

# GRID FRAMEWORKS

Ajit Kunjal

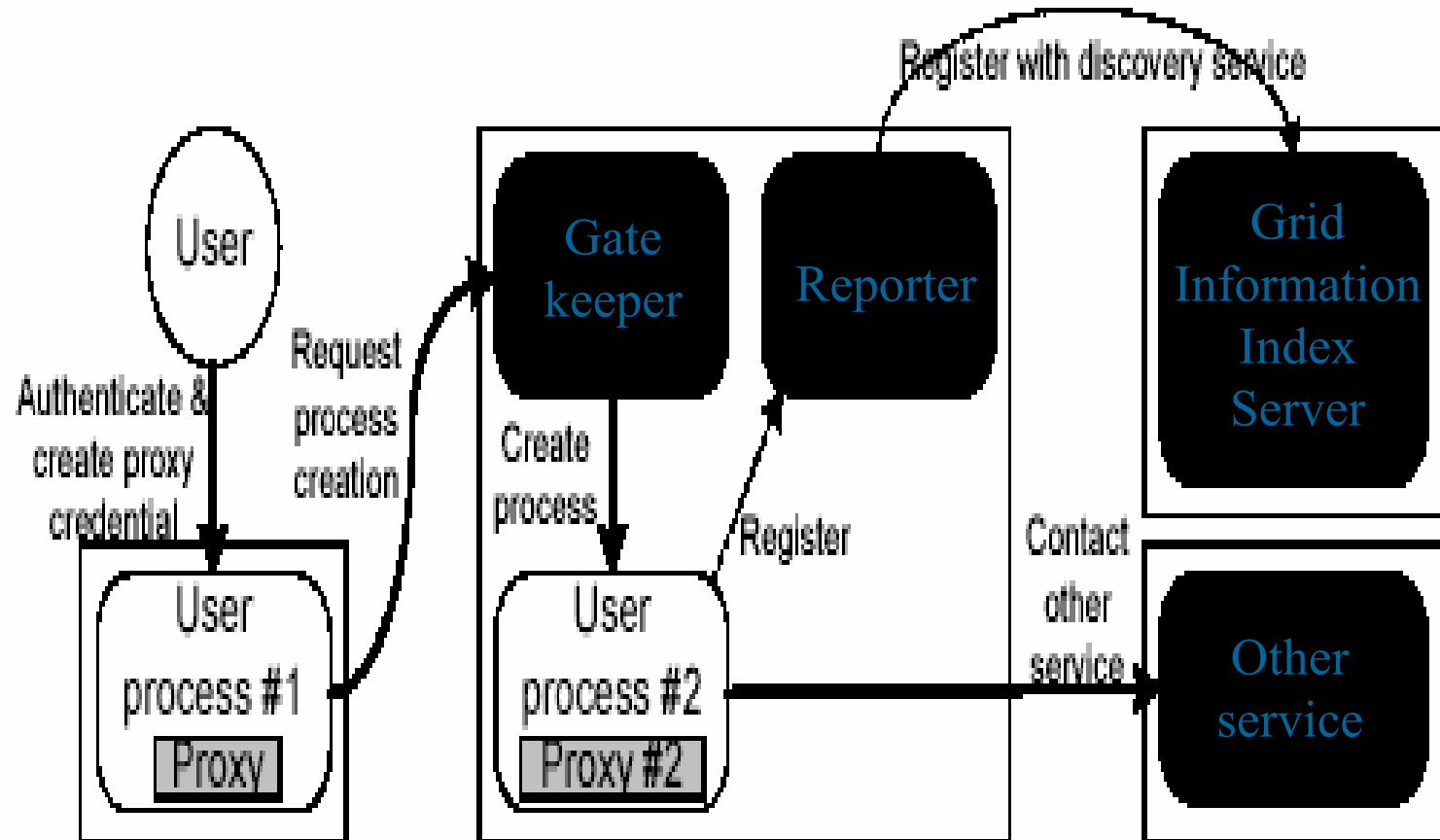
# What we will be looking at today ...

- Globus
- PUNCH
- Legion
- Condor
- An interoperable computational  
collaboratory - DISCOVER

# GLOBUS

- termed as a Metacomputing Infrastructure toolkit
- a community-based, open-architecture, open-source set of services and software libraries that supports Grid and Grid applications

# Selected Globus Toolkit Mechanisms



# Metacomputing ...

- Metacomputers have much in common with parallel and distributed systems yet they also differ greatly
- Long term goal of Globus is to address the problems of configuration and performance optimization in metacomputing environments

# The Globus toolkit ...

Higher  
level  
services

Globus services: AWARE

MPI, CC++,  
CAVEcomm

I-Soft  
scheduler

...

Other services (actual/planned)

Legion

AppLeS

...

CORBA

HPC++

---

*Globus metacomputing abstract machine*

---

Globus  
toolkit  
modules

Comms  
(Nexus)

Resource  
(al)location

Authent-  
ication

Information  
service

Data access  
(RIO)

...

