Three Design Patterns for Secure Distributed Systems

Alan H. Karp
Kevin Smathers

Hewlett-Packard Laboratories

The Eight Fallacies of Distributed Computing

The network is reliable
Latency is zero
Bandwidth is infinite
The network is secure
Topology doesn't change
There is one administrator
Transport cost is zero
The network is homogeneous

Peter Deutsch

Aspects of Security Not Covered

- Physical security
  - Has the hardware or software been corrupted?
  - Is someone reading my keystrokes?
  - Is my private key safe from a sledgehammer?
- Authentication
  - Who am I talking to?
- Authorization
  - How do I decide what this party is allowed to do?
- Non-repudiation
  - Can I pretend I never made this deal?
- Privacy
  - Can someone I don't know about see this thing?

Aspects of Security Covered

- Anonymity
  - Who can know I've seen this thing?
- Auditing
  - How do I know who did what to whom when?
- Access control
  - Should I honor this request?
- Denial of service
  - Can my machine be rendered unusable?

Assumptions

Modest scale à Extremely large scale

Relatively static à Wildly dynamic

Trusted remote systems à Malicious remote systems

Reasonably homogeneous à Heterogeneity rules the day

Little sharing across domains à Be all and end all

Patterns

- Separate authorization from access control
  - Enhances scalability
  - Eases crossing administrative boundaries
- Mediate between requester and service
  - Enforces audit policies
  - Simplifies access control decisions
- Use a local proxy for remote users
  - Improves control over remote parties
  - Prevents some attacks
  - Limits damage done when attack succeeds

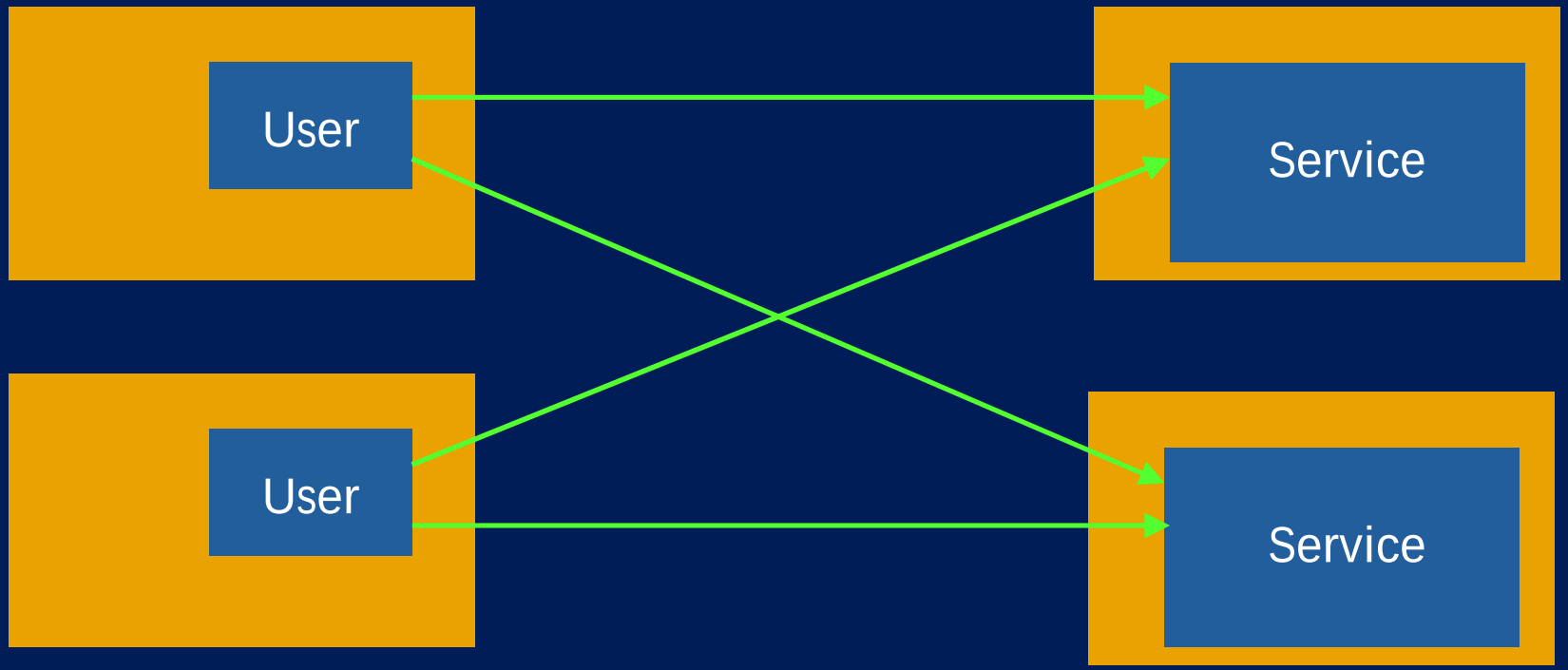
What These Patterns Can't Help

- Attacks against underlying OS
- Attacks against network stack
- Many attacks against the application
- Many denial of service attacks
- Misplaced trust
- Social engineering
- Careless users

