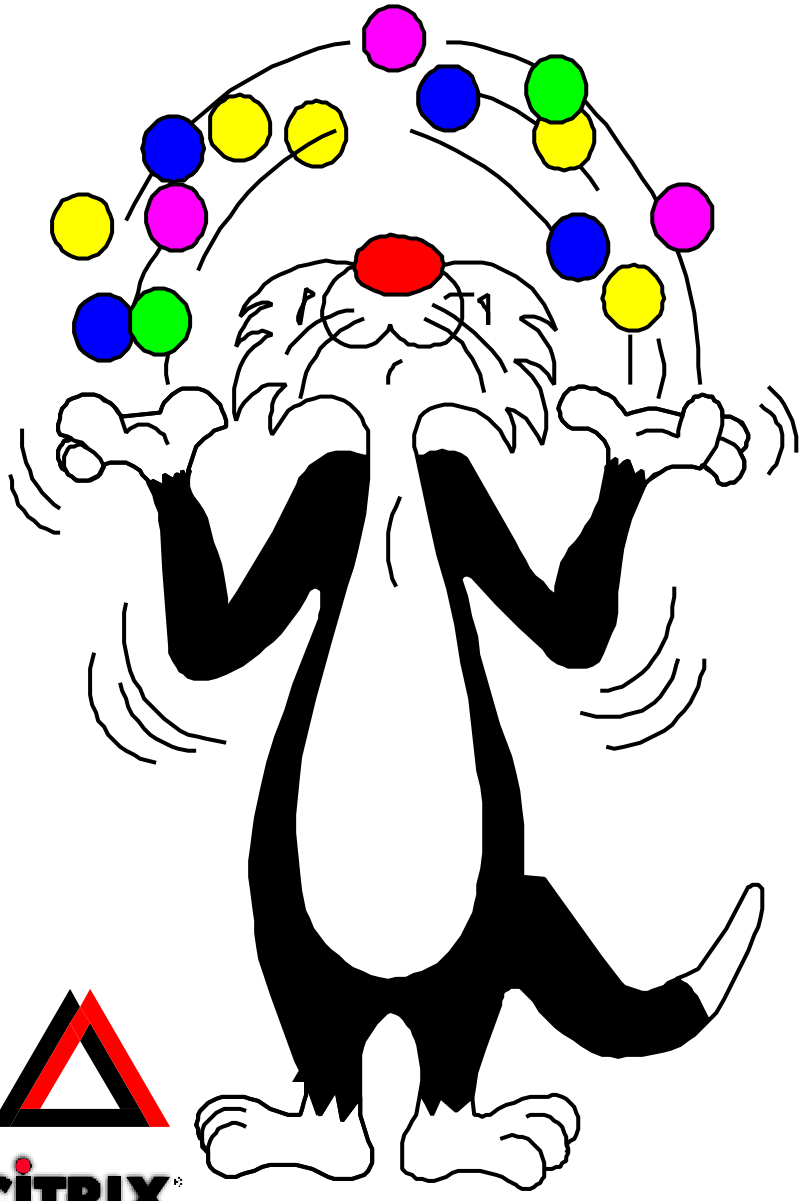
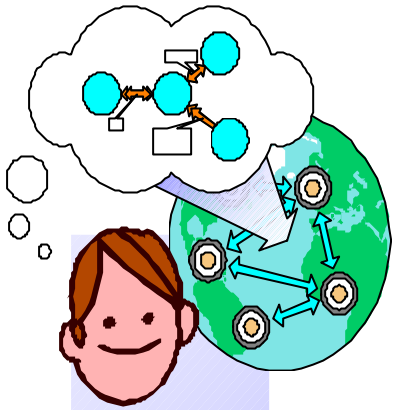


