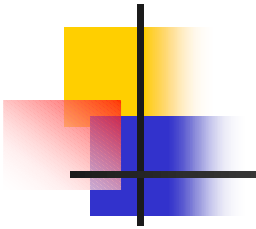


Ribbet!

Cluster-based Computing
with
Active, Persistent Objects on the Web

Frank Sommers
(fsommers@autospaces.com)

Shahram Ghandeharizadeh
(shahram@cs.usc.edu)

Shan Gao
(sgao@cs.usc.edu)

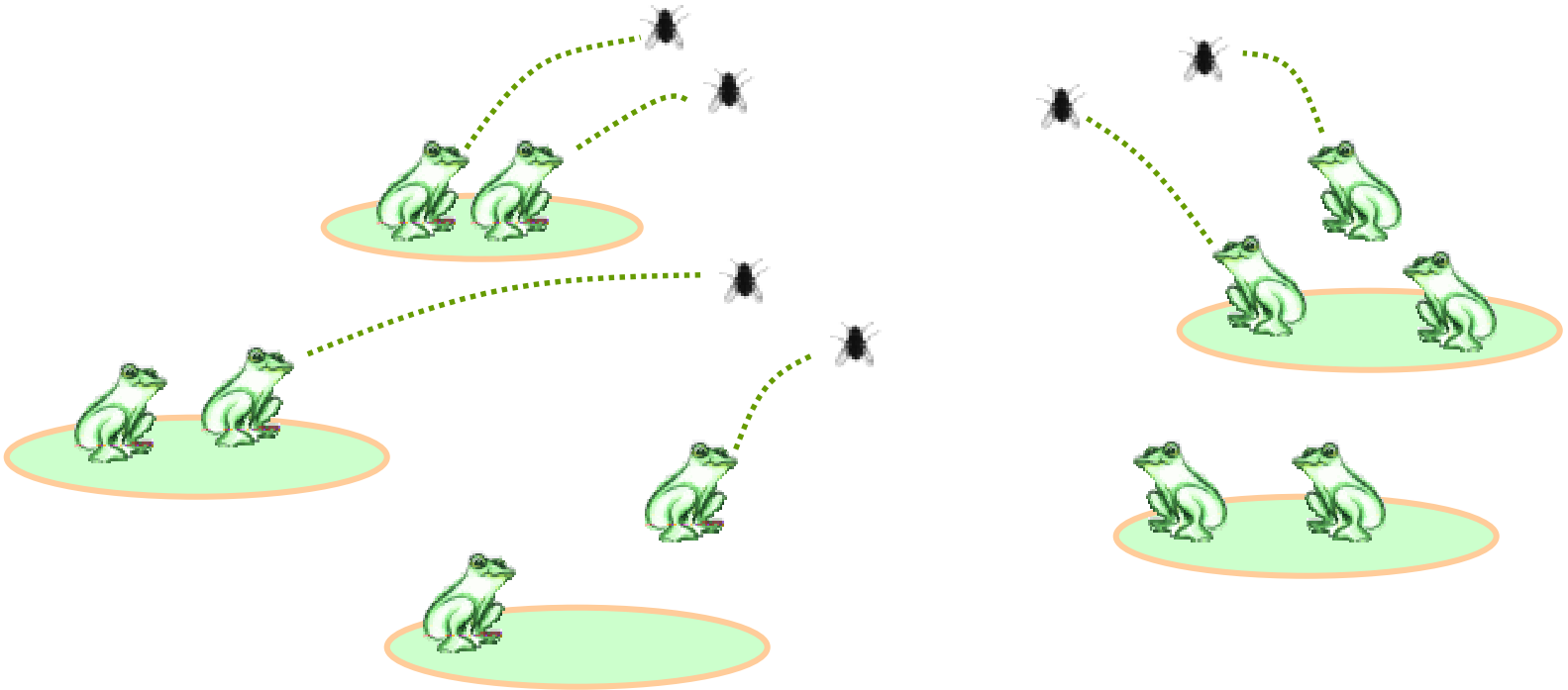
How do you catch a *Cluster Fly?*

Cluster Fly

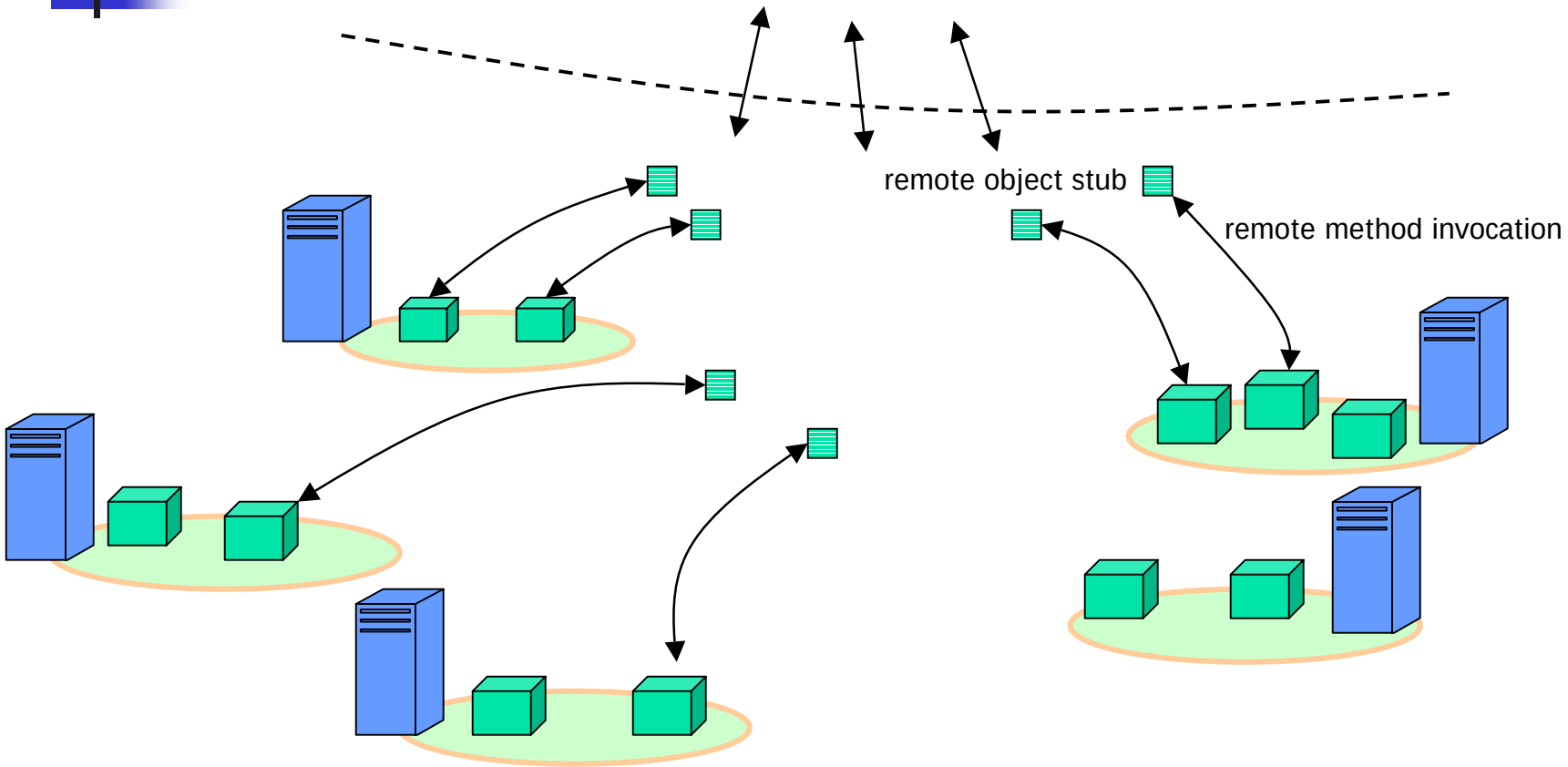
(Pollenia rudis)