Pattern 1

Separate Authorization From Access Control

Existing Systems



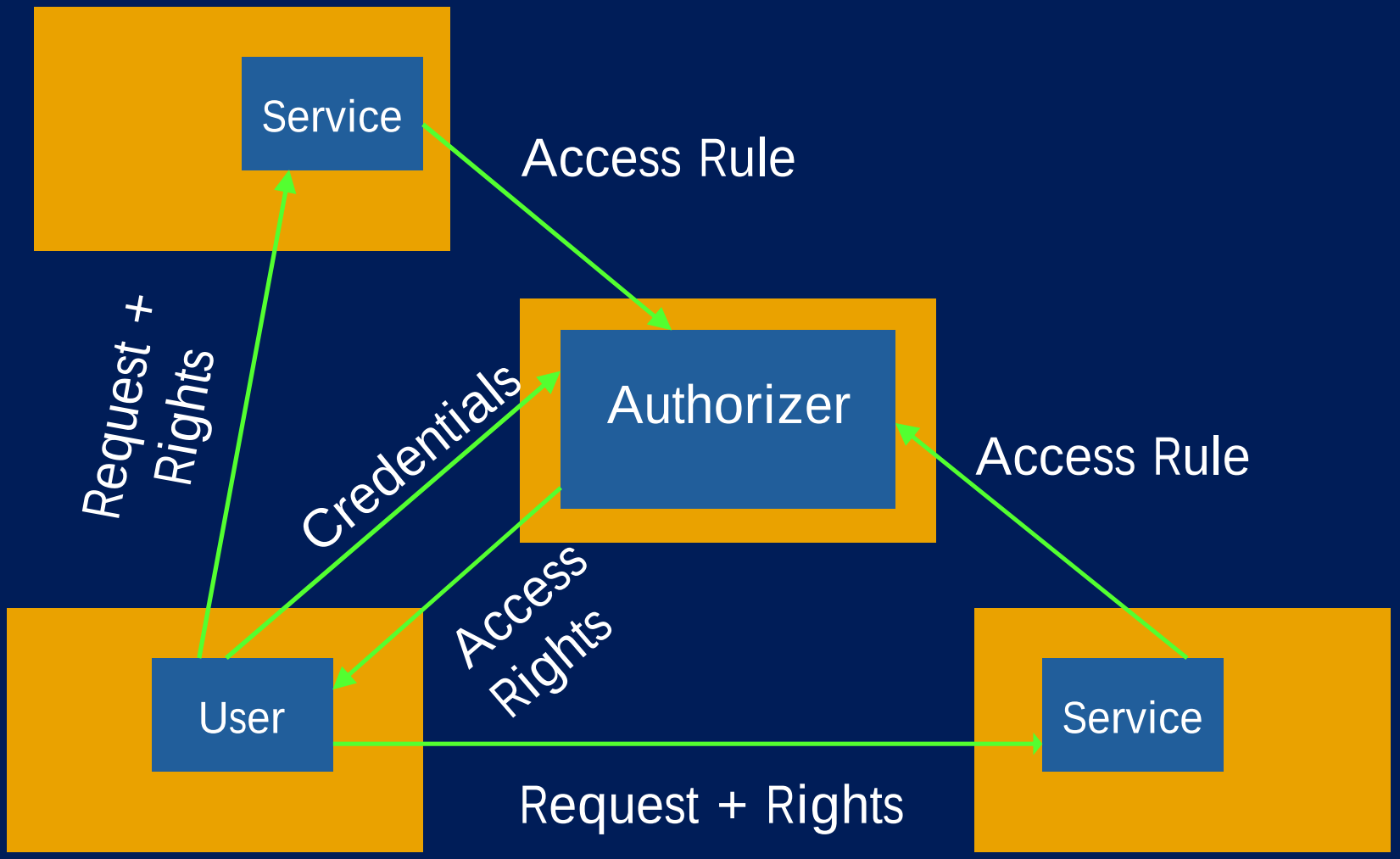
The Sad Tale of Zebra Copy

- Mom and Pop copy center
- Contract with HP
  - 2,000 HP employees with right to order jobs
  - HP told Zebra Copy when an HP employee changed jobs
  - Zebra copy updated its list
- Problem
  - HP has over 10,000 such partners
  - Zebra copy had over 500 such contracts
- Zebra copy now out of business

Problems

- User needs credential per service
  - different types of credentials
  - different rules for use
  - too many passwords
- Every service needs to do
  - authentication
  - authorization
  - access control
- Too many updates
- Authentication doesn’t tell what service needs to know

Access Pattern



Zebra Copy Done Better

- HP and Zebra Copy reach agreement
  - Zebra Copy gives HP a token representing contract
  - Order honored if accompanied by valid token
- Zebra Copy responsibilities
  - Verify token validity
  - Revoke and re-issue when asked by HP
- HP responsibilities
  - Delegate token properly
  - Revoke privilege when people change jobs
  - Pay for any use involving a valid token
  - Ask for replacement token when necessary

