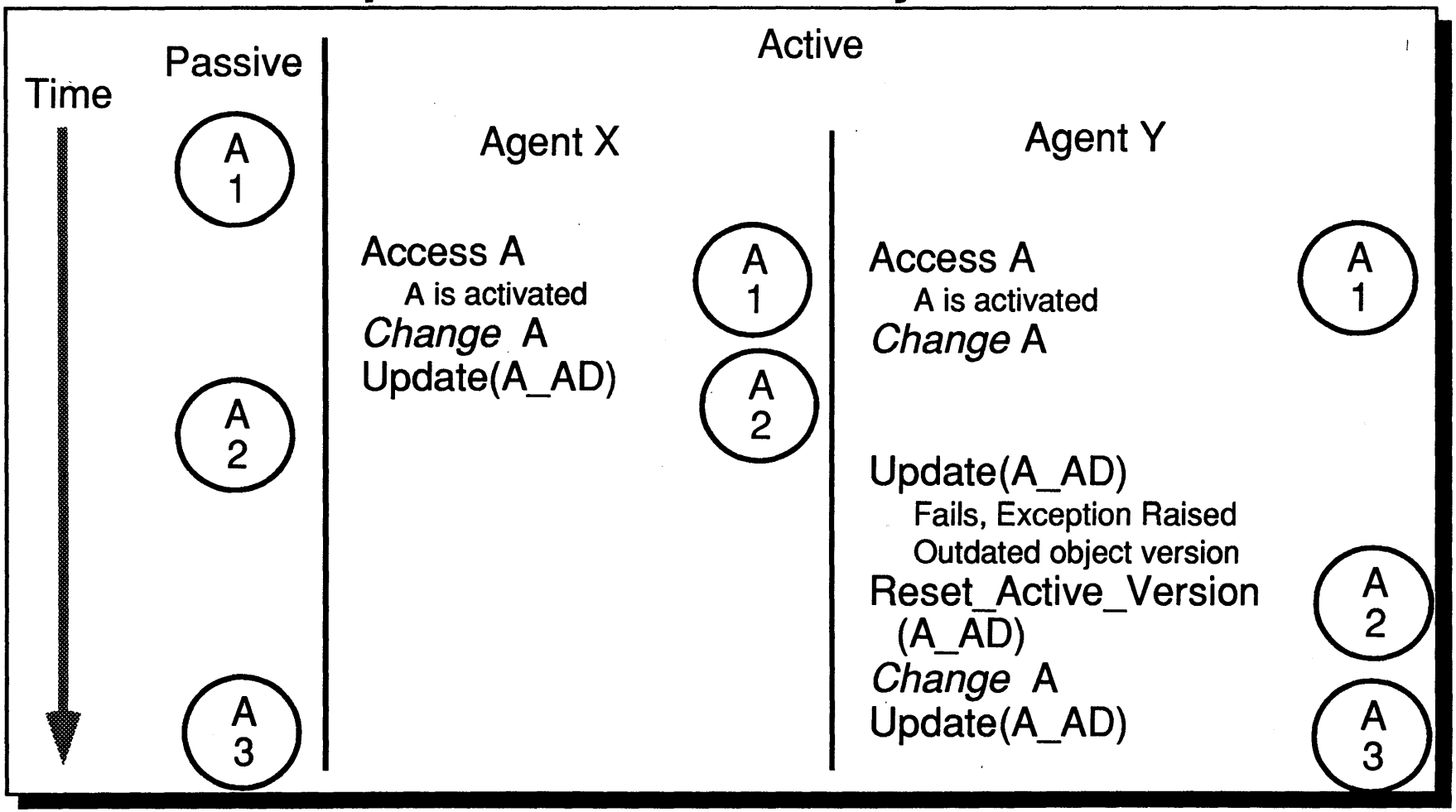
BiiN™

Transaction Service Embedded in BiiN/OS

- Transaction Service Acts as Coordinator
- Multiple Services Can Participate in Same Transaction
  - Files
  - Directories
  - Libraries
- Extendable to New Services
- Distributed
  - using 2-phase commit protocol

BiiN™

Distributed Optismitic Concurrency



BiiN™

Distributed Pessimistic Concurrency

Start_Transaction;

Reserve(A_AD);

Synchronizes. Does Reset, if necessary.

Change A

Update(A_AD);

Commit_Transaction;

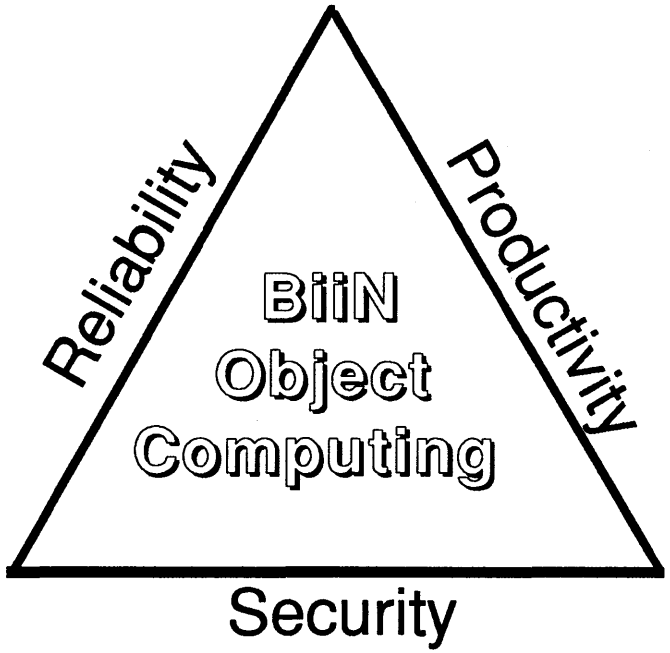
BiiN™

Persistent Object Summary

- Supports a Permanent Network of Distributed Typed Objects
  - Network can cross disk and node boundaries
  - Accessible independent of location
- Supports Concurrent Distributed Access
  - Based on transactions for synchronization and data integrity
  - Both optimistic and pessimistic synchronization are supported
- Activation is Implicit for Ease-of-use
- Passivation is Explicit for Data Integrity

BiiN™

Summary



BiiN™