---

*Heterogeneous, geographically distributed devices and networks*

---

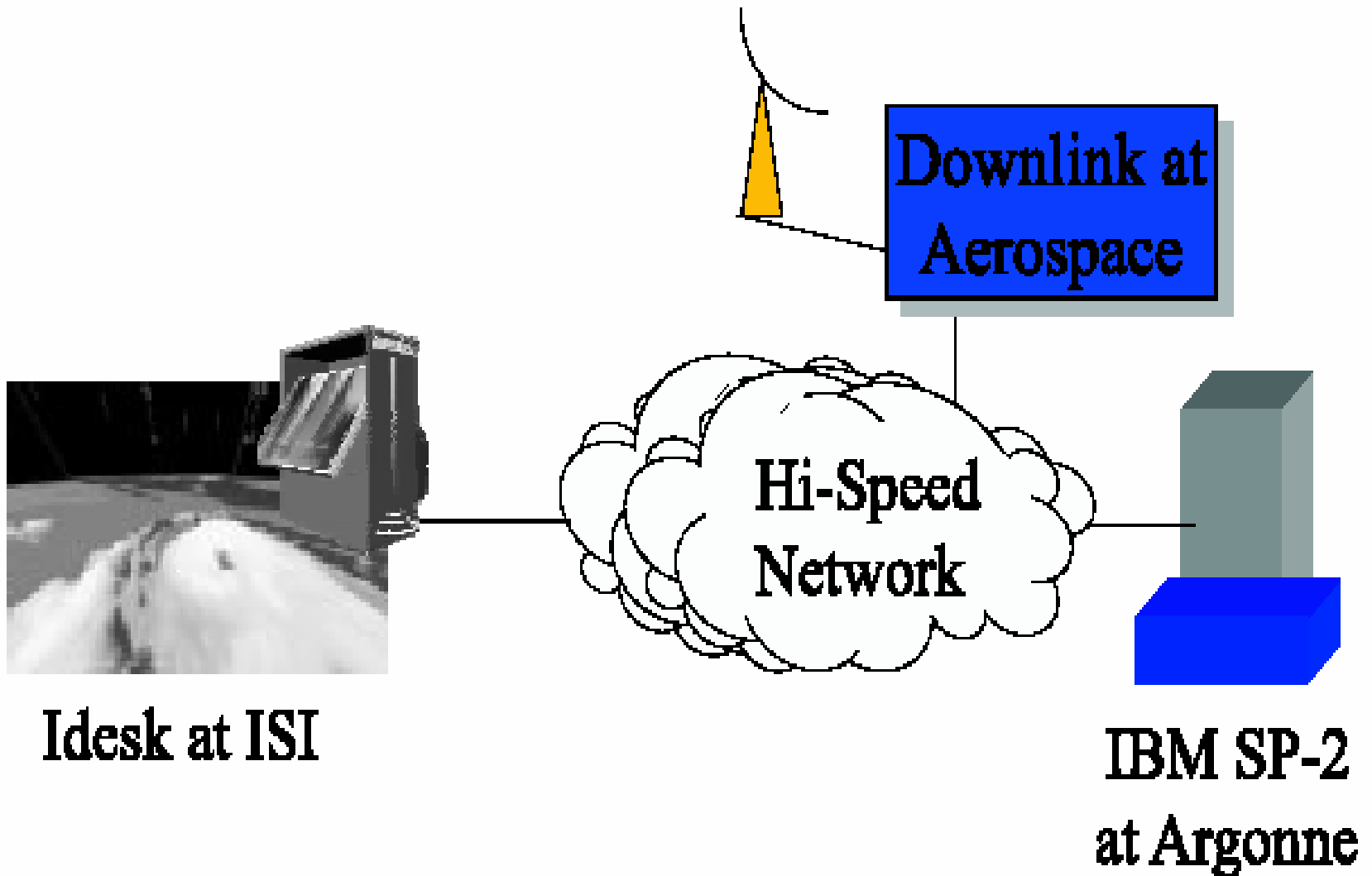
Meta  
computing  
testbeds

I - W A Y

G U S T O

...

# The I-WAY Setup ...



# Globus Infrastructure Toolkit ...

- The focus in the Globus project is to
  - Develop low level mechanisms that can be used to implement higher level services
  - Techniques that allow these services to observe and guide the operation of these mechanisms.

# Globus Infrastructure Toolkit ...

- The currently identified modules are
  - Resource location and allocation
  - Communications
  - Unified resource information service
  - Authentication interface
  - Process creation
  - Data access

# Globus Infrastructure Toolkit ...

- Globus toolkit modules address the problem of support for resource-aware services and applications by providing interfaces for
  - Rule based selection
  - Resource property inquiry
  - Notification

# Globus Toolkit Components ...

- The components of the Globus toolkit are
  - Communications
  - Metacomputing directory service
  - Authentication methods
  - Data access services

# Communications Module

- Based on the Nexus communication library
- Nexus defines 5 basic abstractions
  - Nodes, Contexts, Threads, Communication links and Remote service requests.

# Metacomputing Directory Service

- Is used to provide information such as
  - Configuration details about resources
  - Instantaneous performance information
  - Application specific information
- Built upon the LDAP framework where the entries are stored as attribute-value pairs

# Authentication Methods

- Globus is moving towards implementing the Generic Security System (GSS) to increase the degree of abstraction at the toolkit interface

# Data Access Services

- Services that provide access to persistent data can face stringent performance requirements and must support access to data in different administrative domains
- The Globus data access module defines primitives that provide remote access to parallel file systems.

# Globus - Summary

- We list three areas in which Globus has made contributions;
  - Definition of a core metacomputing system architecture
  - Development of a framework that allows applications to dynamically adapt their behavior
  - Demonstration in testbeds that useful higher level services can be layered on top of the toolkit interfaces

# PUNCH

- It is an architecture for a universally accessible, web-enabled, wide area network computing
- The defining feature here is that the system can make automatic cost/performance tradeoff decisions at runtime
- It is a demand-based network computing system that allows users to access and run unmodified tools via WWW browsers