Benefits

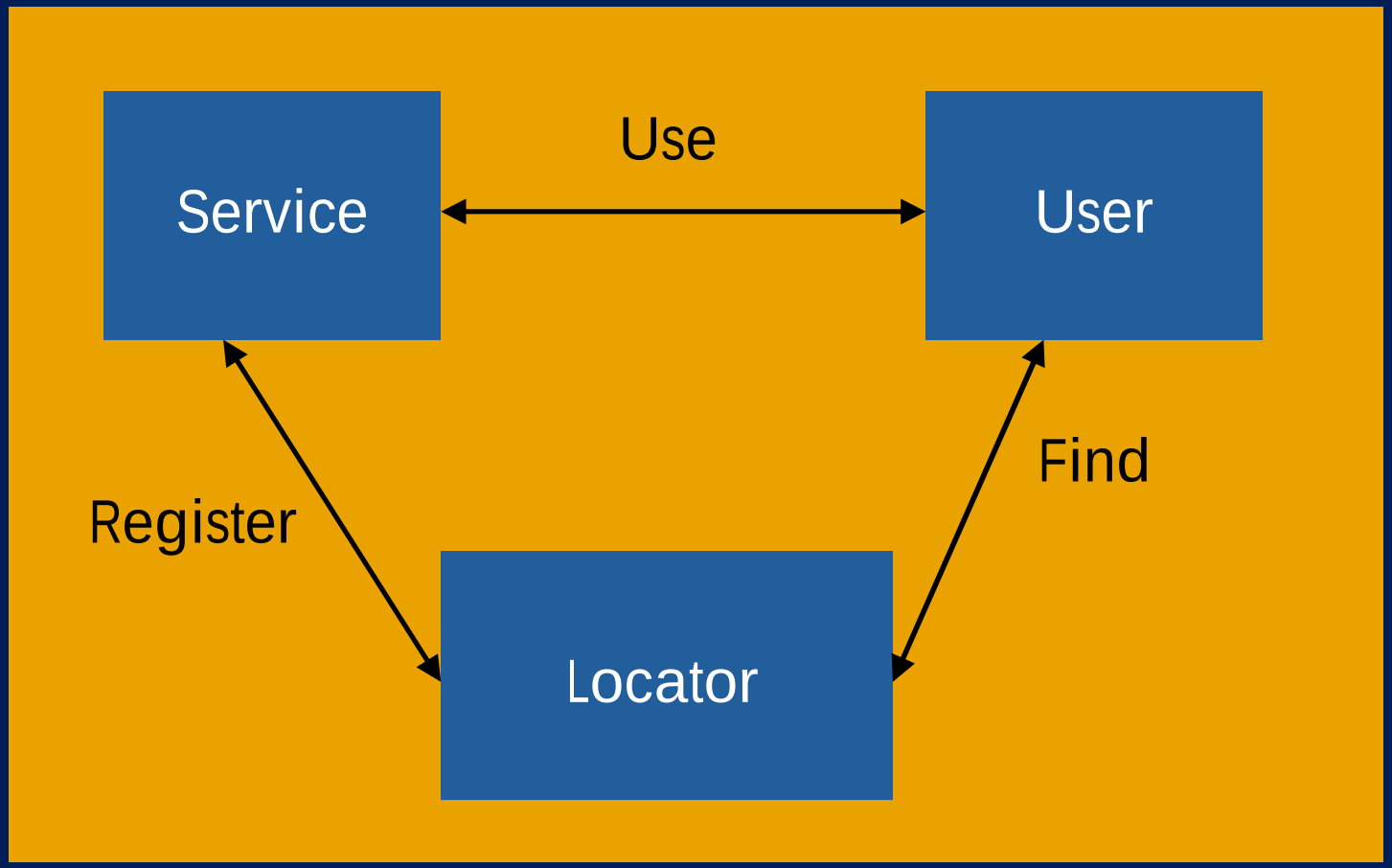
- Service only needs simple access control mechanisms
- An access right can be meaningful to one service
- Rules can be registered by third party
- Each company manages its own employees
- Admits multiple management schemes for one service
- Can add new service, types of access, etc. without affecting management mechanisms
- Can add new management mechanisms without affecting services