Mobile Java Objects

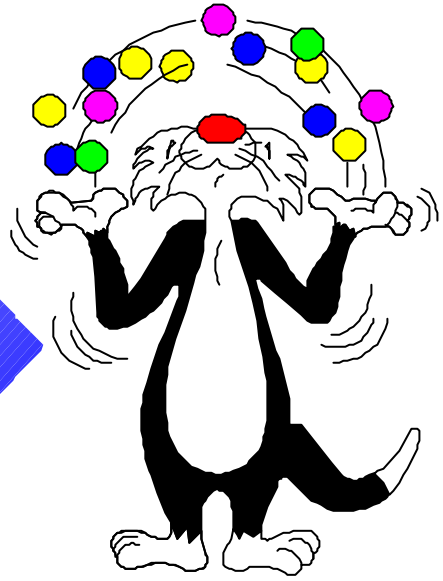
Richard Hayton

APM Ltd

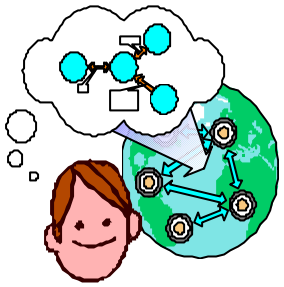
a subsidiary of Citrix Inc



Flexible binding
Flexible deployment



Mobile objects
for mobile agents



Method Invocation

ODP Distribution Transparencies

- Access
- Location

Access objects regardless of object or client location

- Migration
- Relocation

Allow the object and client to move

- Persistence