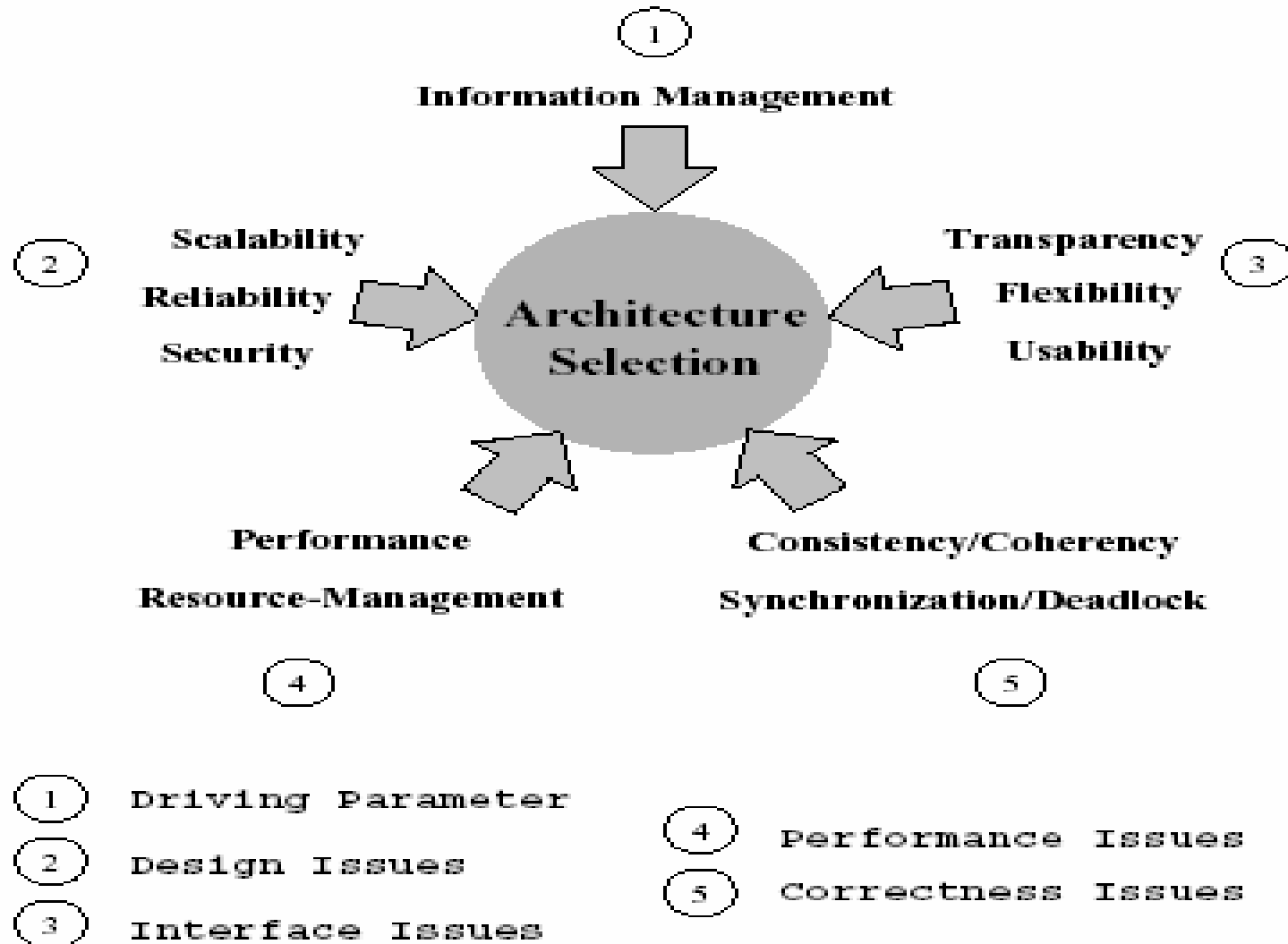
# PUNCH - Goal

- Design and deploy a production wide-area computing framework in which the computing framework is not tied to the characteristics of the individual applications
- Functionally, this is equivalent to designing a multi-user operating system for networked resources

# PUNCH – Design Issues

- The issues that drive the design of a demand-based network computing system are
  - Information management
  - System architecture
  - Scalability and Reliability
  - Security and Access control
  - Interface and Performance issues

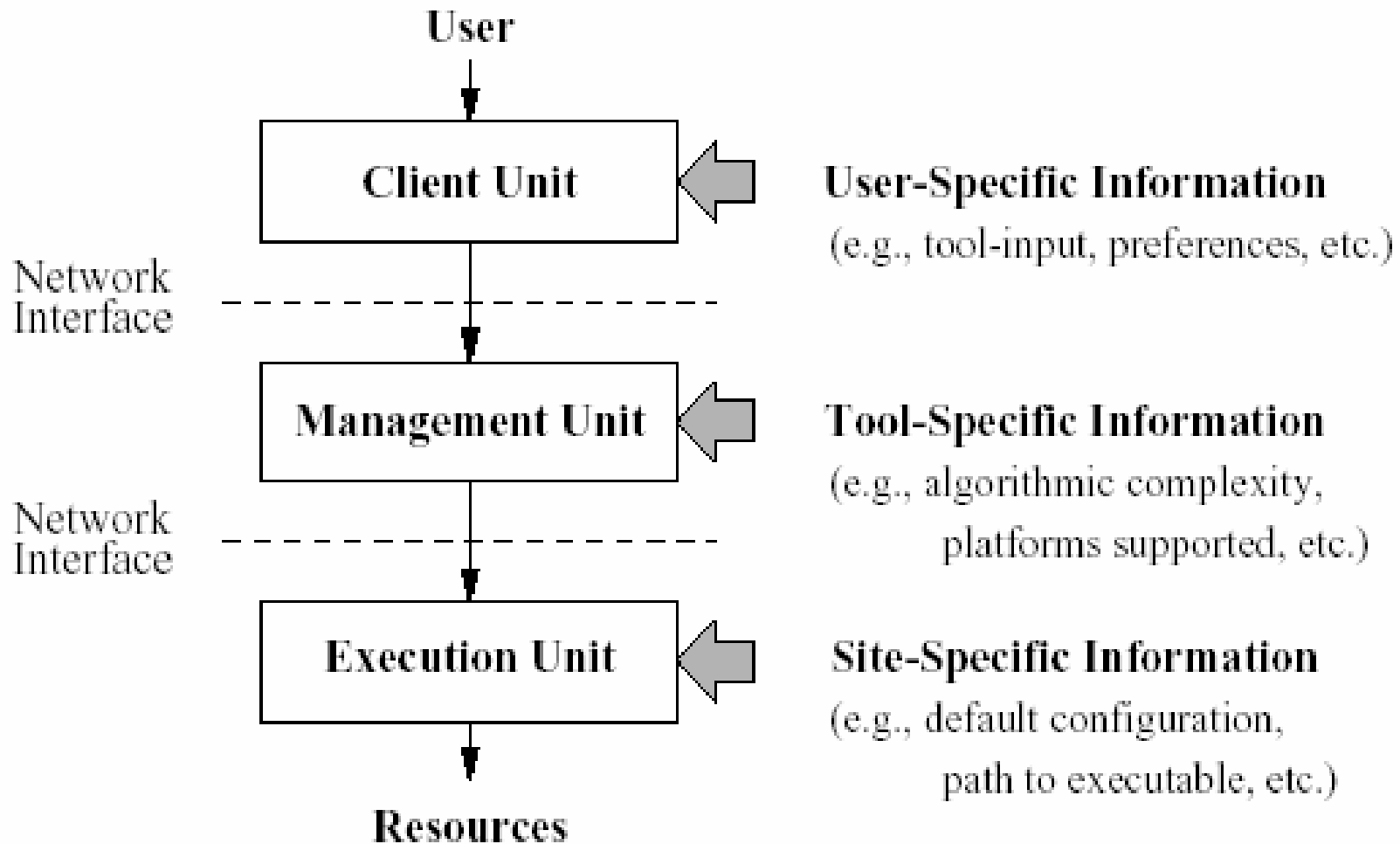
# Architecture selection process ...