Photo: Courtesy of Virginia Polytechnic Institute

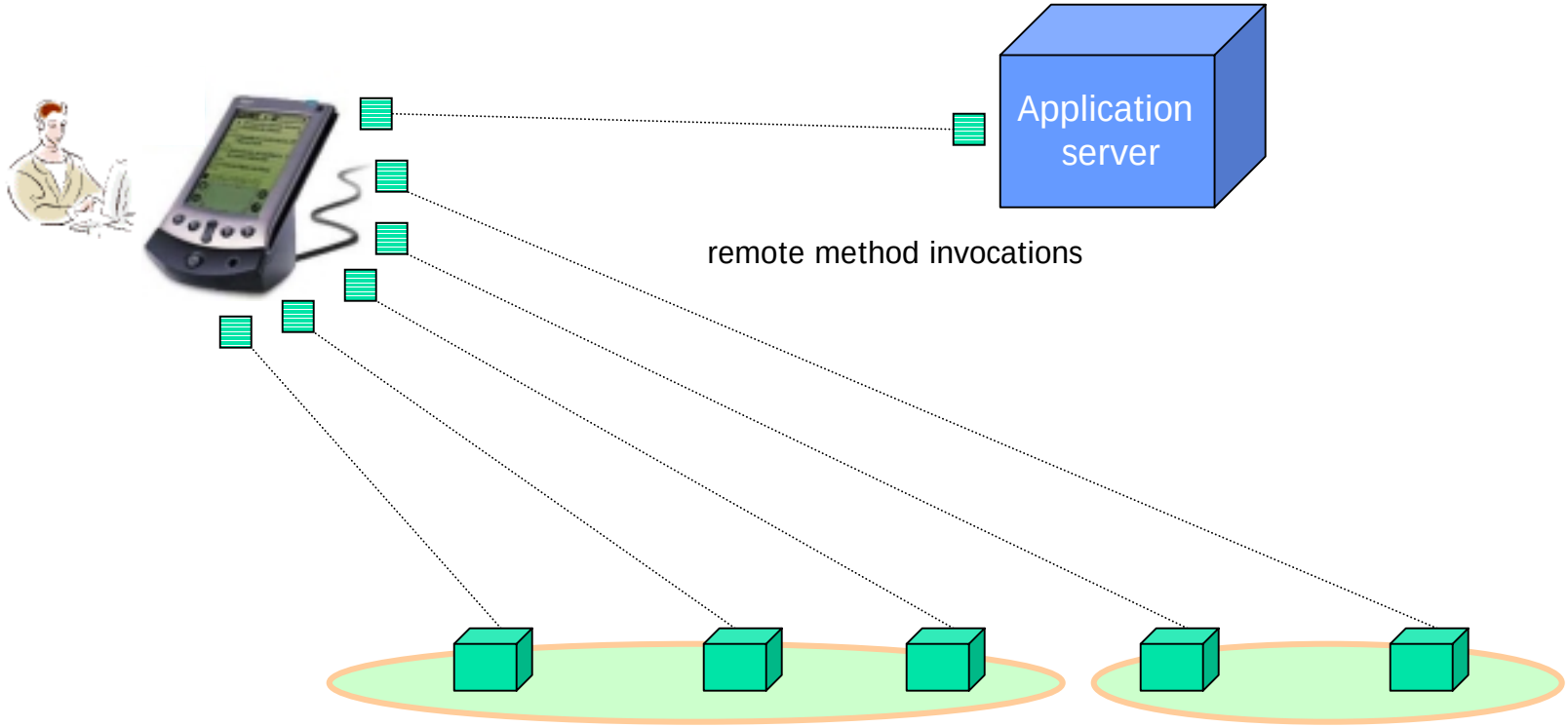
With a cluster of frogs!