Pattern 2

Mediate between

Client and Service

Existing Systems



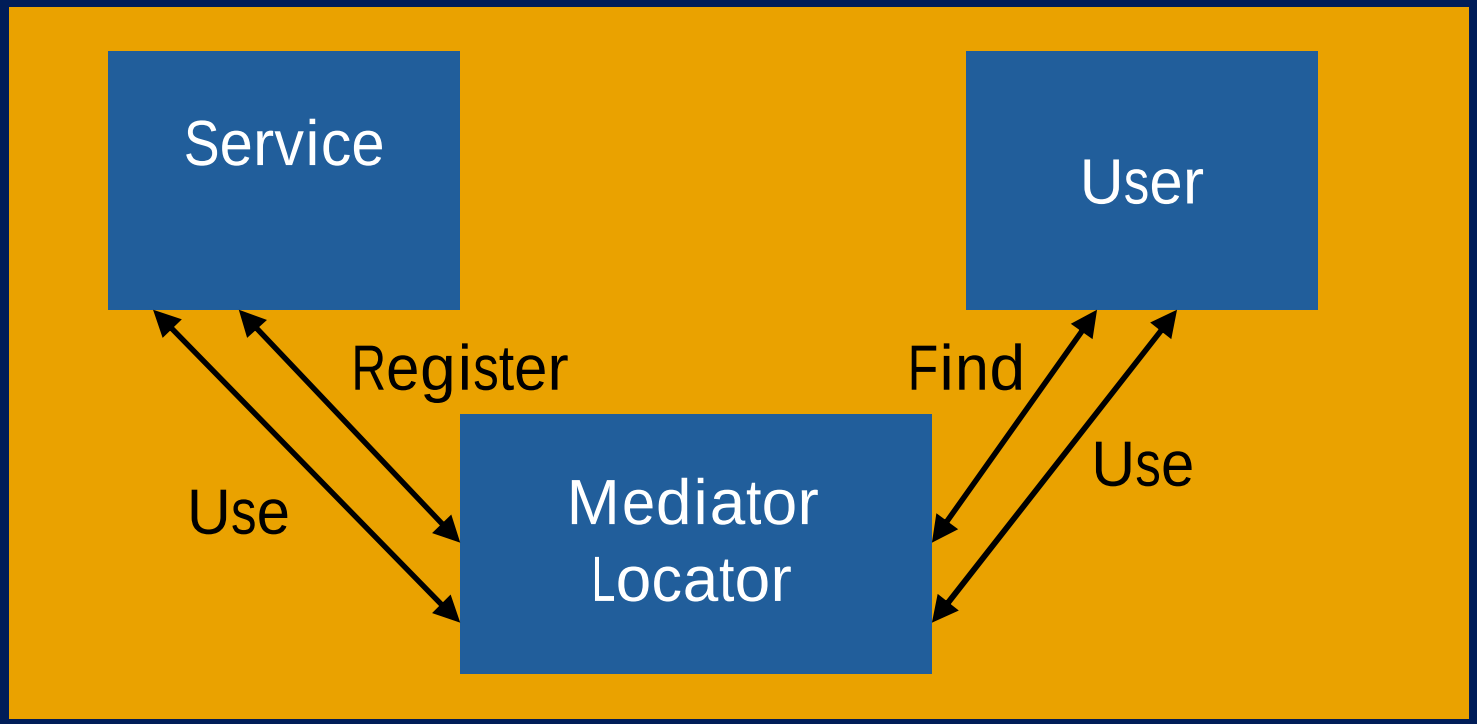
The Jini Jive

- Designed for sharing of resources in trusted workgroup
  - Promised access control within a year
  - Still doesn't have it (some talk about ACLs but no details)
- System structure is the problem
  - Look up the service by interface
  - Invoke service directly
- Control points
  - JavaSpace only knows interface, not specifics of service
  - Service is a printer
- Do you really want to do authentication, authorization, and access control in every device?

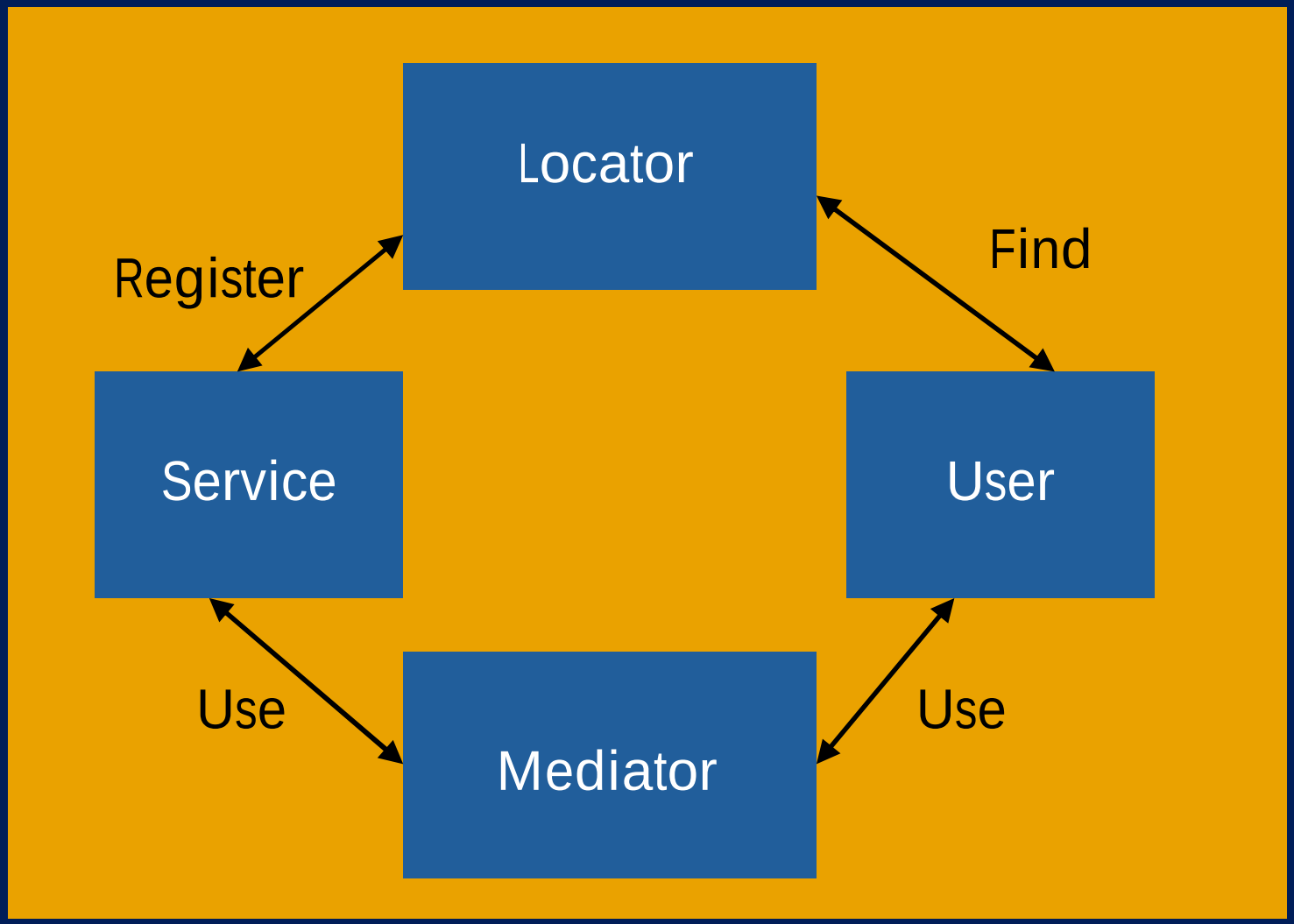
Problems

- Each service needs to
  - authenticate users
  - make authorization decisions
  - produce audit trail
  - protect against malicious or erroneous users
- Common library linked with service
  - guaranteeing proper use
  - rolling forward versions