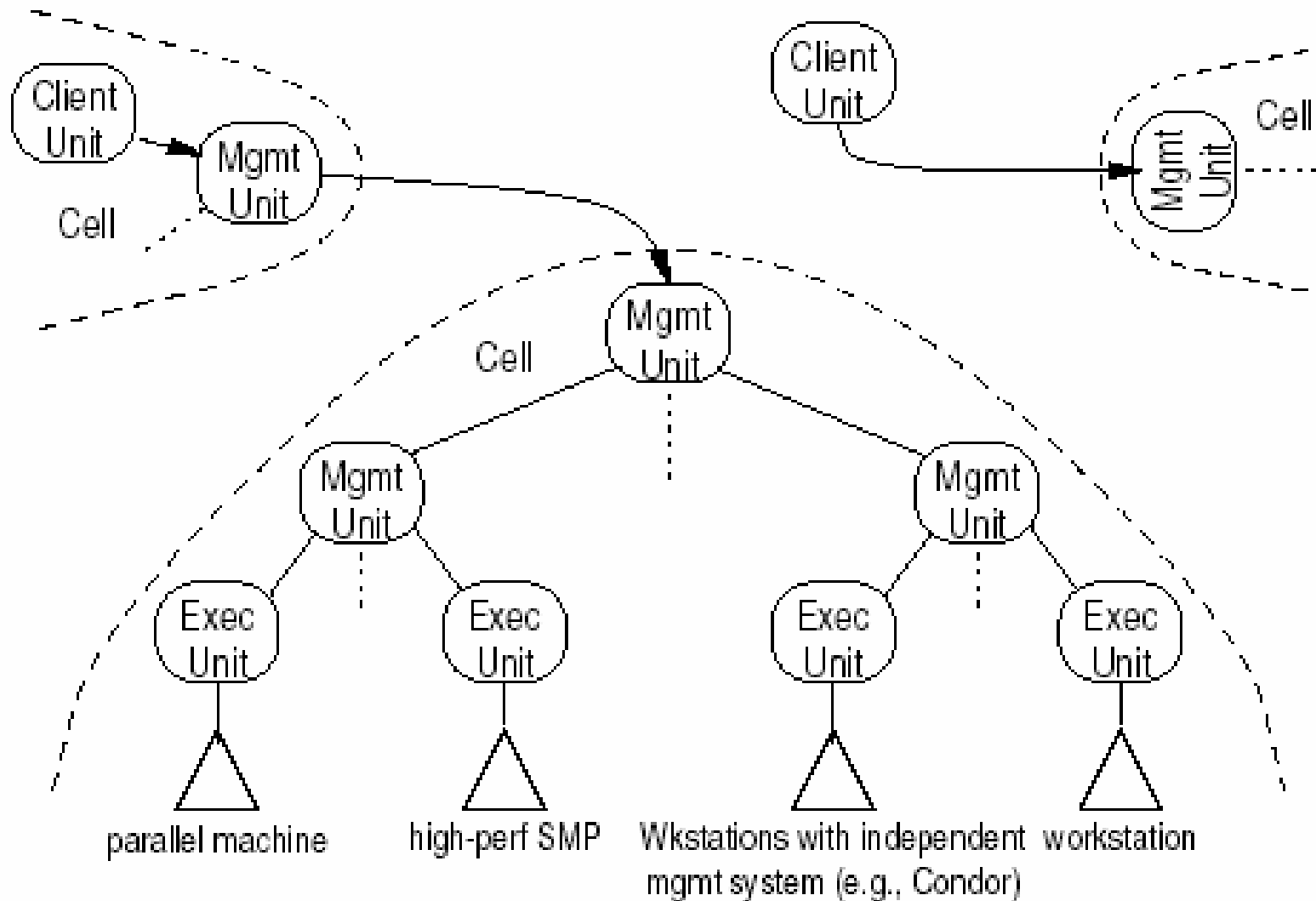
# Information Management ...

- The tool specific C/P tradeoff decisions are based on
  - The portability of the given tool
  - Its run specific resource-usage characteristics
  - Site specific policies
- The dynamic, incremental and distributed nature of this information makes it extremely difficult to collect and maintain

# System Architecture ...



# System Architecture ...



# Scalability and Reliability

- The scalability of a network-computing system can be defined in terms of
  - Users
  - Software resources
  - Hardware resources
  - Administrative domains

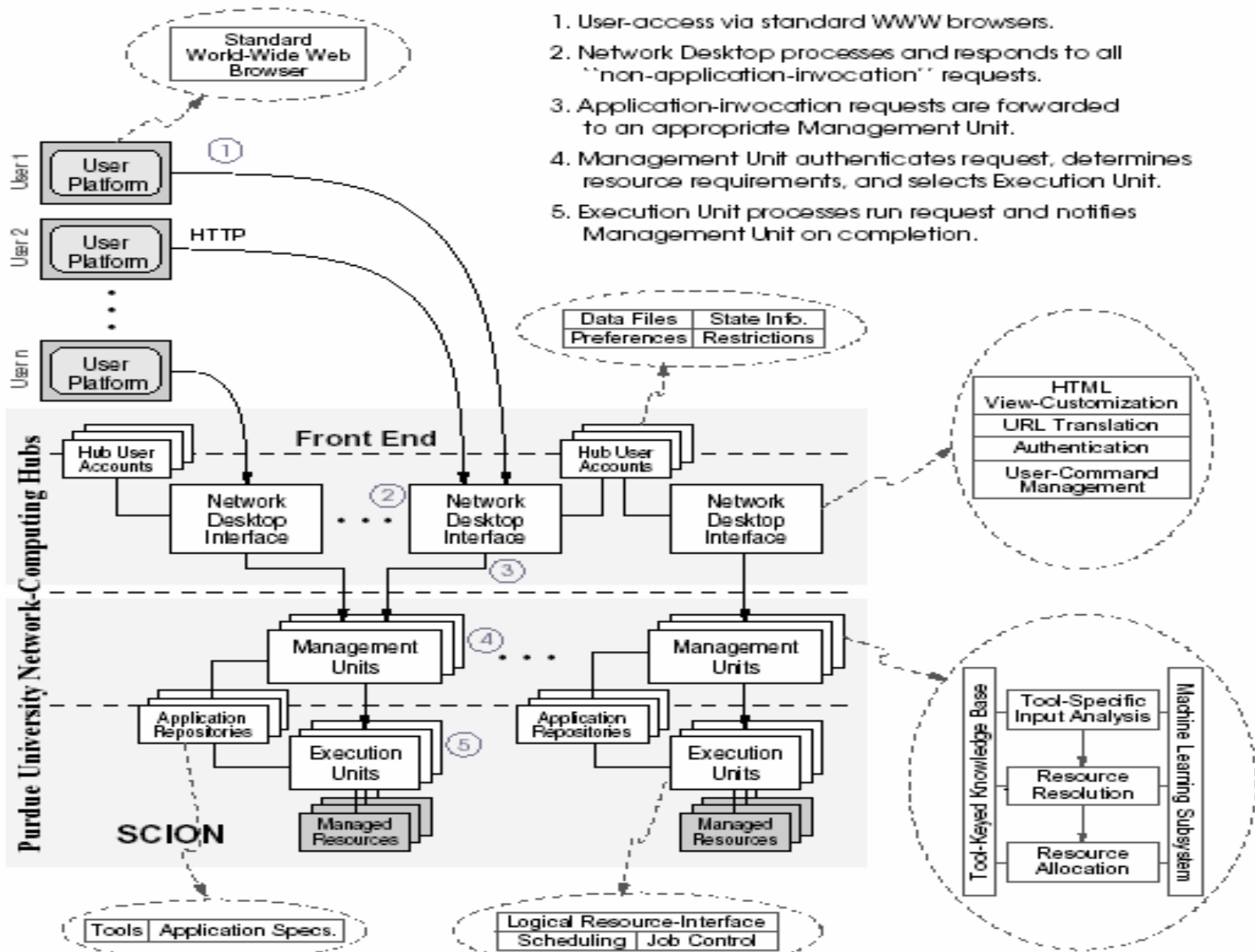
## Security and Access control ...