Objects on a cluster

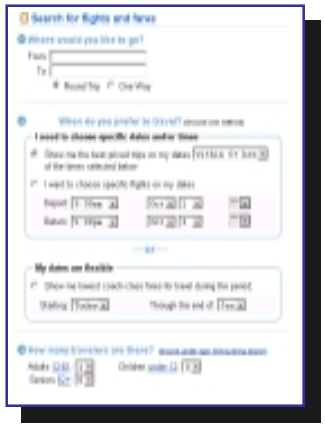


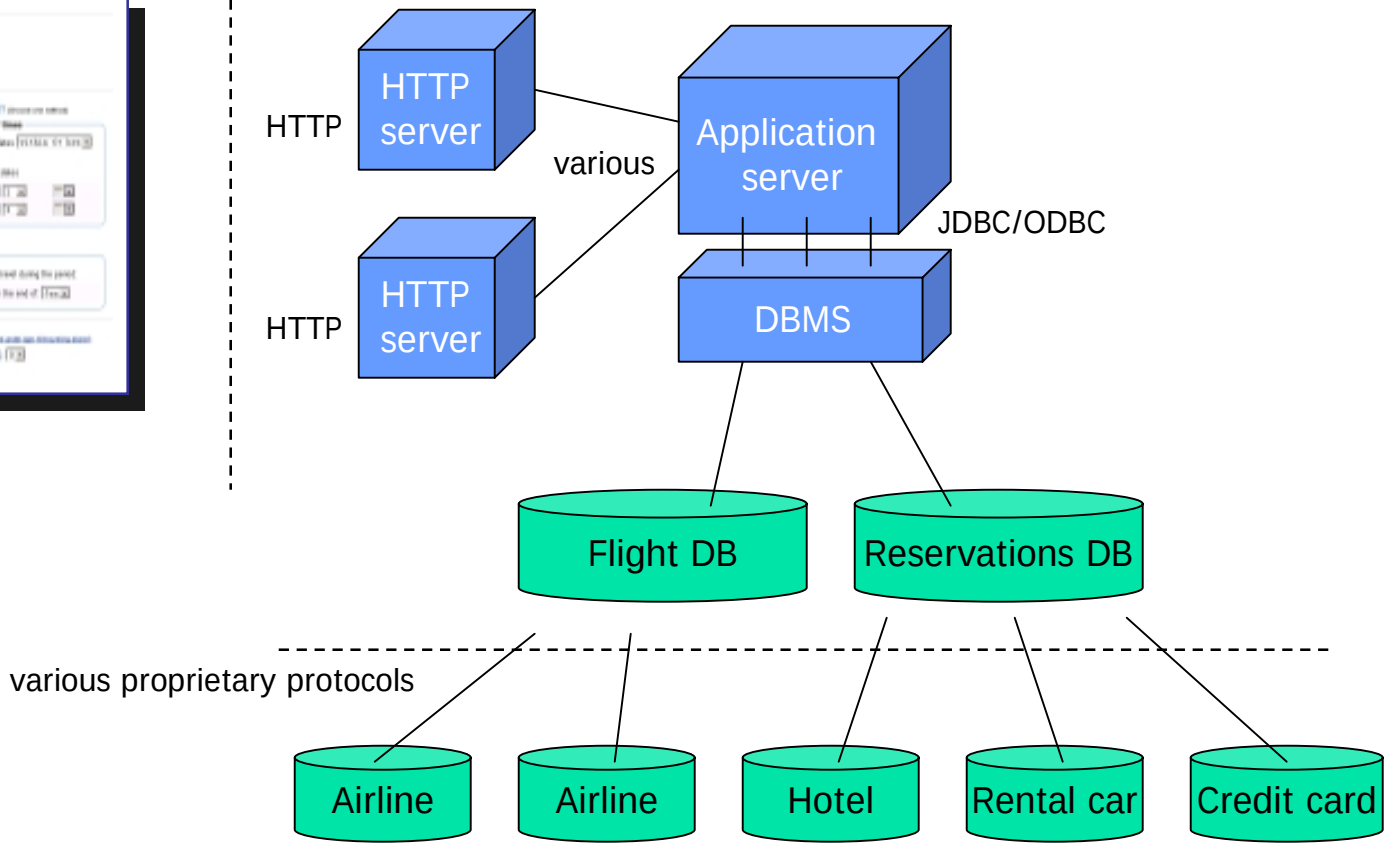
A Proxy-based Web service

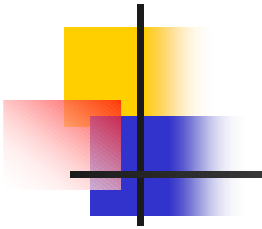


How does the cluster help?

A flight reservation Web service

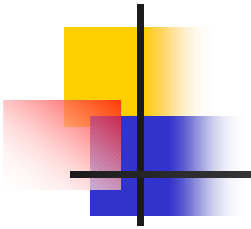






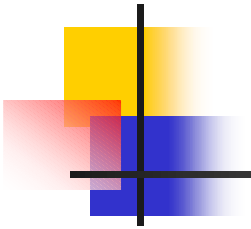
A few challenges...

- Multi-database application:
  - Data is distributed among the enterprises involved
  - Each enterprise might use its own protocol for access to their portion of data
  - Reliability, security differ at each site
- Network introduces possibility of partial failure
  - Service must operate reliably in the presence of partial failure



XML helps...

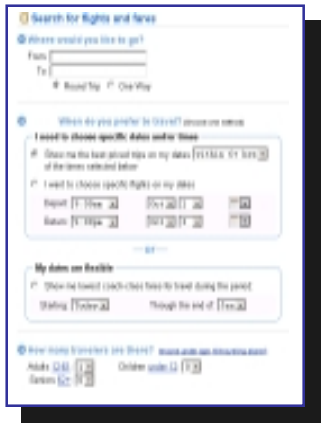
- XML schemas help bridge heterogeneous data formats
- XML RPC-style mechanisms help bridge heterogeneous protocols