Mediation Pattern



Mediation Pattern

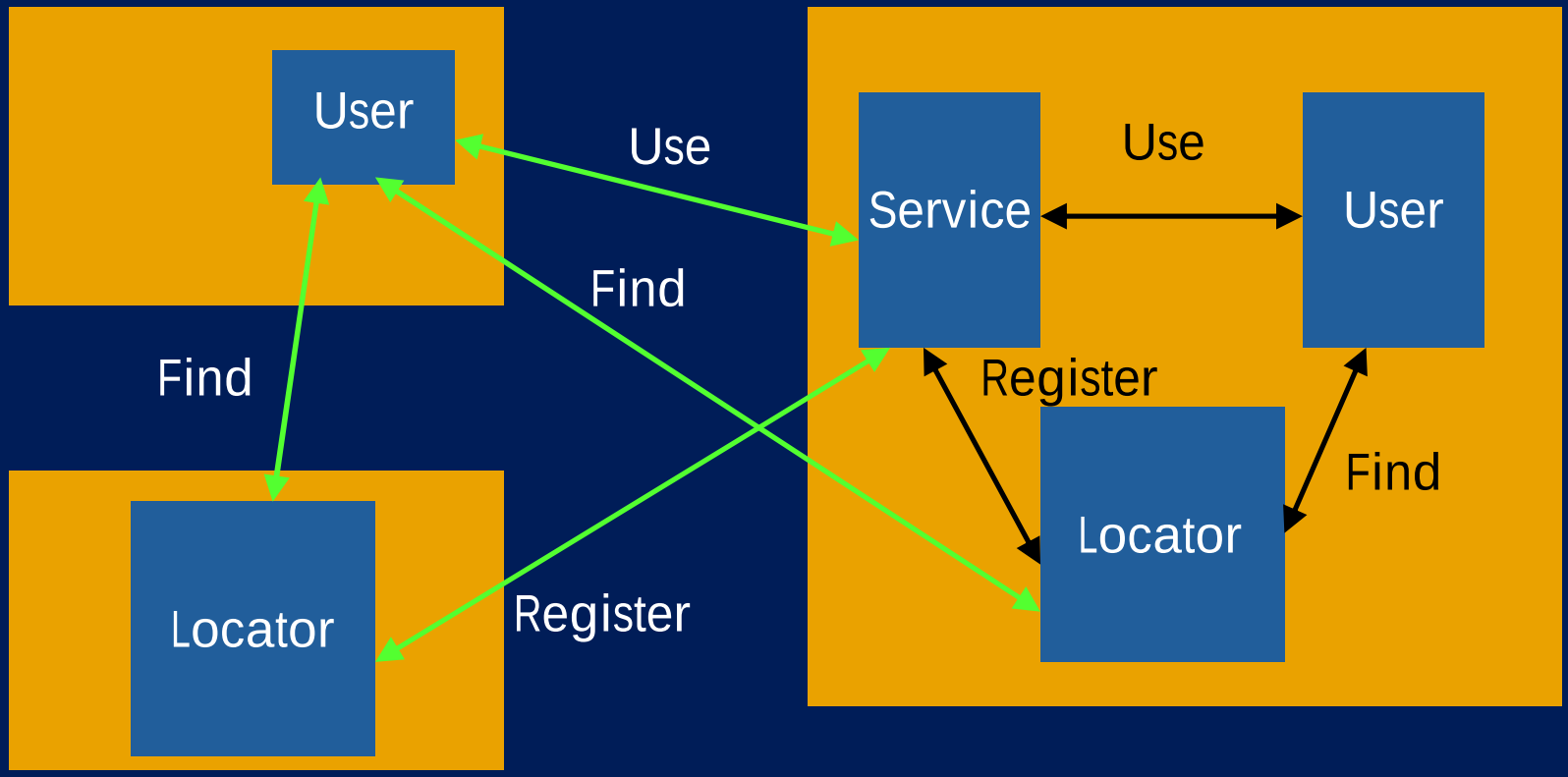


Benefits

- Mediator provides
  - trusted path
  - additional metadata
  - mutual anonymity
  - audit information
- Service and client don’t have to trust each other directly
  - turns N*M problem into N+M
- Mediator can translate authorization information into a form meaningful to service
- Mediator can verify protocol adherence