- Infrastructure that spans multiple domains must allow each one to independently control its policies
- Security issues are managed by partitioning the infrastructure into independent cells
- Execution units accept requests only from the local management unit

# Interface and Performance Issues ...

- The interface should work within the framework of the existing web infrastructure and at the same time should be flexible enough to accommodate diverse needs
- Performance metrics considered here are
  - Reduced job execution times
  - Improved resource utilization

# PUNCH Infrastructure ...



1. User-access via standard WWW browsers.
2. Network Desktop processes and responds to all "non-application-invocation" requests.
3. Application-invocation requests are forwarded to an appropriate Management Unit.
4. Management Unit authenticates request, determines resource requirements, and selects Execution Unit.
5. Execution Unit processes run request and notifies Management Unit on completion.

# Legion

- Is a metasytem software project whose goal is a highly usable, efficient and scalable system
- Aims to provide a single coherent virtual machine that addresses issues like scalability, programming ease, fault tolerance, security and site autonomy

# Design Objectives ...

- Site autonomy
- Extensible core
- Scalable architecture
- Seamless computational environment
- Parallelism
- Single namespace
- Security
- Managing heterogeneity
- Interoperability
- Fault tolerance

# Legion

- The common framework enabling a solution is object orientation.
- In Legion, all components are objects and all objects are instances of defined classes
- Hence users, data, applications, etc are all objects

# Legion System Philosophy

- Legion is designed to allow users and class implementers the greatest flexibility in their application semantics
- Users should be able to select both the kind and level of functionality, making their own tradeoffs between function and cost

# Legion prototype ...

- The first prototype Legion implementation was released in the summer of 1995 – the Campus Wide Virtual Computer (CWVC)
- It consists of more than 100 workstations and an 18 processor IBM SP2 parallel computing supercomputer and uses two completely disjoint underlying file systems

# Condor

- It is a scheduling system that aims to maximize the utilization of workstations by identifying idle ones and scheduling background jobs on them.
- The system guarantees that the job will be completed and very little, if any, work will be performed more than once.

## Condor – Motivation ...

- An analysis of workstation usage patterns revealed that only about 30% of their capacity was utilized.
- Available intervals were also very long
- These factors make workstations ideal candidates to serve as a source of remote processing cycles.

## Condor – Motivation ...

- Research in the area of algorithms for idle workstation management showed that some users try to acquire all the capacity available while others require resources occasionally
- The UP-DOWN algorithm by Mutka and Livny was designed for fair access to remote capacity for light users

# Remote Job Execution ...

- Condor uses the Remote Unix (RU) facility to execute remote jobs
- This is suited for background jobs that are computationally intensive and run for long periods without user intervention
- A unique feature of the RU facility is “checkpointing”

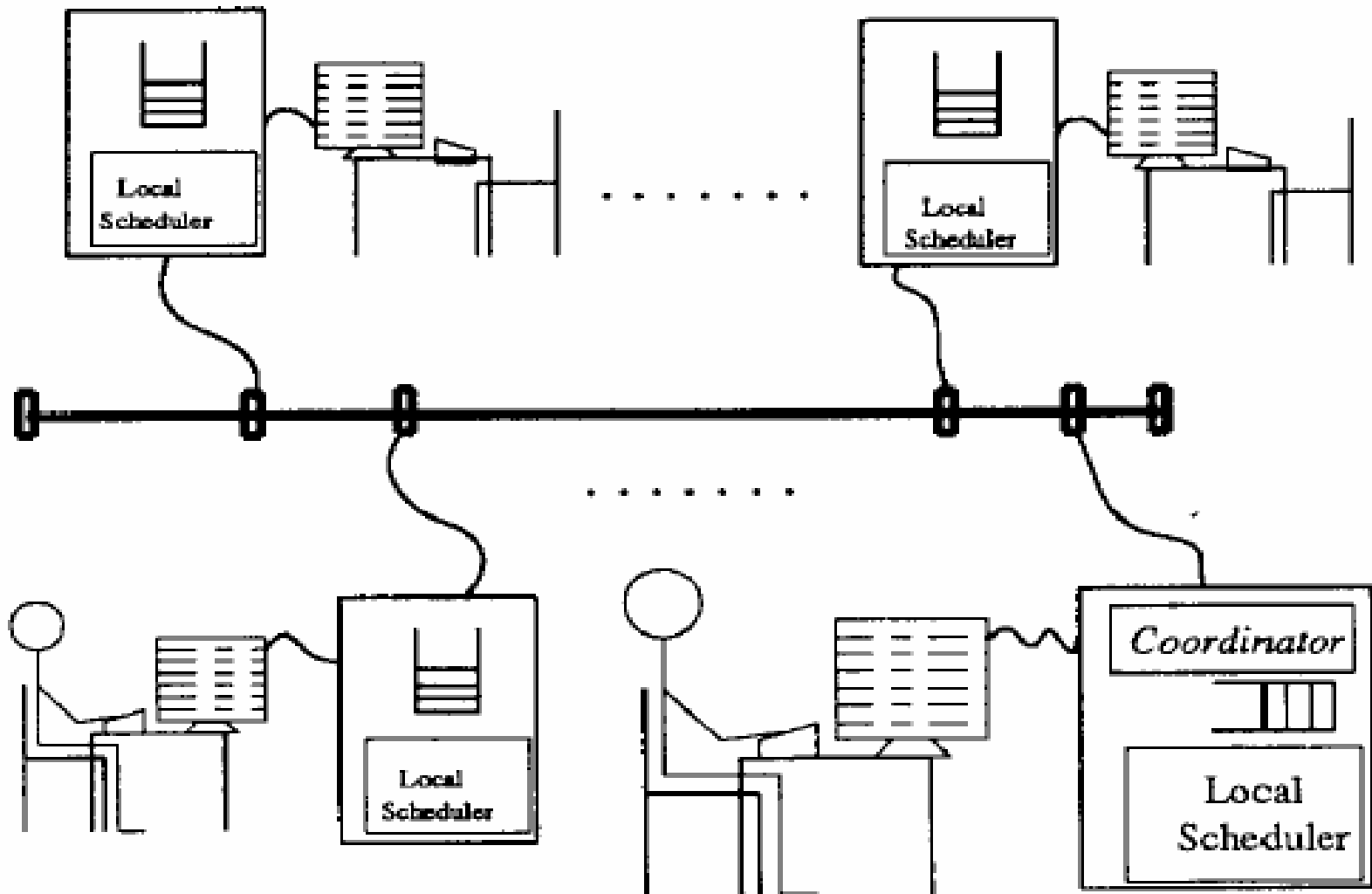
# System Design

- The design issues involved are
  - Scheduling structure
  - The Remote Unix facility
  - Checkpointing
  - Fair access to remote cycles