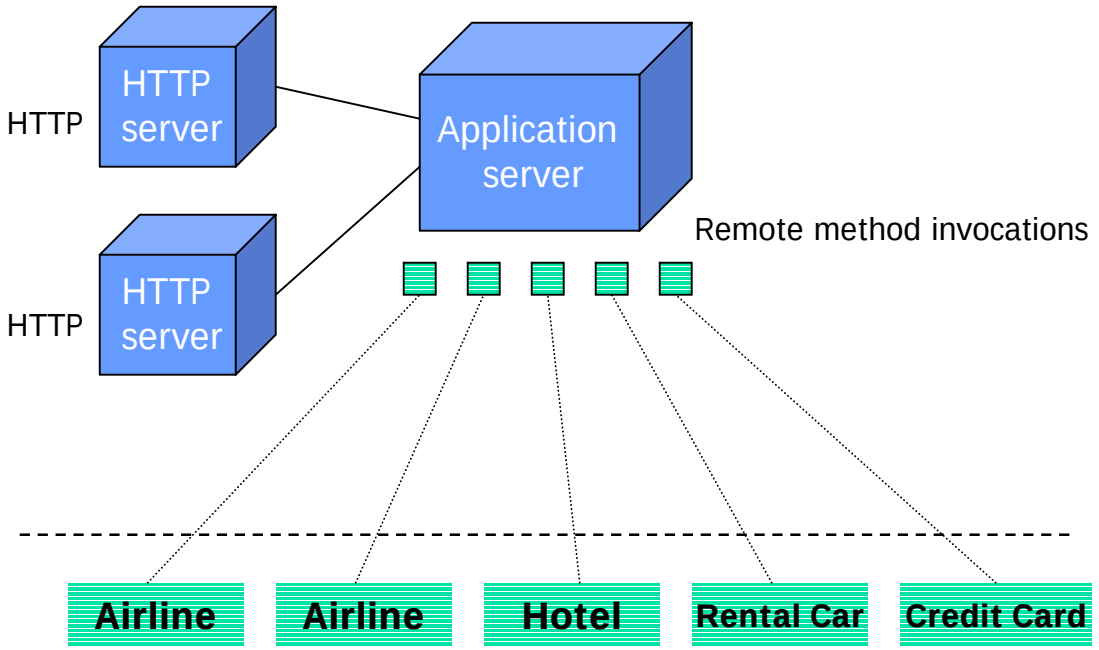


But…

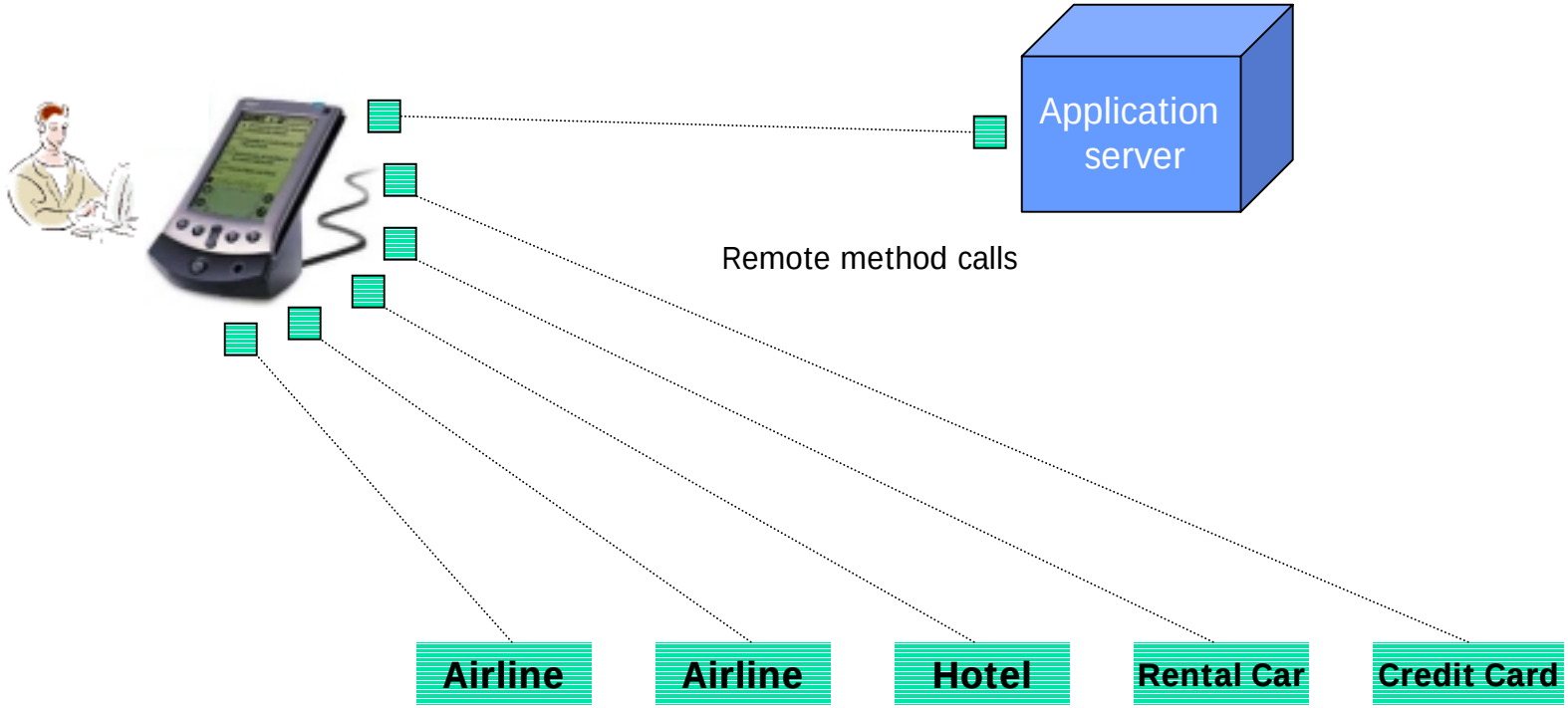
- Not an infrastructure for high-availability:
  - Services might be used by other services.
  - Failure of one service would cause cascading, Byzantine-style failures
  - Unreliable services might be automatically eliminated over time (“Network-darwinism:” Only the “fittest” services survive)
- Not an infrastructure for ubiquitous availability of data

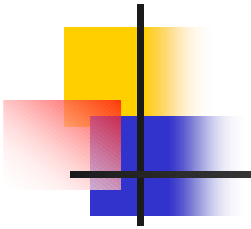
Service proxies





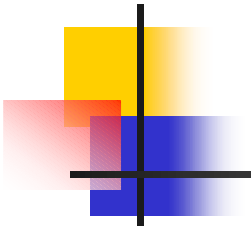
A Proxy-based Web service revisited





Proxies are objects

- Represent service to other services or to humans
- First-class objects in a programming language's type system
  - Represent semantics as well as data
  - Can implement any exception handling strategy
  - Can implement any (or many) communication protocols, e.g., XML-RPC, Java RMI, sockets, etc.



Benefits of a service proxy

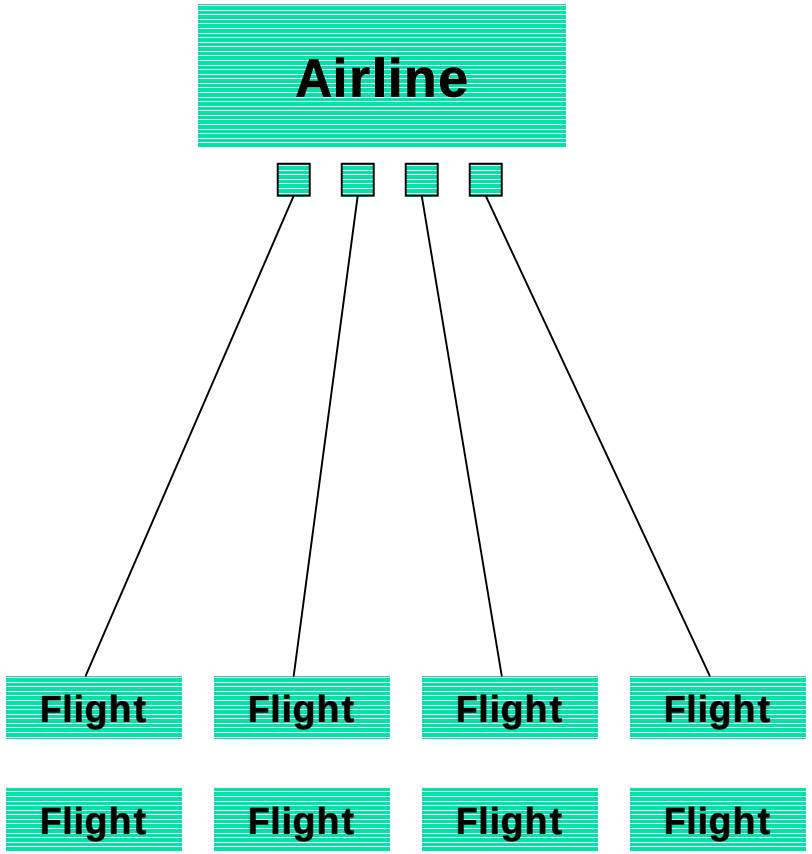
- Separates service's implementation from how that service participates in the network:
  - A service's implementation can change over lifetime of service
  - An implementation's location can change
  - An implementation can be replicated
- Can recover from intermittent network failures
- Independent of any communication protocol

Service composition

The data *is* the service!