- Failure

Long lived, failure tolerant objects

- Transaction

Consistent, concurrent access

- Replication

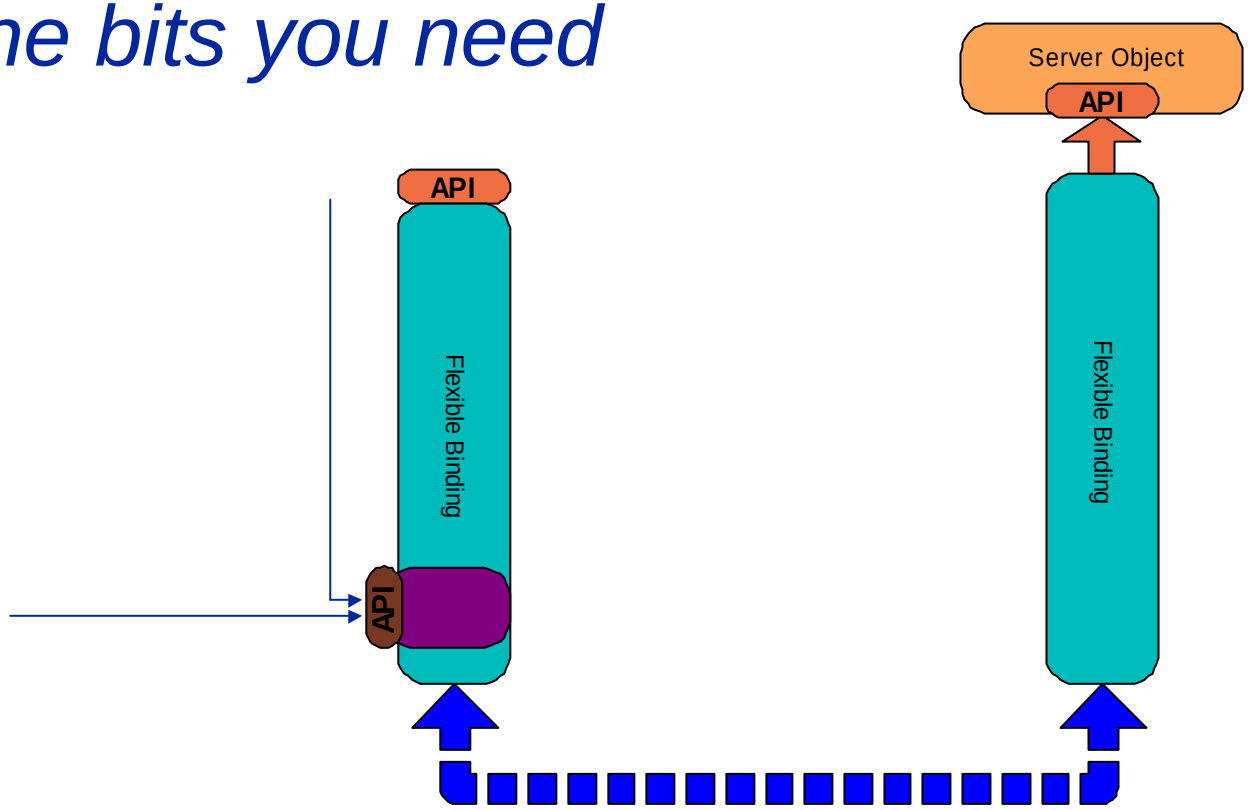
For scale, performance, availability

- Security

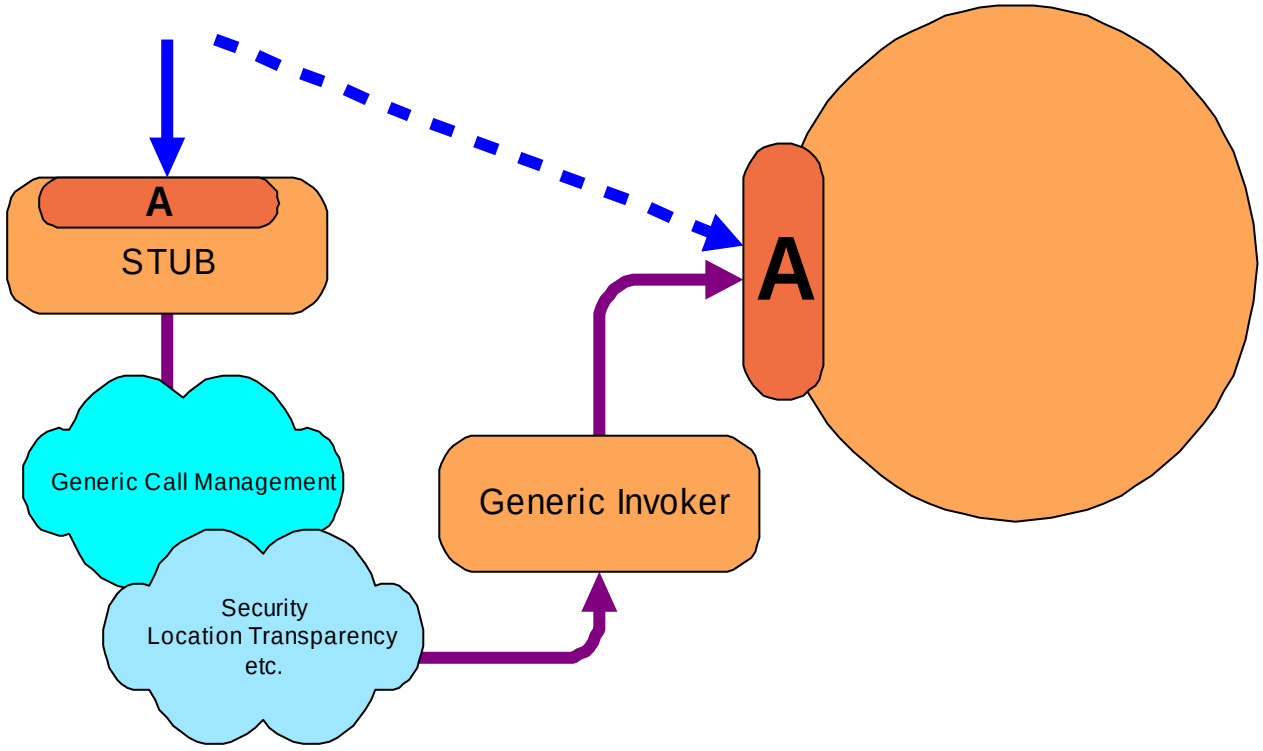
Control and audit access

Our Approach - Flexible Binding

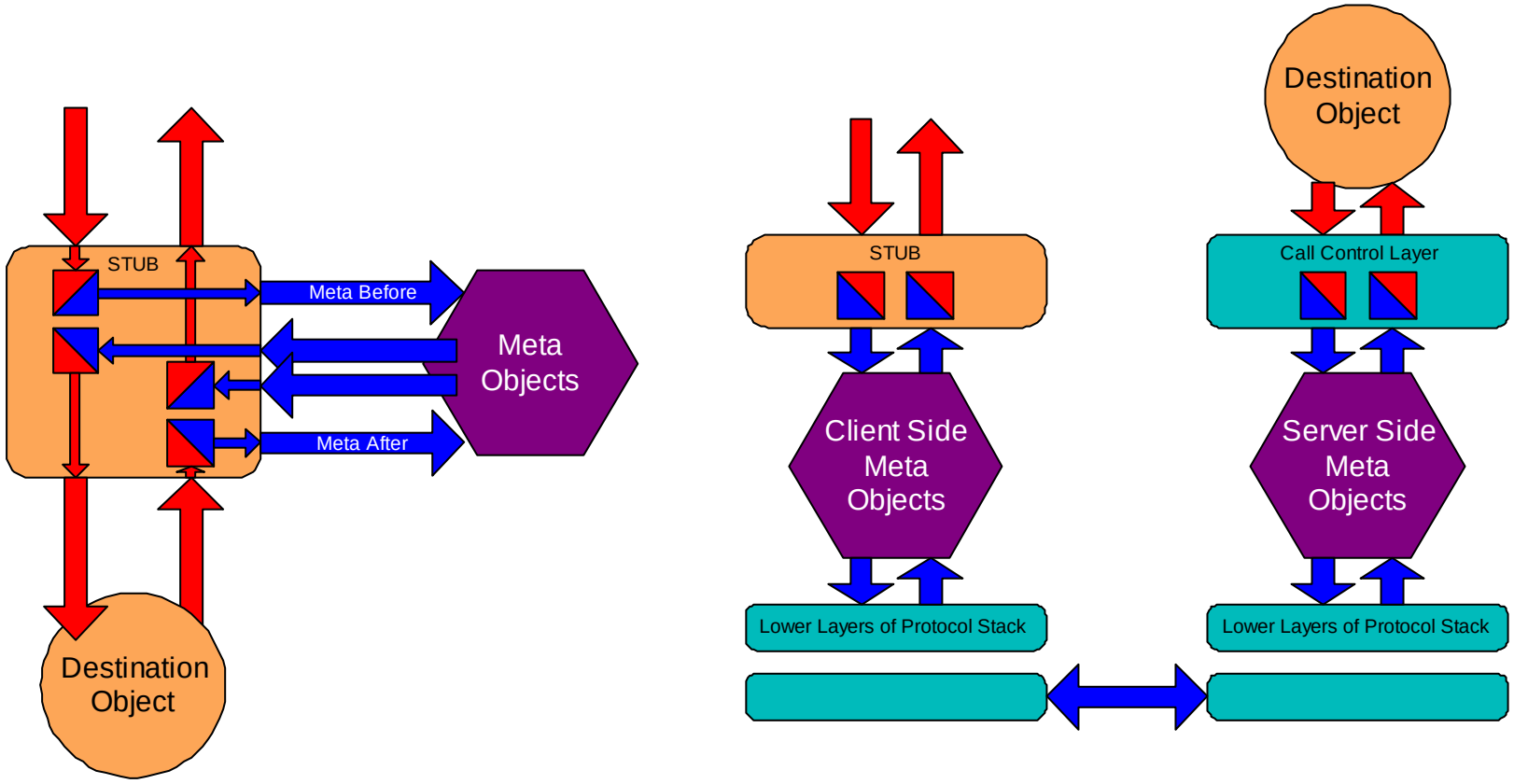
- API remains unchanged
- New external control APIs
- *...Just the bits you need*