# Scheduling Structure ...

- The remote job scheduling structure should be transparent to the user
- The Condor job scheduler adopts an approach that lies between centralized static and fully distributed
- The principle here is that workstations are autonomous computing resources and they should be managed by their own users

# Scheduling Structure ...



# Remote Unix Facility

- It turns idle workstations into cycle servers
- When explicitly invoked, a “shadow” process runs locally as the surrogate of the process running on the remote machine
- System calls made by the remote program communicates with the shadow process
- Interrupted jobs can restart from their intermediate state due to the checkpointing feature

# Checkpointing

- It is the saving of the state of a program so that its execution can be restarted
- The state consists of text, data, stack, etc
- When a job is removed from a remote location, RU checkpoints it

# Fair access to remote cycles ...

- Light and heavy users
- Unless their workload is taken into account, heavy users may inhibit light users access
- Available capacity is managed using the Up-Down algorithm which maintains a schedule index for each workstation

# Performance ...

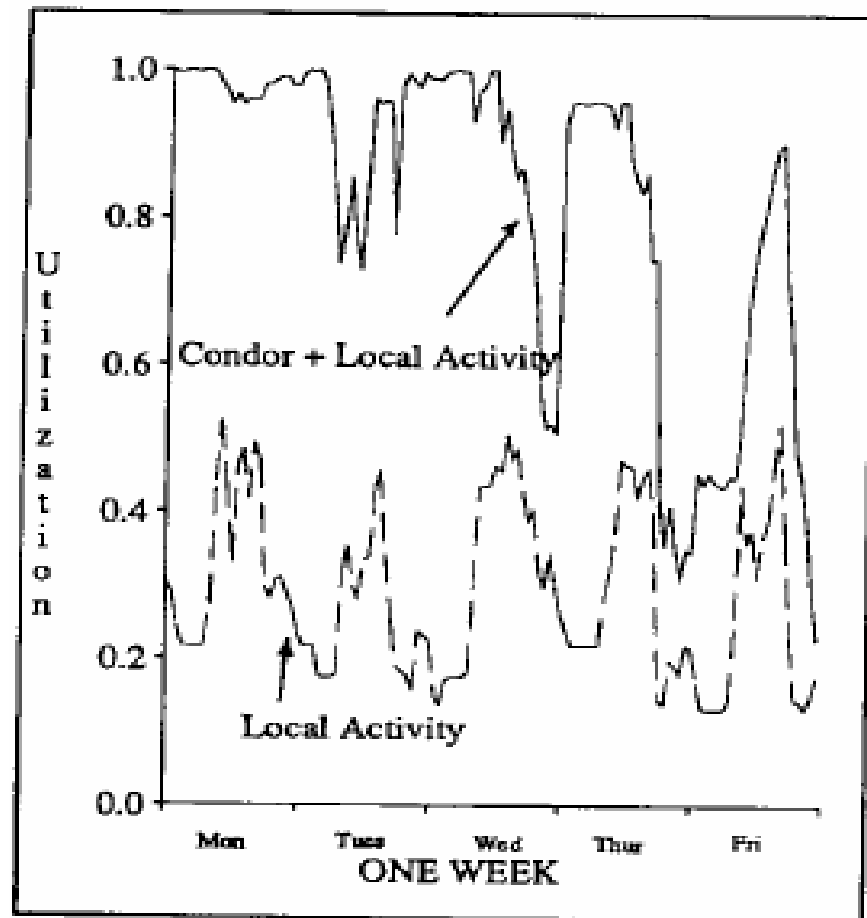
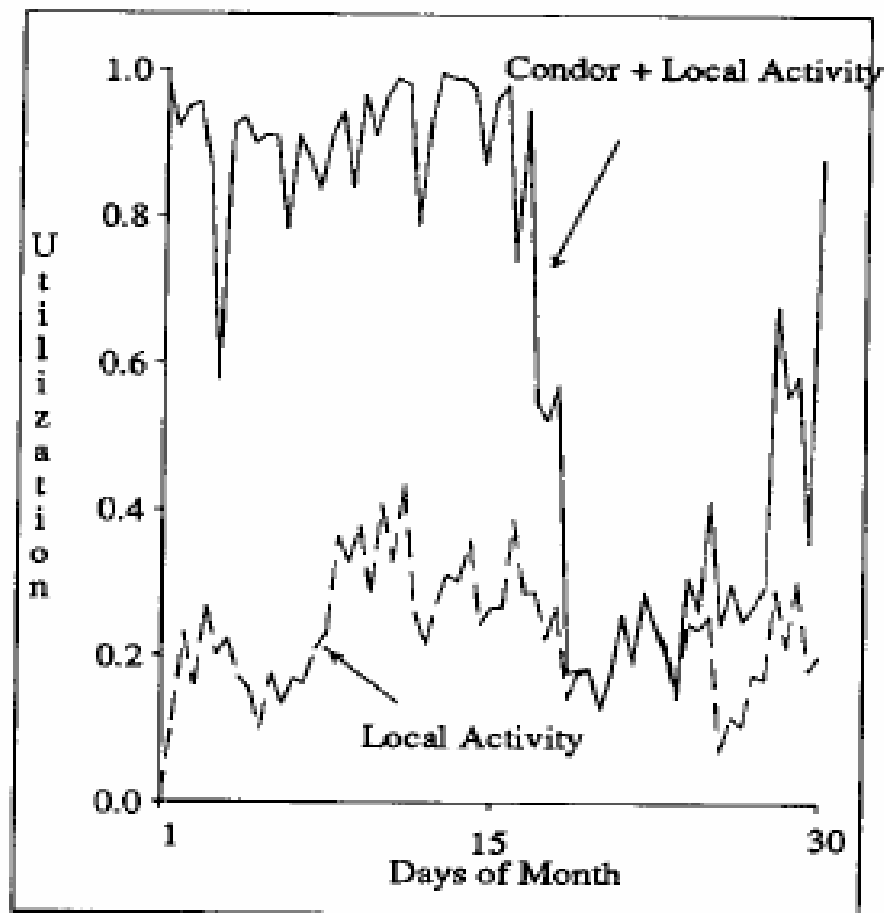


Figure 5: Utilization of Remote Resources.