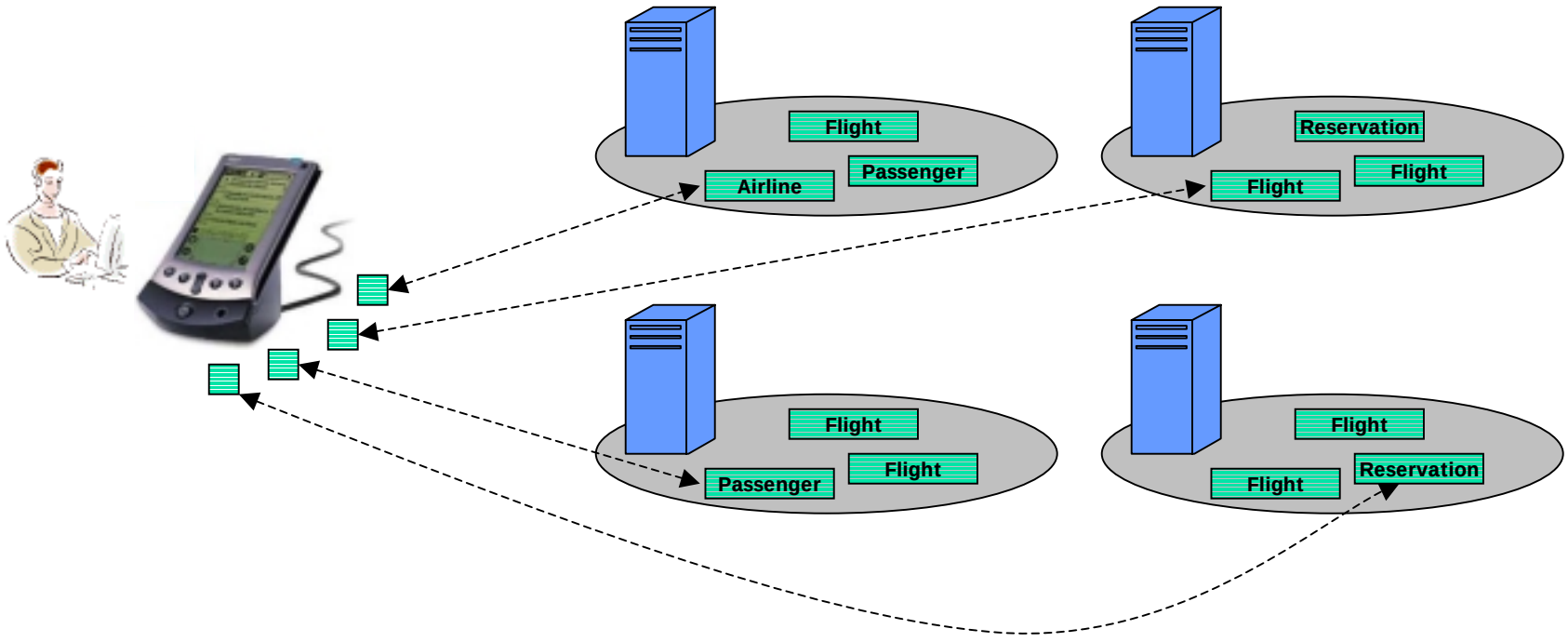
```
public interface Airline {

  public Flight[] getSchedule (Airport departure
            Airport destination,Date travelDate);
}
```

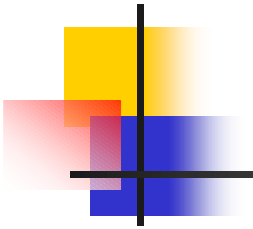
```
public interface Flight {

    public boolean available (int numSeats);

    public float getPrice (SeatClass seatClass);

    public Reservation reserveSeat (Passenger passenger,
            CreditCard card, seatClass seatClass);

    public FlightStatus currentStatus();
}
```



Cluster-based distribution

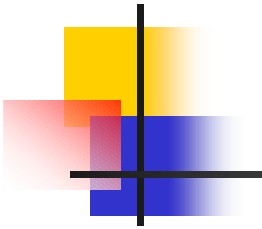
of active, persistent objects





Ribbet's goals

- Assume heterogeneous nodes
  - Any off-the-shelf (COTS) operating system
  - Inexpensive personal computer (PC) hardware
- Minimal administration
- Require no changes from clients of Web services
- Require only minimal changes from a service's implementation