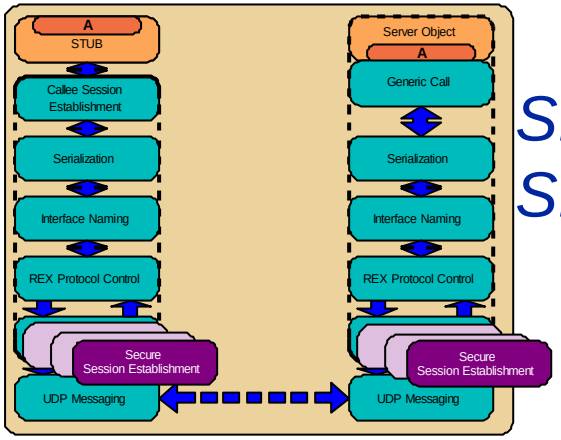
JAVA allows Generic Communication



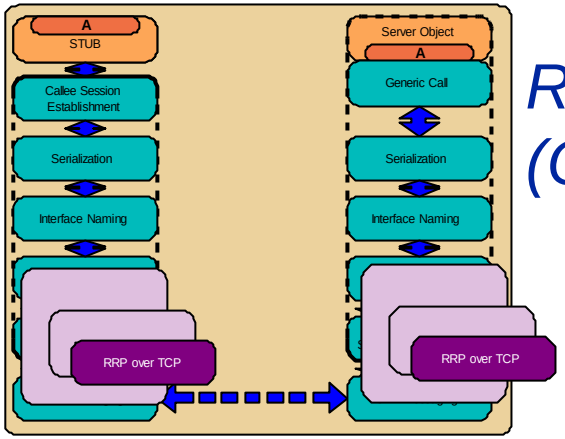
FlexiNet Reflection



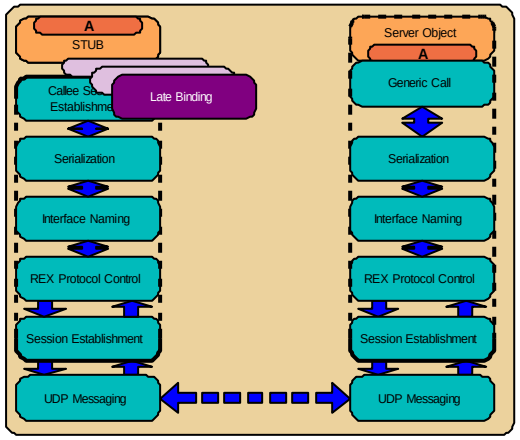
Reflective Remote Invocation Stacks



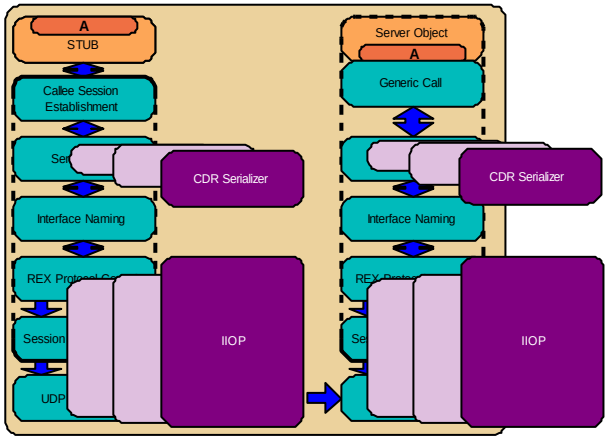
SECURE SESSIONS



RRP Transport (Optimized TCP)