Figure 6: Utilization for One Week.

# DISCOVER

- Is an interoperable computational laboratory
- Combining these laboratories and allowing them to interoperate can lead to truly collaborative, multidisciplinary problem solving

# Discover ...

- It provides collaborative access to high performance parallel and distributed applications for interaction and steering using web-based portals
- The middleware substrate enables DISCOVER interaction and steering servers to dynamically discover one another and form a peer-to-peer network

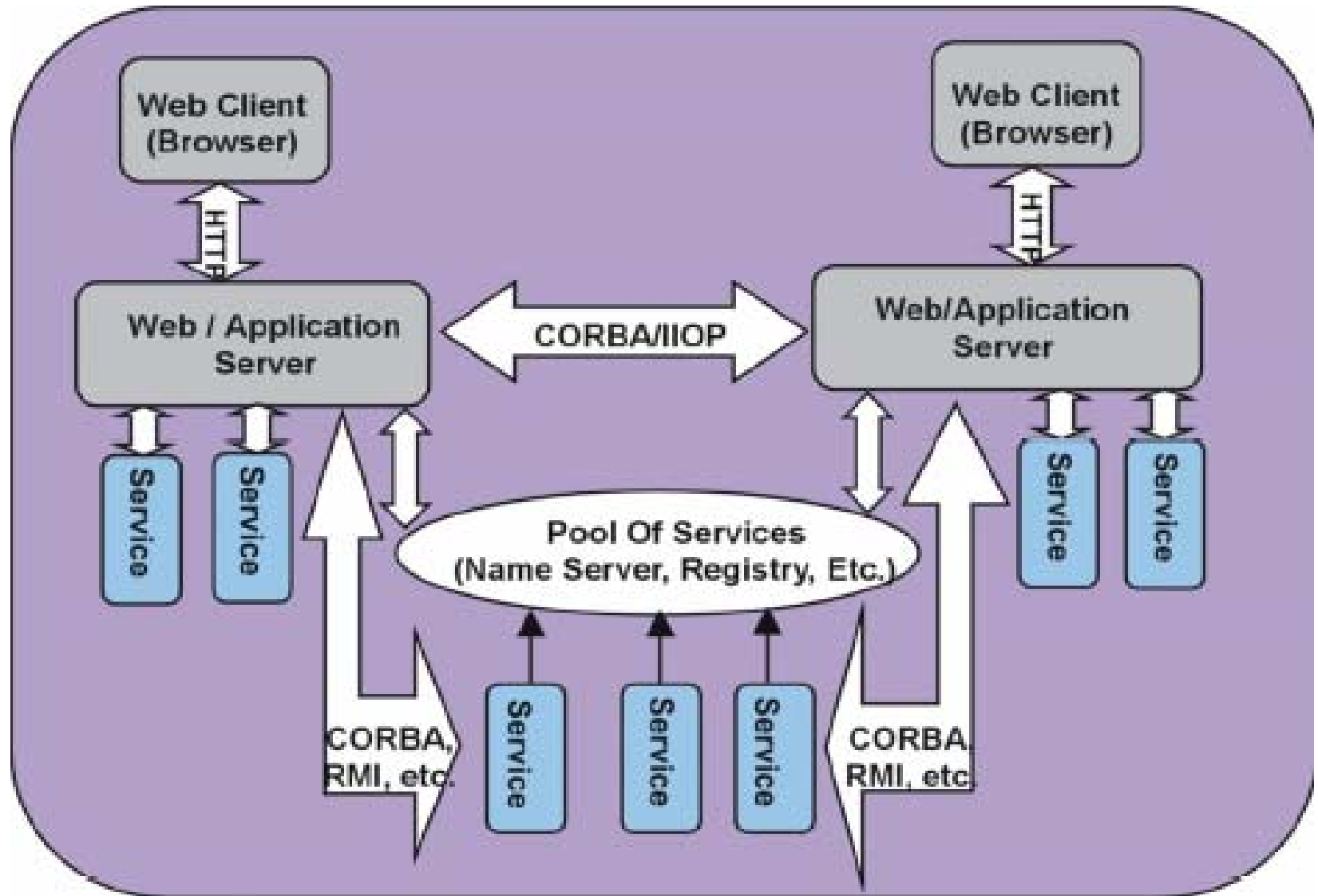
# Approaches to Interoperability ...

- There are three ways to implement interoperable higher-level services
  - Shared implementation
  - Shared interfaces and APIs
  - Shared protocols