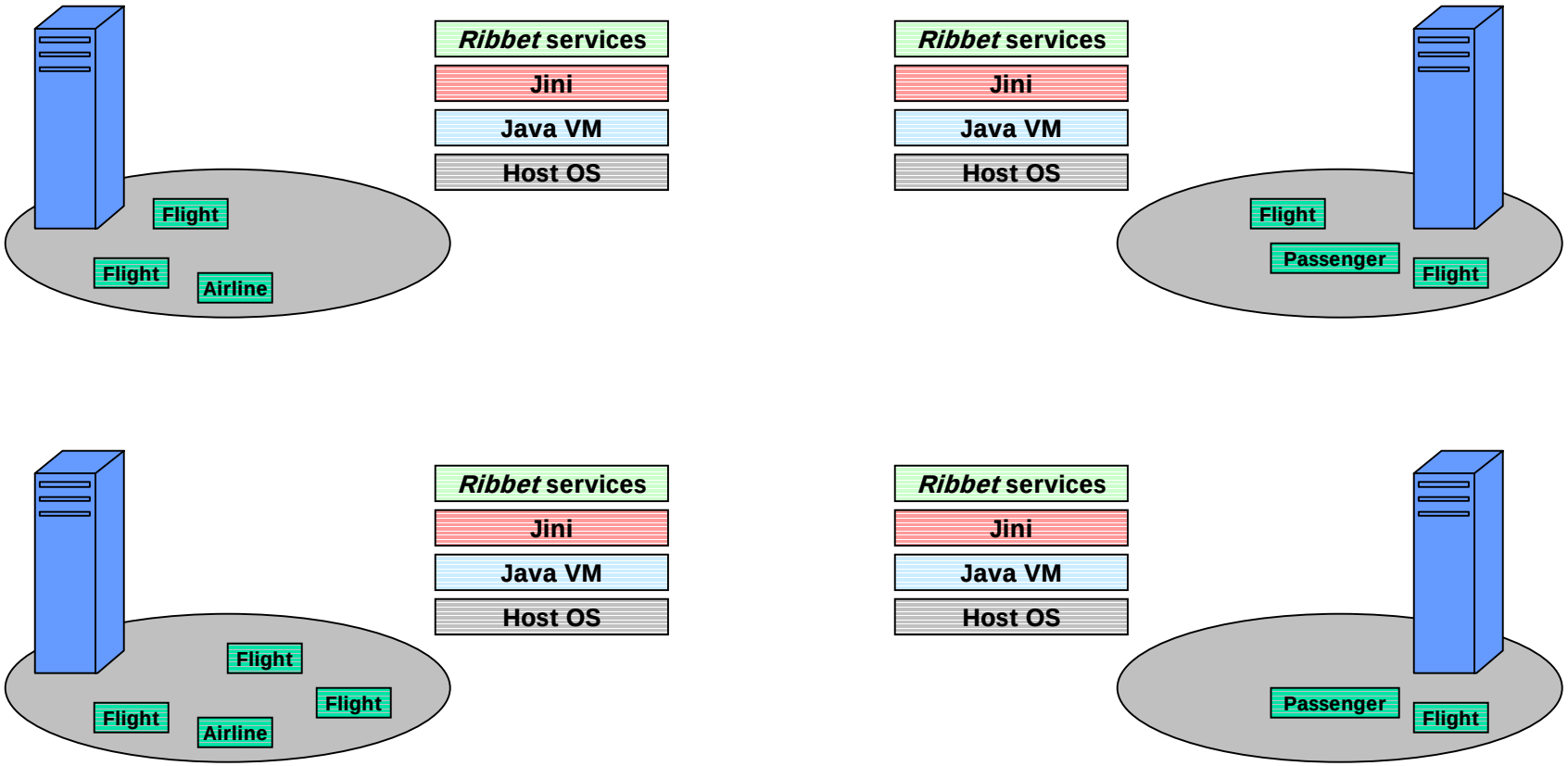


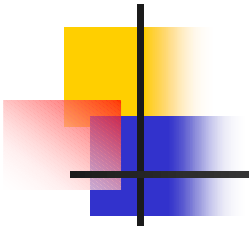
Ribbet's goals, cont'd

Minimize administration

- Automate management of cluster nodes
  - Nodes discover one another's presence and absence
  - Load/utilization information is shared between nodes
- Automate utilization of cluster nodes
  - Facilitate run-time load (re)distribution
  - Provide "single-system" view from the point of the service proxy

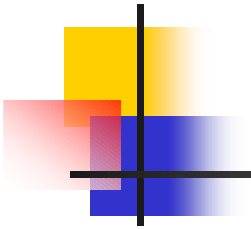
A Java-based implementation *with Jini*





A Java-based implementation

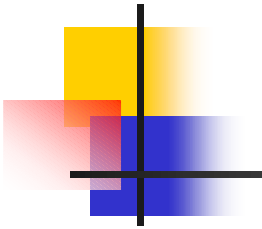
- Java Virtual Machine
  - Masks difference in hardware, operating systems
  - Facilitates dynamic loading of classes and objects from any network location
  - Supports object serialization
  - Fine-grained support for security
  - Support for many programming languages
  - Many implementations exists



A Java-based implementation

cont'd

- Java bytecode representation of objects
  - Preserves strong typing
  - Supports type evolution
  - Compact format
  - Supported by compilers for many programming languages
- Java API
  - Support for remote method invocation (RMI)
  - Support for object activation system (RMI activation)