LATE BINDING

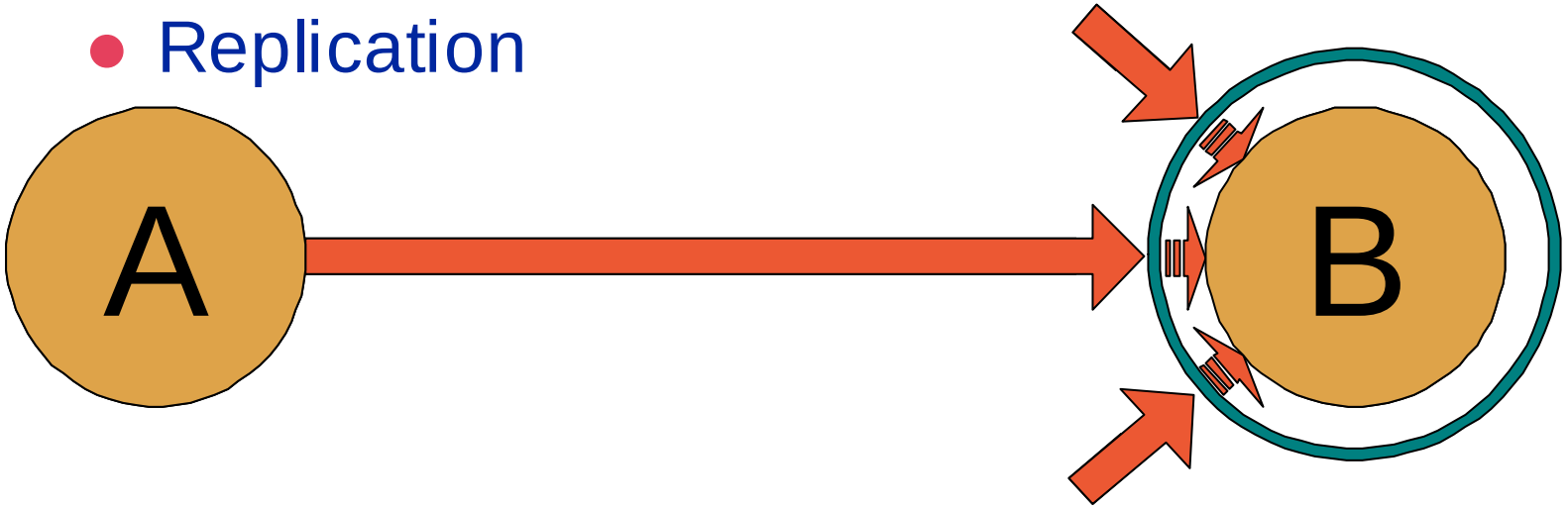


IIOP TRANSPORT

ODP Distribution Transparencies

- Migration
- Relocation
- Persistence
- Transaction
- Replication

Encapsulate to control behaviour = ODP Clusters

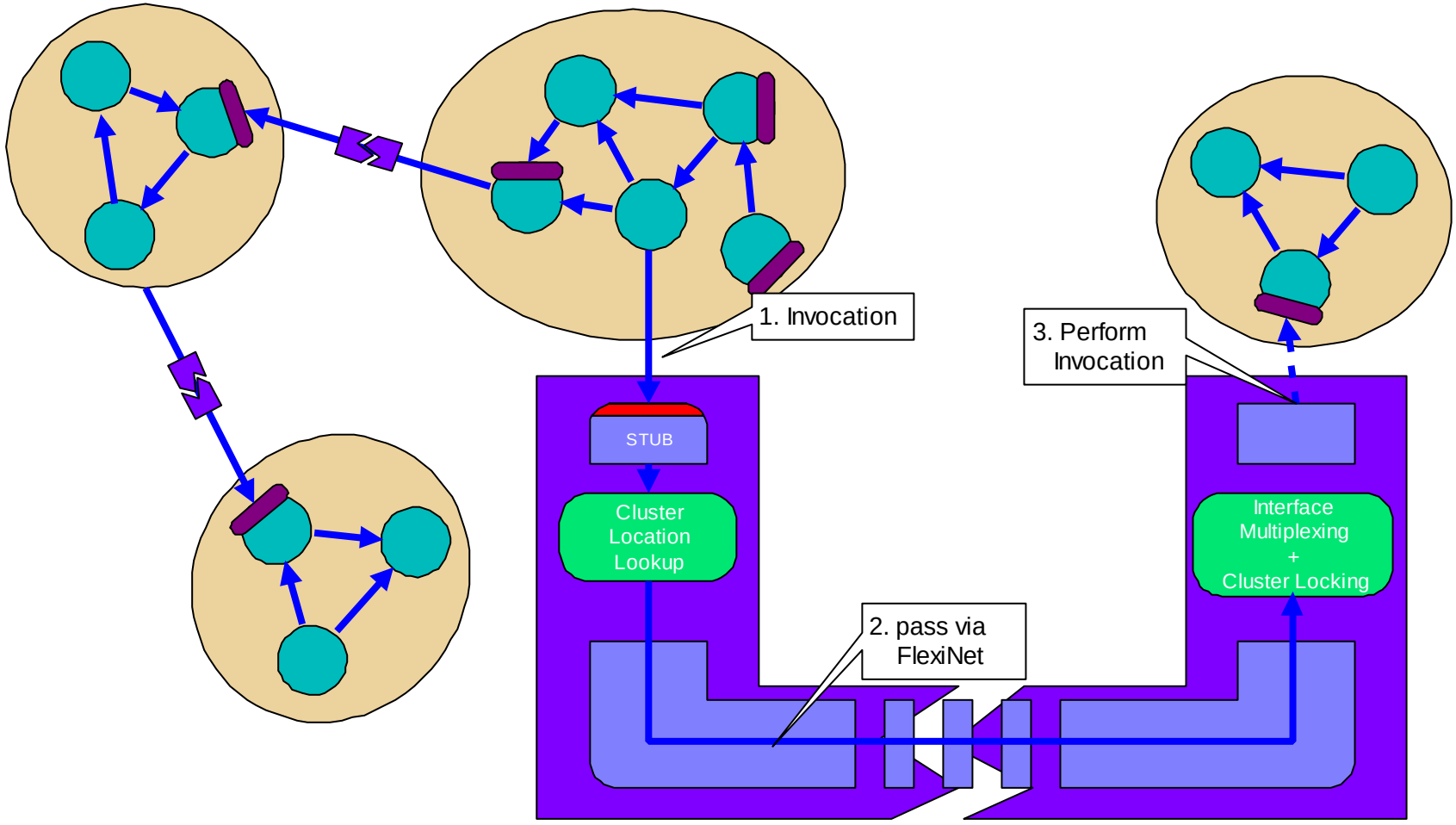


Strong Encapsulation

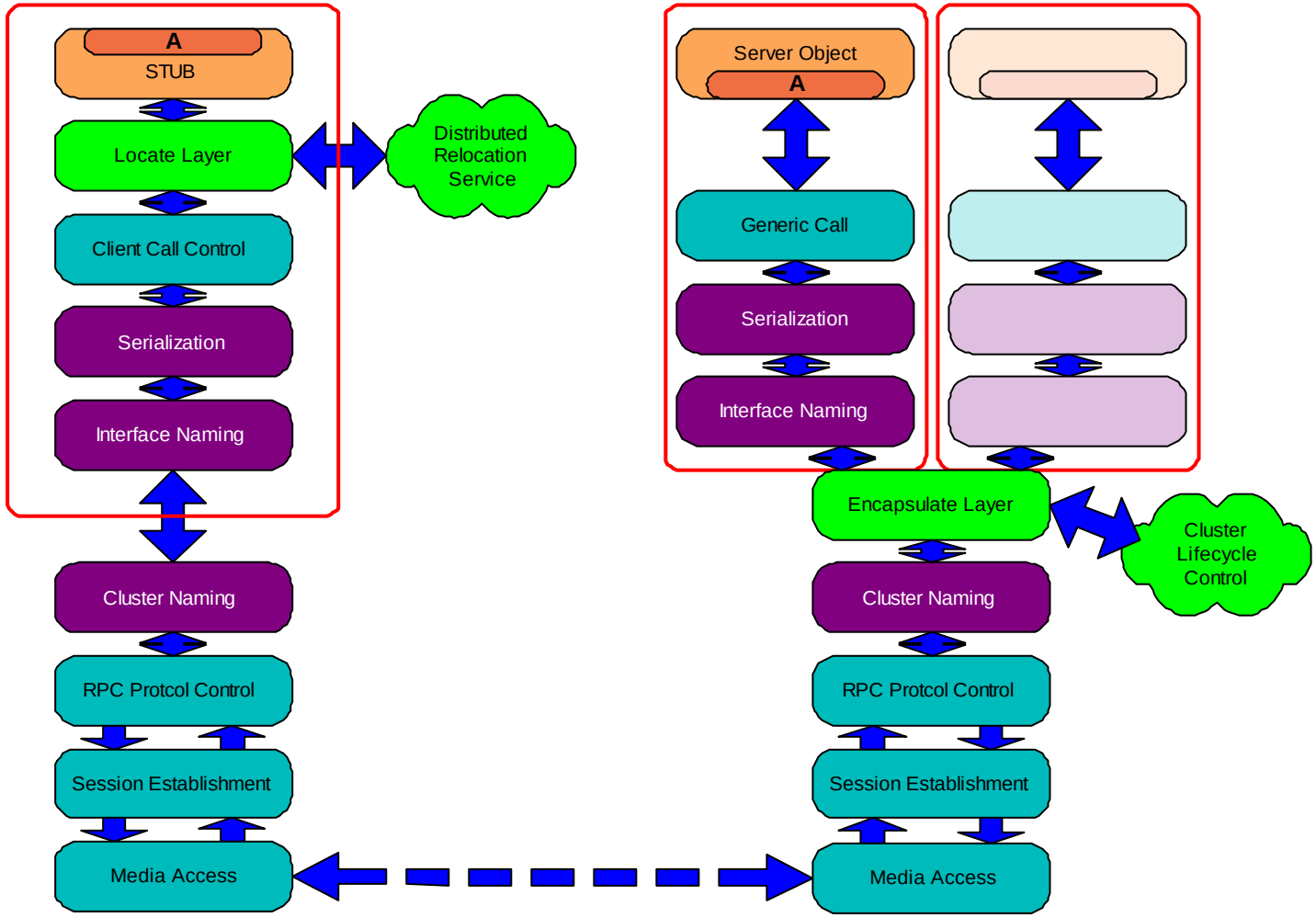
- We use strong encapsulation to keep clusters apart
  - Objects are always passed by copying