Use a Service-side Proxy for Remote Users

Existing Systems



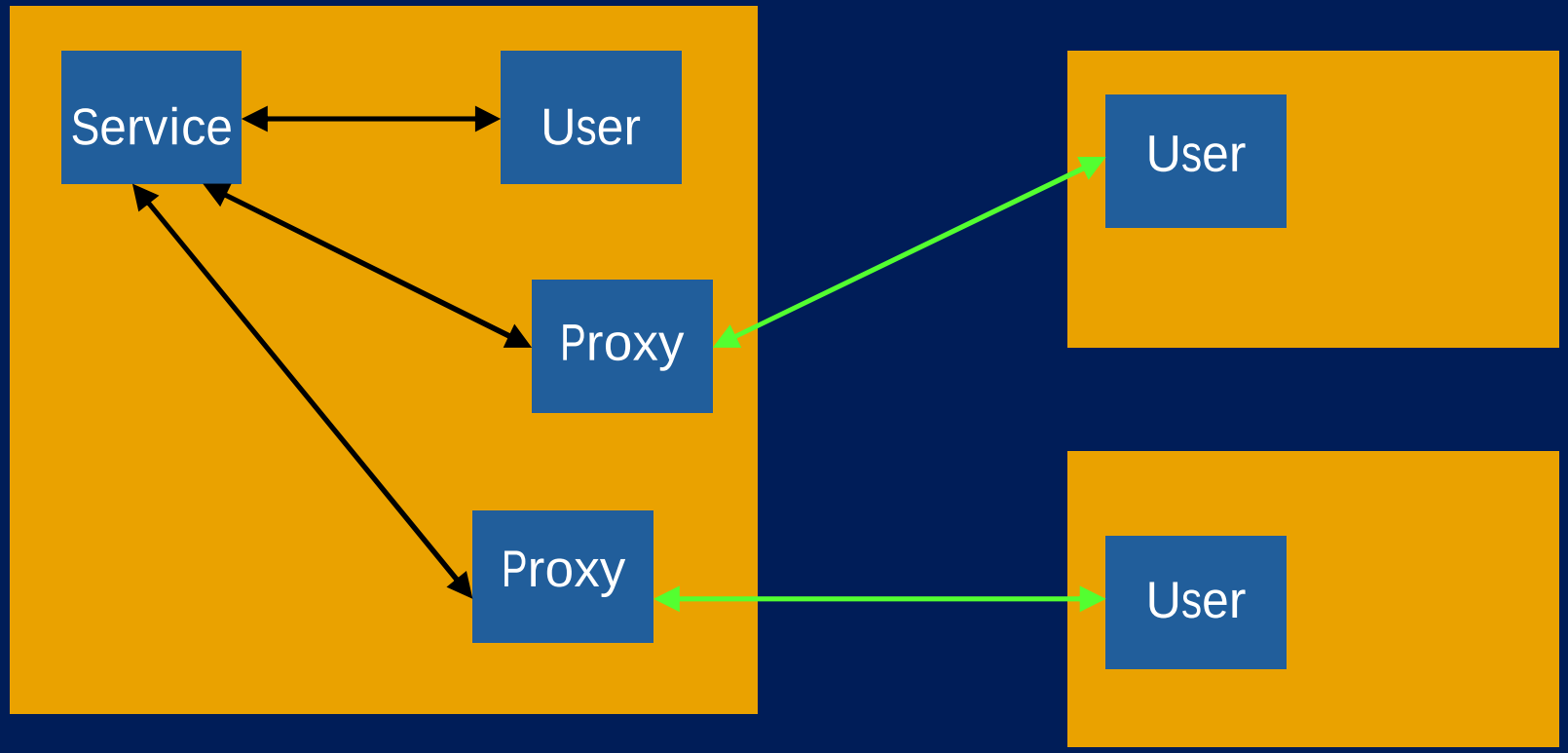
Web Server Woes

- Clients talk directly with server
  - denial of service attacks hard to prevent
- Server has more privileges than any client
  - confused deputy a possibility
  - successful attack can give client all server privileges
- No place to enforce system policies