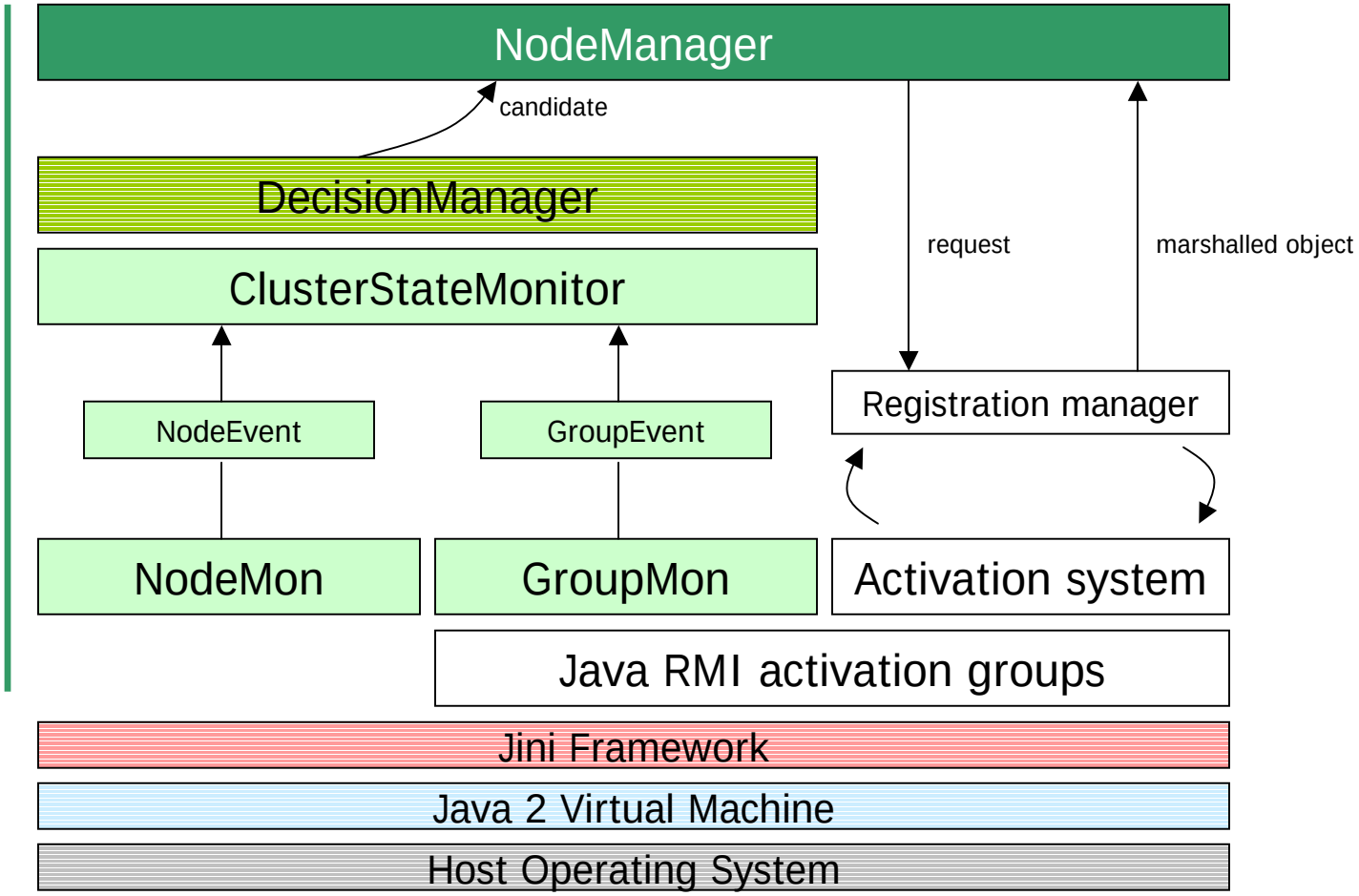


Jini

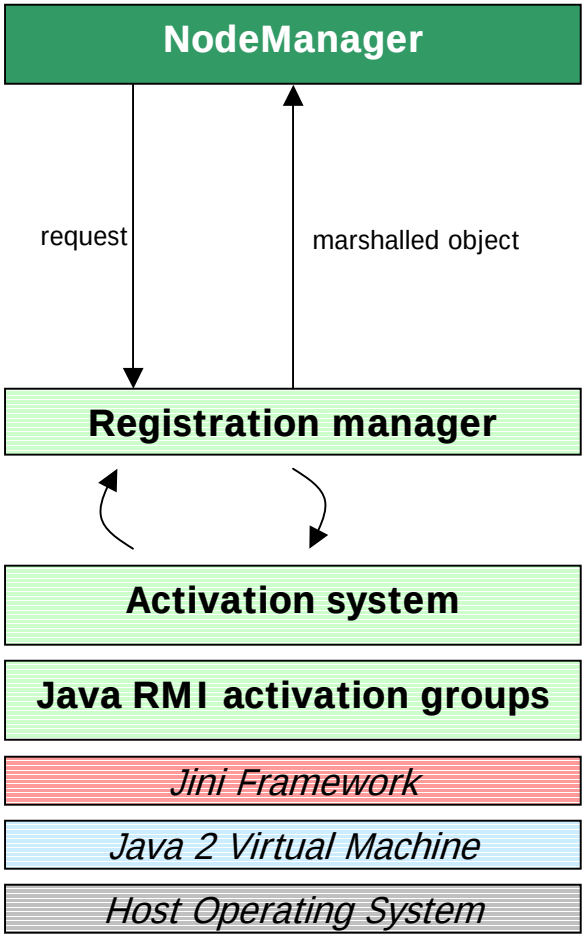
Java's distributed computing framework

- Proxy-based architecture
  - Jini proxies are Java objects
  - Based on semantics of Java RMI:
    - Dynamic code downloading
    - Object serialization
- Lookup/discovery mechanism
- Remote event support
- Leasing
  - Time-based reservation of resources

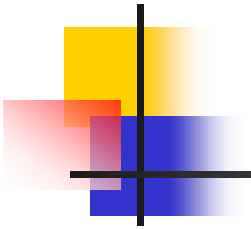
Ribbet's architecture



Activation system



- Register and unregister objects with a node
- Track number of active objects
- Track method invocations for each object
- Manage Java VMs for active objects
- Relay changes via Jini remote events (e.g., object registrations, unregistrations)

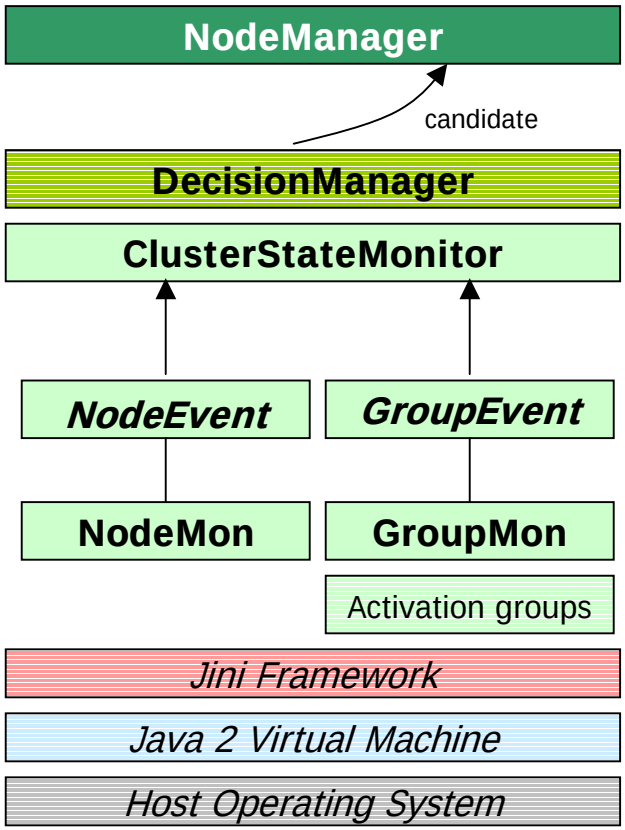


Activation system, cont'd

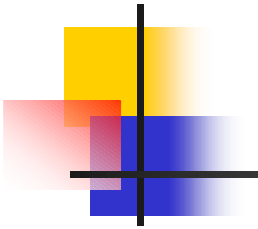
- Implements `java.rmi.activation.ActivationSystem`
- Offers semantics of Java RMI Activation Specification
- Additional capabilities:
  - Creates containers for objects to run in
  - Signals objects to become inactive
  - Automates object registration
  - Creates serialized stream for shipping an inactive, unregistered object
  - Generates remote events

Cluster events

Collecting group and node characteristics



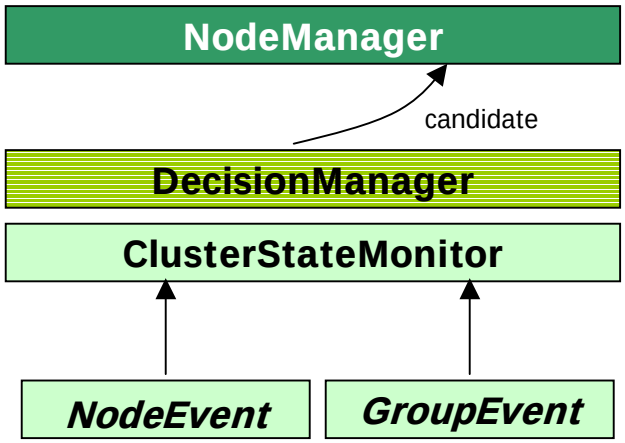
- Remote events
  - ***NodeEvent***: Relays changes in state of the node
  - ***GroupEvent***: Relays changes in state of an activation group
- **`ClusterStateMonitor`**
  - Track events not only from local node, but from all nodes in the cluster
  - A global state of cluster



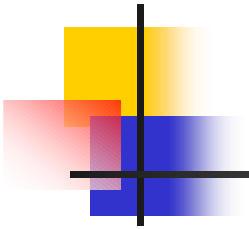
Cluster events, cont'd

- Jini remote event semantics
- **NodeMon** and **GroupMon** are interfaces. An implementation can specify:
  - What triggers events generation
  - What information is encapsulated inside event objects
- **NodeMon**'s implementation interfaces with an operating system's instrumentation API (Windows Management API, /proc filesystem, etc)
- **GroupMon**'s implementation obtains its information from the activation system
- Remote events are not guaranteed to arrive!