  - Interface references are passed by value
  - No objects are shared between clusters

- Strong encapsulation supports “virtual processes”
  - De-couple Threads to manage control flow in clusters
  - Separate class name spaces
  - Separate security managers and security policies

Communication between Clusters



Mobility Binder

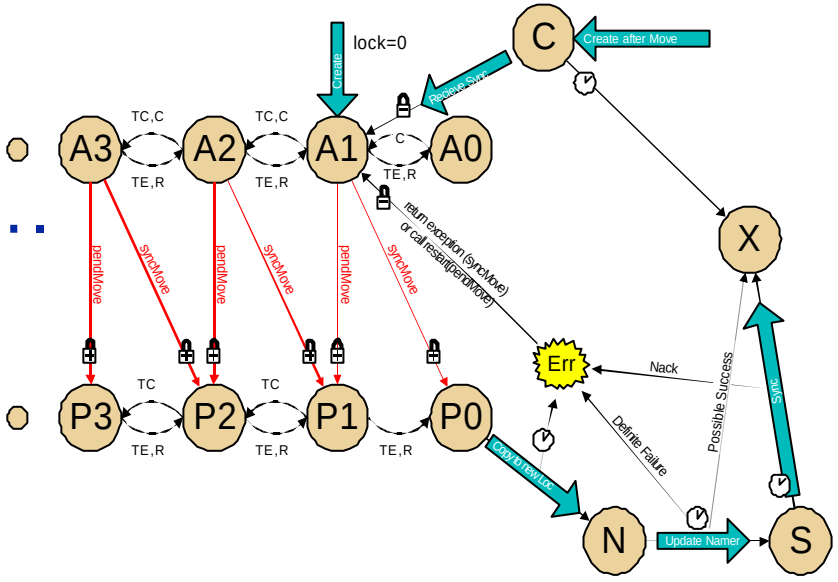


What else is required for mobility?