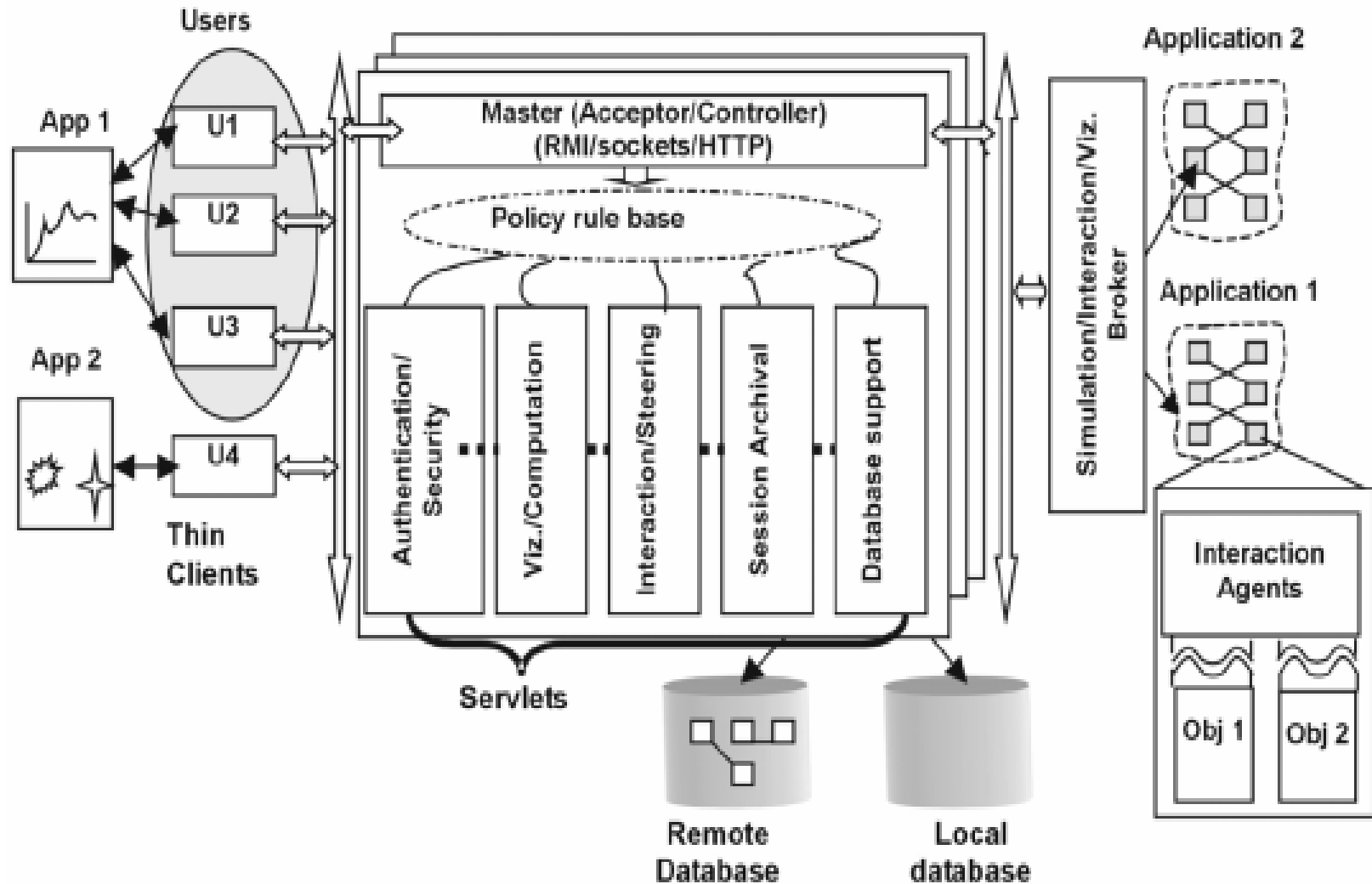
## Middleware Substrate ...

- The overall goal is to define interfaces and mechanisms for a peer-to-peer integration and interoperation of the services provided by domain specific laboratories
- It builds on existing Web servers and leverages commodity technologies and protocols
- Interoperability is got by sharing interfaces defined in the CORBA IDL

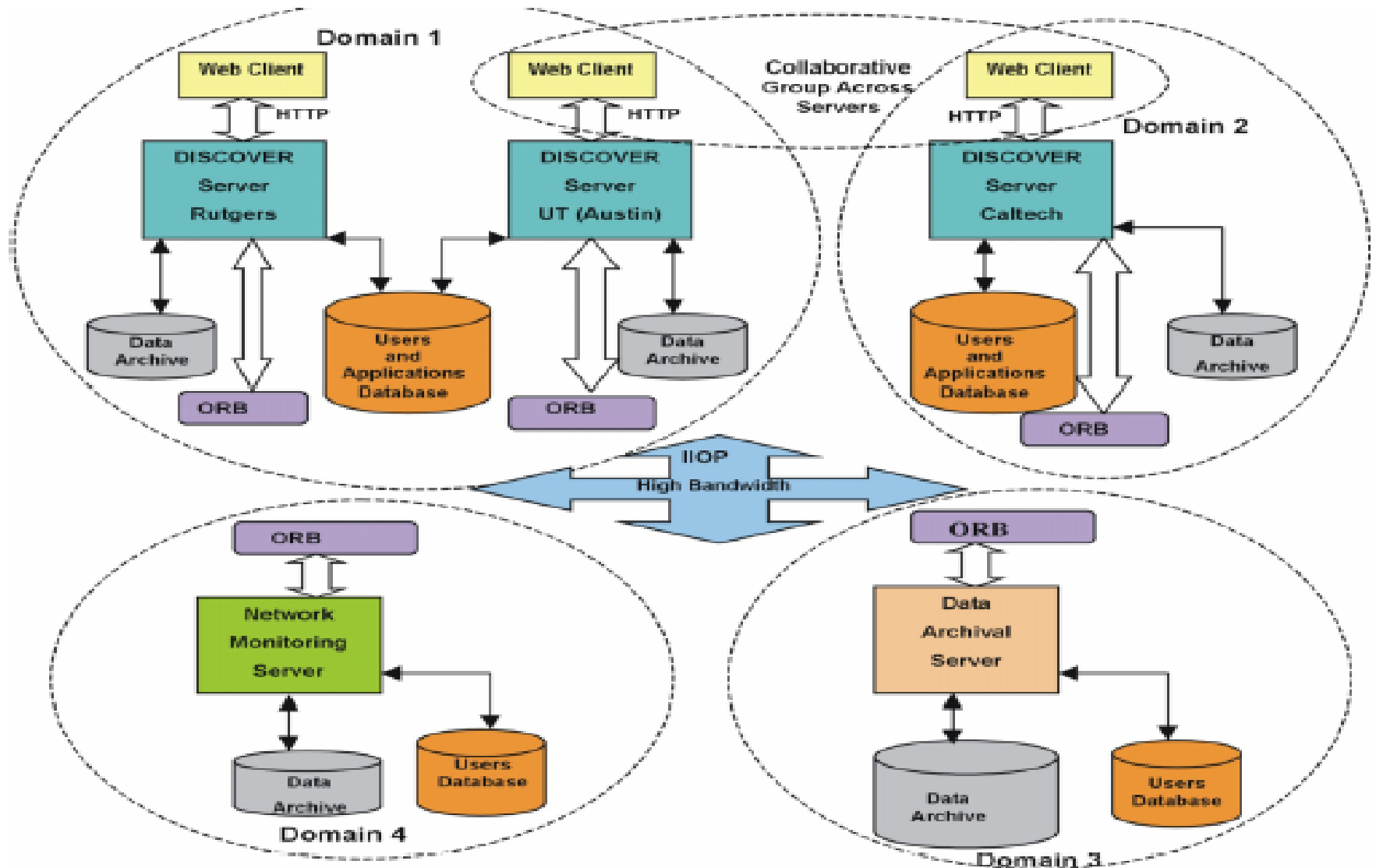
# Middleware Design ...



# Discover Architecture ...



# Discover Collaboration ...



# Middleware Operation ...

- The key mechanisms are
  - Discovery of services and applications
  - Security across servers
  - Collaboration across servers
  - Distributed locking
  - Distributed logging

# Discover

- The main idea here is the design of a middleware substrate that enables interoperability between multiple, distributed and independently managed instances of an entire collaboratory.
- The primary goal was to provide global access and enable sharing of resources across these instances

**THANK YOU !!!!**