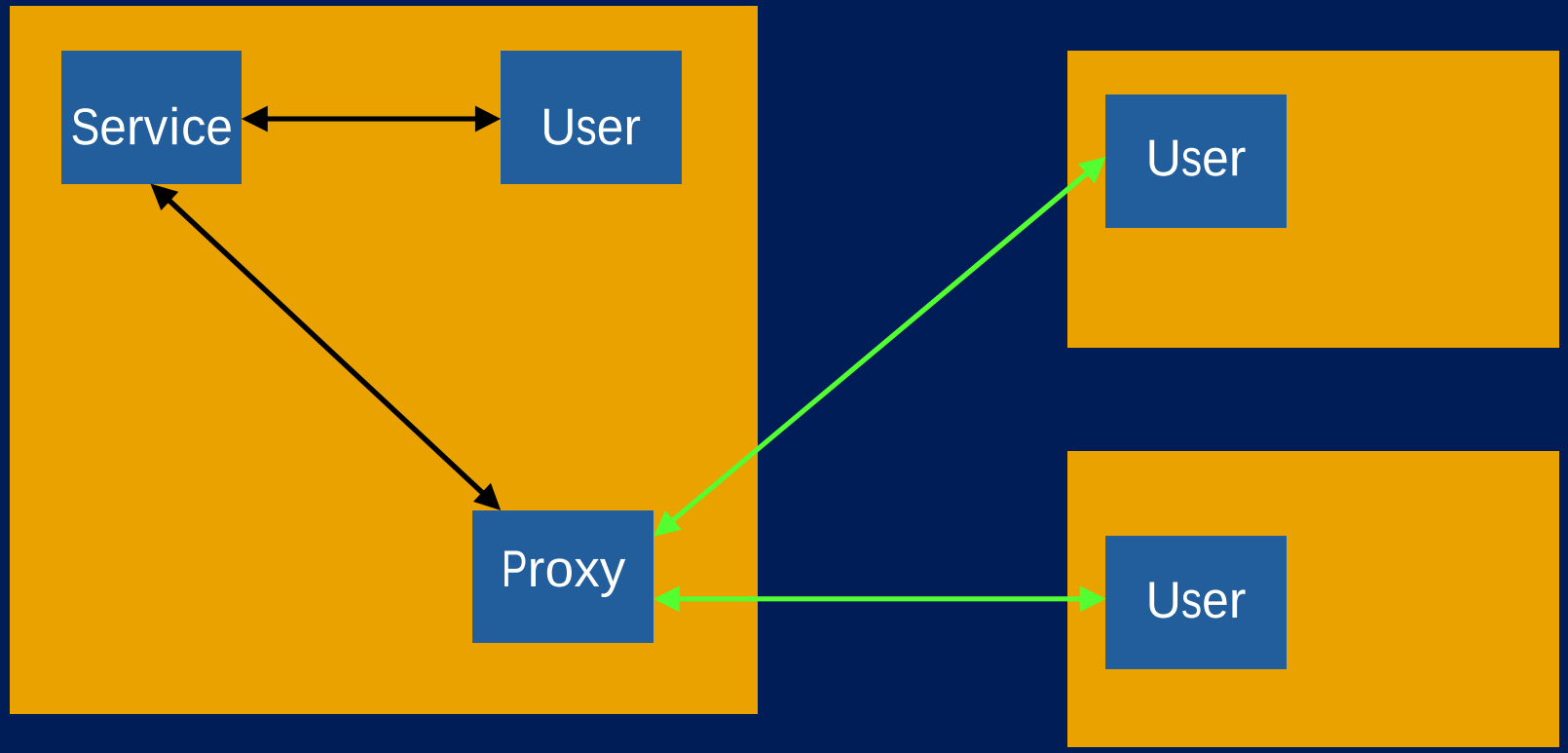
Problems

- Every service subject to a variety of network attacks
- Successful attack can do anything service can do
- Crashing server cuts off all users
- Service must
  - deal with authentication, authorization, access control
  - enforce machine-specific policies

Proxy Pattern



Use a Proxy for Remote Users



Benefits

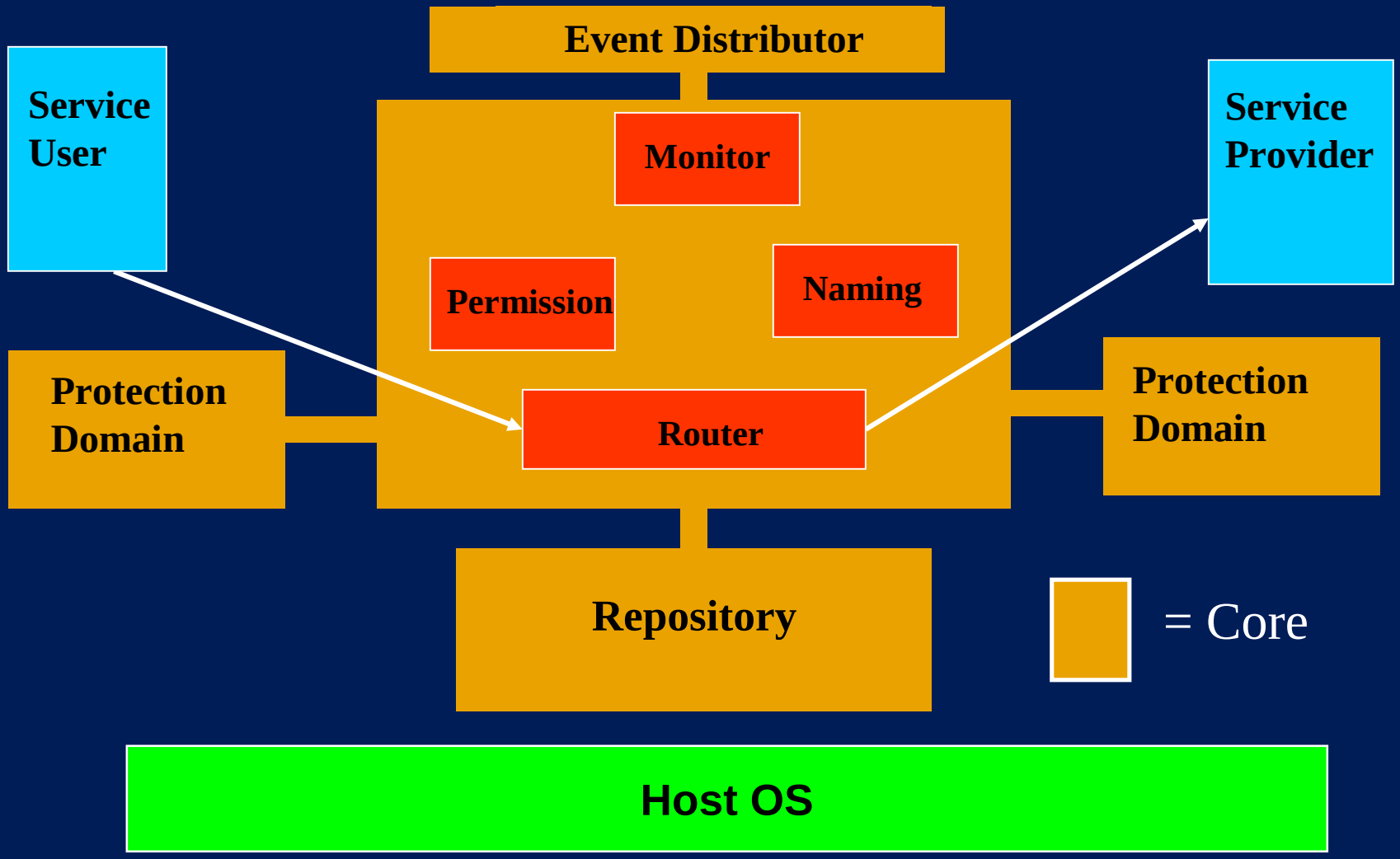
- Crashing proxy cuts off attacker
- Protection domain of remote user under local control
- Proxy can be dumb and repeat requests verbatim
- Proxy can be smart and filter requests
- Smarts added to proxy implementation protect all services