DecisionManager



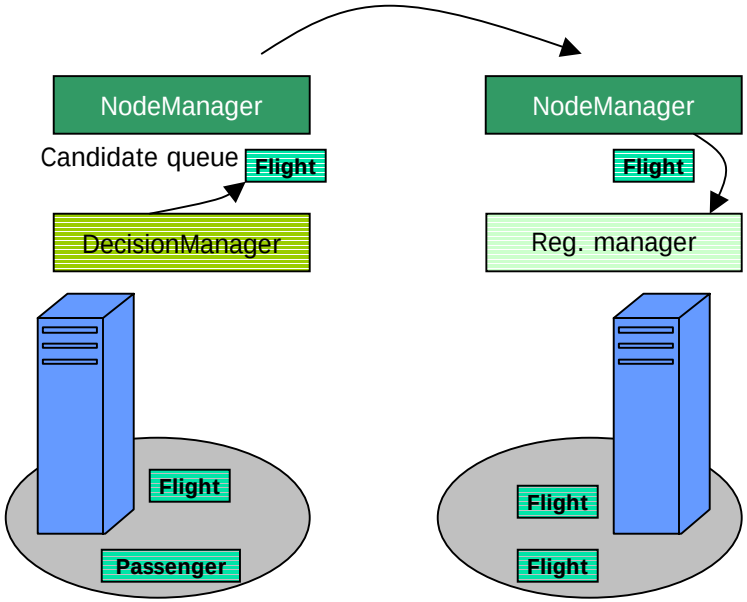
- Heuristics are configurable:
  - What to migrate?
  - When to migrate?
  - Where to migrate it to?



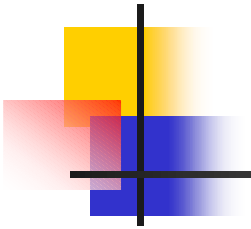
DecisionManager, cont'd

- Can accommodate *any* heuristics
- Acts as an expert system:
  - Events are "facts" inserted onto a white board.
  - A set of facts trigger the activation of rules
  - A rule's activation leads to the selection of a migration candidate
- Other implementations are possible
- Heuristics depend on application domain
- Currently: Even distribution of objects

NodeManager



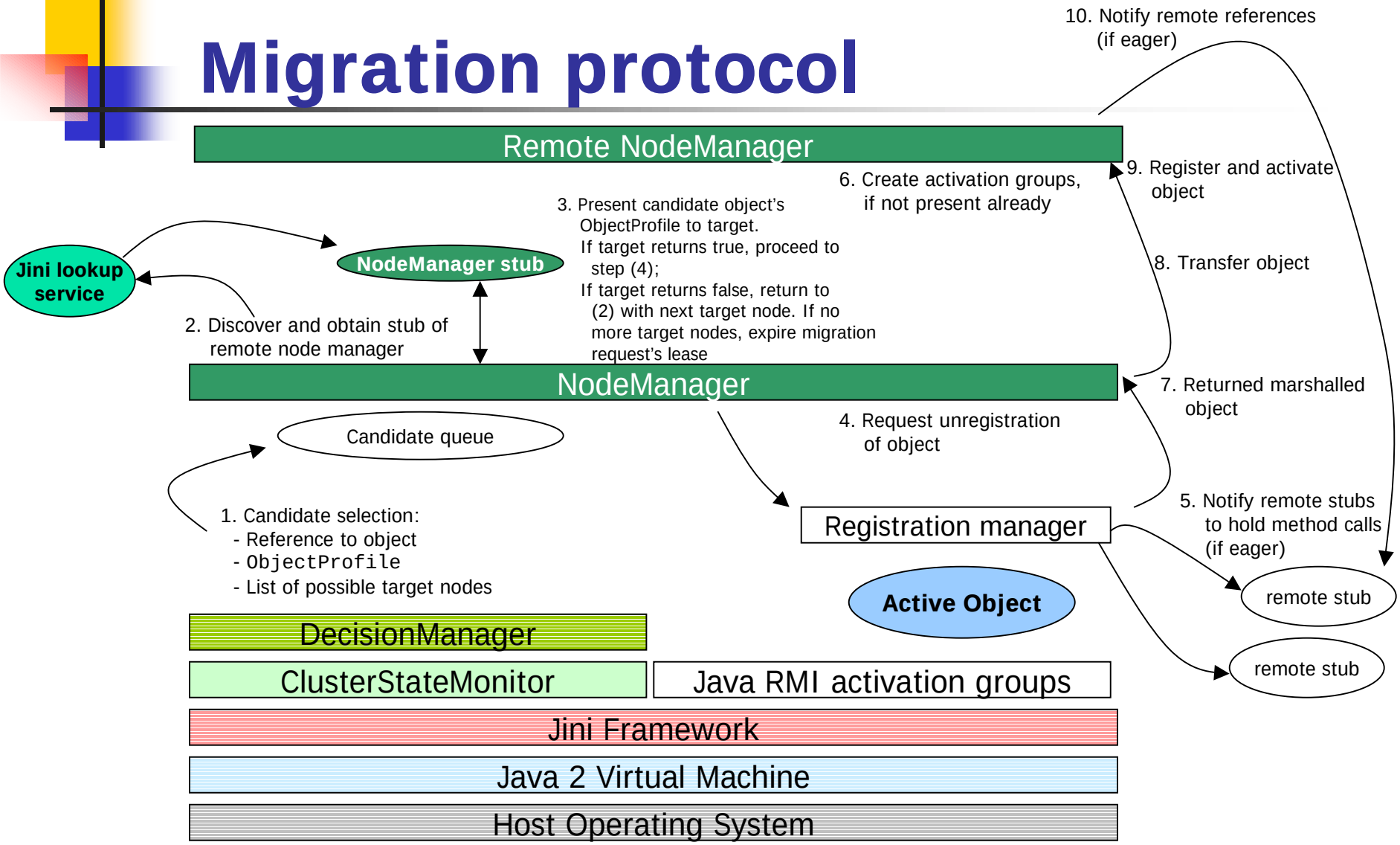
- Negotiates an object's transfer with another node's **`NodeManager`**
- A Jini service
- An RMI Remote object
  - It's remote interface can be invoked from remote virtual machines
- Registers incoming objects with activation system's **`RegistrationManager`**

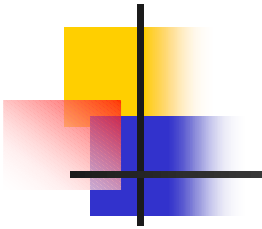


NodeManager, cont'd

- Registers with, and discovered, via Jini lookup services
- Maintains a queue of migration candidates
- Drives the object migration protocol

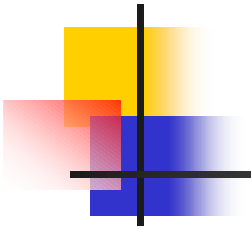
Migration protocol





Migration protocol, cont'd

- Consensus of two nodes required for migration
- Request is leased
- Any steps can fail: No migration takes place then
- Three stages:
  - Discovery: Discovery and contact remote node
  - Negotiation: Is the node willing to take on object?
  - Transfer: Ship the marshalled object to new node
- Eager and lazy methods (eager does not scale)



Future directions

- Better system information needed for Java VM
  - Java Real Time VM may help
- Notion of `Process` in Java
- Adaptive and predictive algorithms?
- Management tools
- Registration service: Single view of cluster when registering new service
- Deploy in real-world application domains (6-9 months)

Resources

- USC Database Systems Laboratory

  http://dblab.usc.edu
- Jini Community Web site:

  http://www.jini.org