- Cluster must be passed by value
  - consistent state
  - atomic move
- Rebind references to the cluster
  - indirected names
  - scaling and performance issues
- Control
  - mobility API

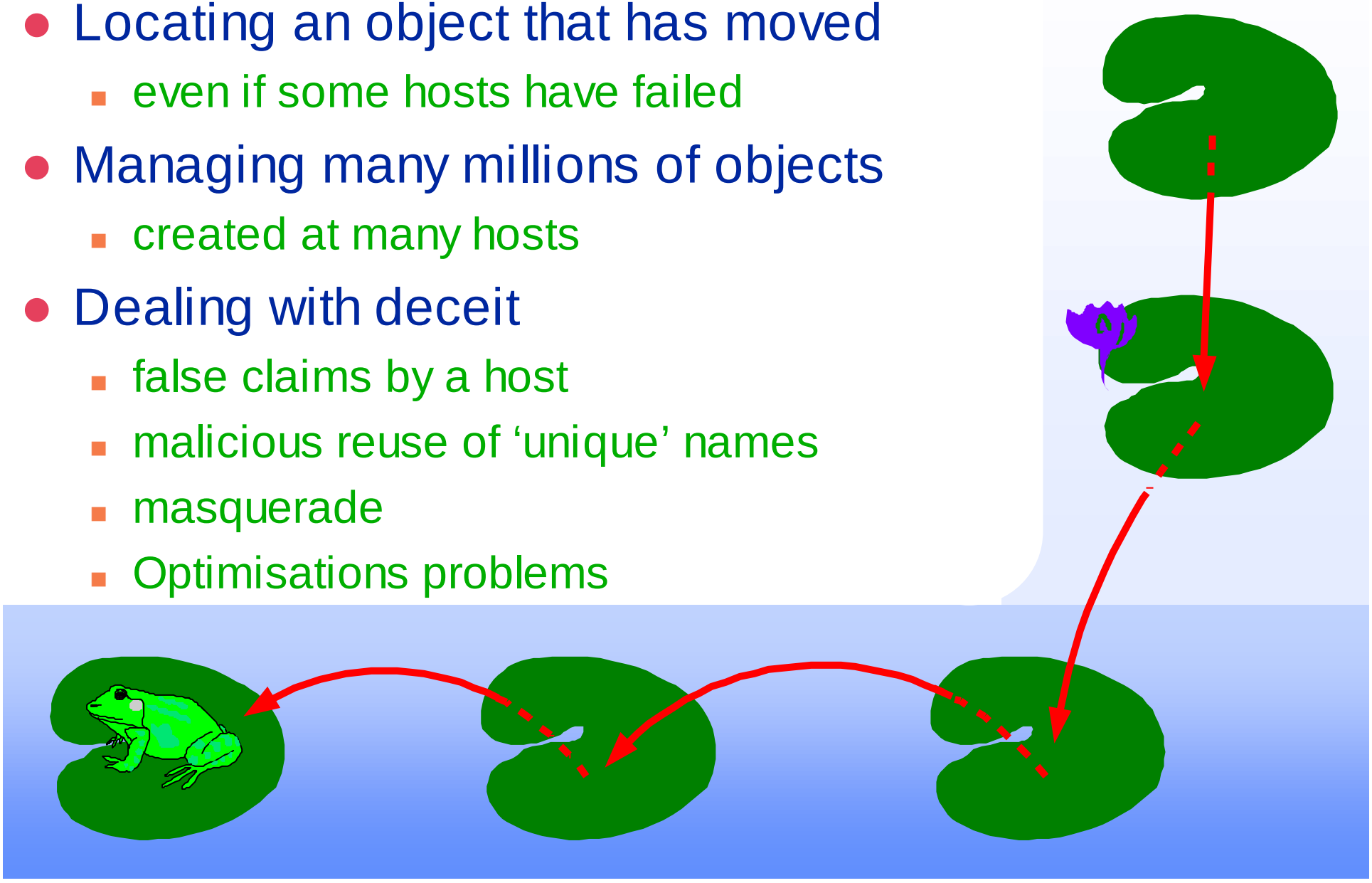
Ensuring Consistent & Atomic Moves

- Track threads within the cluster
  - Block threads wishing to enter
  - Wait until there are no active threads
- Two phase commit
  - ensures agreement about object location
- See paper for details….



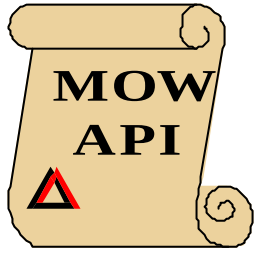
Locating a moved cluster

- Locating an object that has moved
  - even if some hosts have failed
- Managing many millions of objects
  - created at many hosts
- Dealing with deceit
  - false claims by a host
  - malicious reuse of ‘unique’ names
  - masquerade
  - Optimisations problems



Mobility API

```
public class MobileObject extends Cluster
{
 void pendMove(Place dest)  throws MoveFailed;
 void syncMove(Place dest)  throws MoveFailed;
 Object copy(Place dest)    throws MoveFailed;
 abstract Iface init(...) throws InstantiationException;
 abstract void restart(Exception e);
}
public interface Place extends Capsule
{
   public Iface newCluster(Class cls,Object[] args)
      throws InstantiationException;
}
```



Are Mobile Agents just mobile objects?