Example of Use

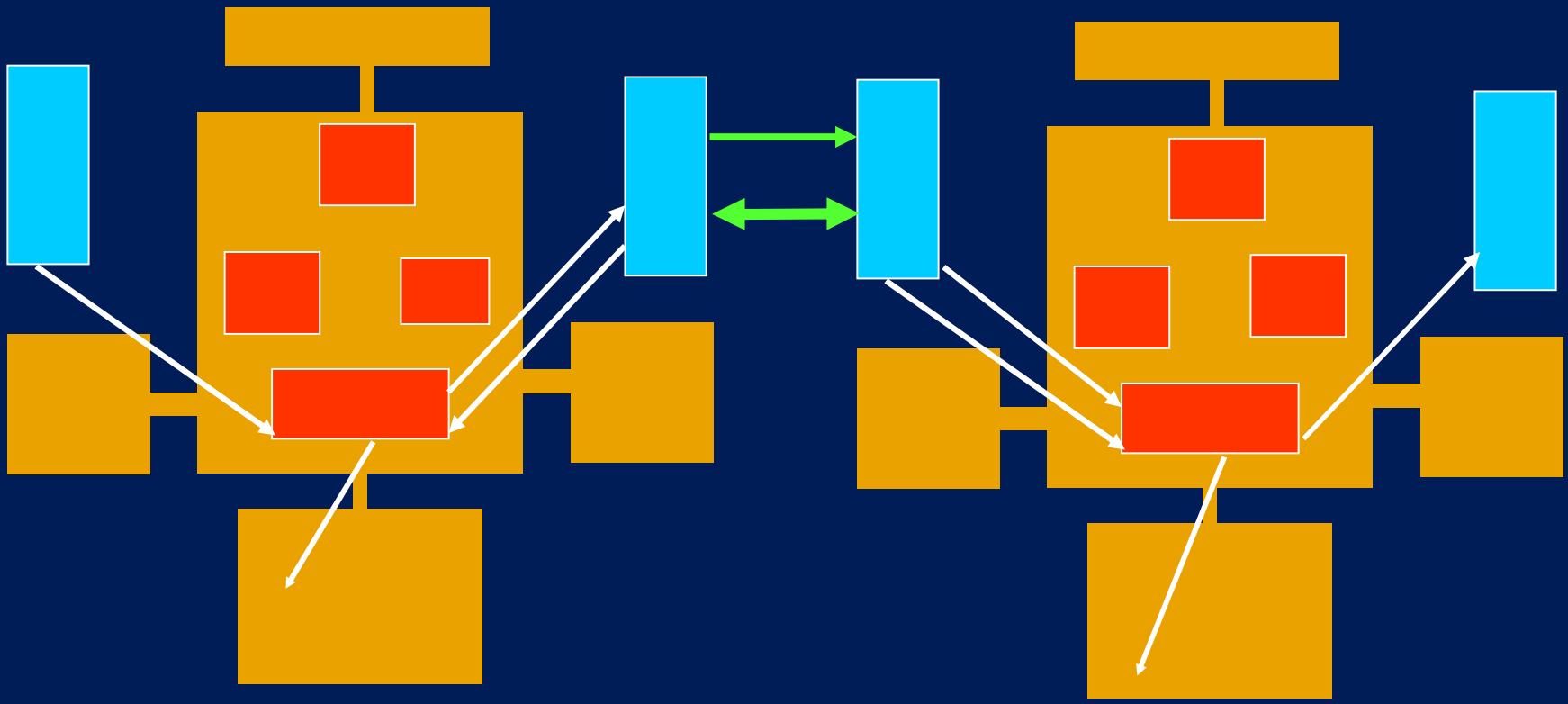
E-speak E-xplained

- First attempt at a web services infrastructure
  - Alpha 1.0 – May 1999
  - Beta 2.2 – December 1999
  - Release 1.0 – June 2000
- Several novel features
  - Access control
    - Split capabilities for access control (thru Beta 2.2)
    - SPKI capability certificates (1.0)
  - Extensible ontology system
  - Path dependent naming
  - Powerful event system

Single Machine View



Using a Remote Resource



Key Features

- Service sees only local requests
- Client makes only local requests
- Only proxies know about the wire
- Brokering comes for free
- Secure firewall traversal for messages
- Events are messages

Design Principles

- Separation of responsibility
  - Authorities determined when protection domain attached
  - Access control independent of authorization mechanism
  - Only components that talk on the wire know about the wire
- Separation of control
  - Each machine sets own policy via proxy protection domains
  - Different mechanisms for authentication and access control
- Limit damage done by a successful attack
  - Limited permissions of proxies
  - Mediator can filter some bad requests
  - Strong and rapid revocation

Summary

Fundamental Idea

- Design patterns
  - useful in object oriented coding
  - no reason they won't help in distributed systems
- Propose using three patterns
  - separate authorization from access control
  - mediate between client and service
  - use a local proxy for remote users
- Others
  - connection manager
  - your favorite here

A Definition

A distributed system is one in which a component I never heard of can make my computer unusable.

Leslie Lamport


hp
invent