- **NO!** …. even if we ignore the ‘intelligence’ issue
- Agents form a community of separate applications, not just parts of a single application
  - different code bases
    - What if two people use the same class name?
    - What if two people use different class versions?
  - complex trust relationships
    - Will I run your agent written using his library?
    - Did he modify your agent before it got to me?
  - no global coordination
    - can assume very little

Summary

- Java is great for building middleware
  - code generation on the fly, dynamic class loading, OO class management, introspection, strong typing,…..
- Java is not perfect
  - reflection is limited (but better in JDK 1.2)
  - Sun doesn’t play by the rules (RMI cheats)
- What about mobile objects?
  - Mobility is just one more abstraction
    - in itself, it was straightforward
  - to make it useful we had to tackle many hard problems
    - trust, reliability, scale, code evolution, heterogeneity

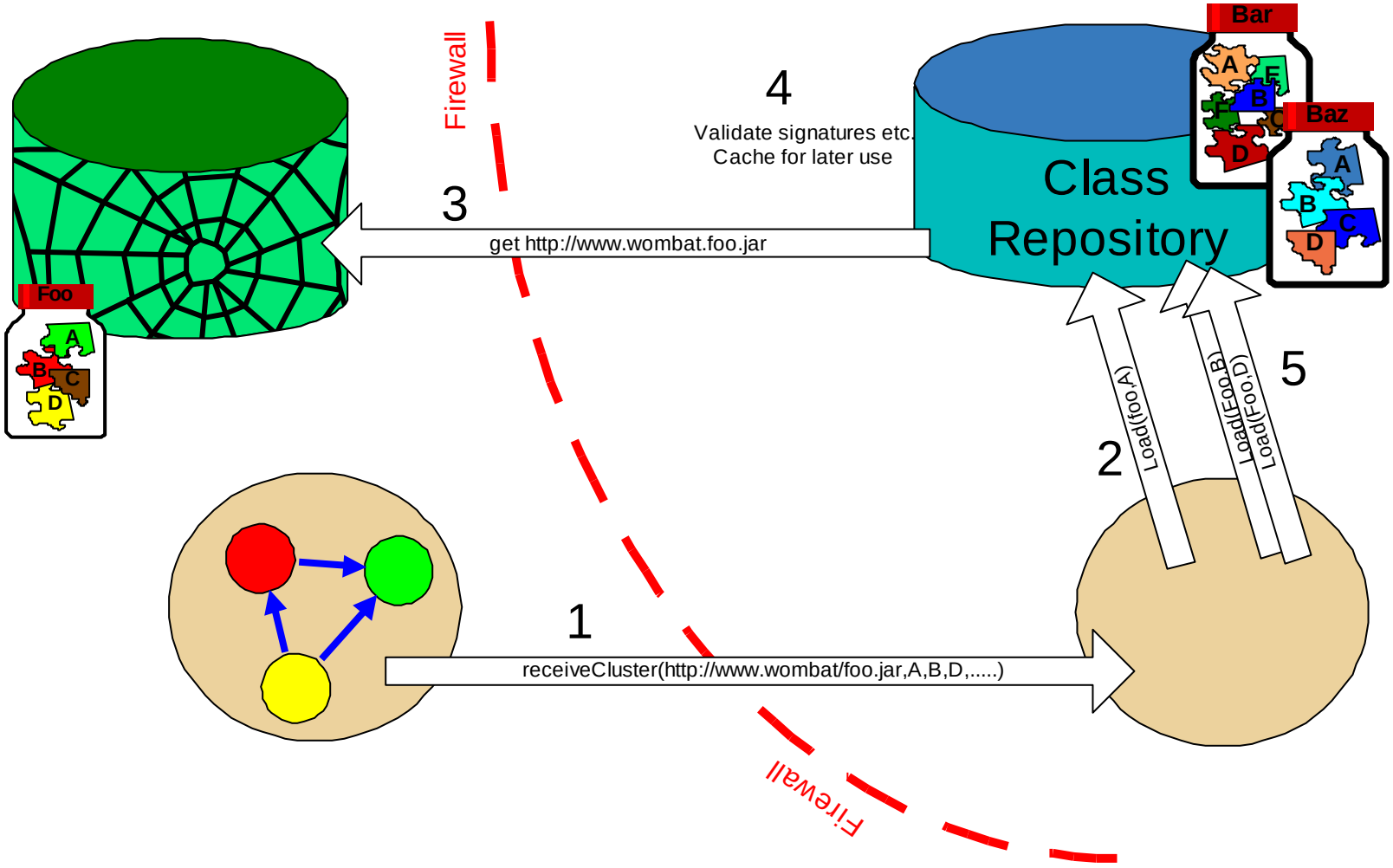
Class Loading Issues

- Where do we load classes from?
  - There is no single ‘code base’ for a global agent system
  - The previous agent location may not be suitable
- Do we trust the class?
  - Who wrote the code?
  - Has it been modified?
- Naming the class
  - Have we already loaded a class with the same name?
  - Is there more than one version of the class out there?

Class Loading Issues

- Where do we load classes from?
  - A local class repository cache (~WebCache)
- Do we trust the class?
  - HARD JDK 1.2 certificates
    Agent integrity statements
- Naming the class
  - Java's naming scheme is inflexible
  - ….multiple class loaders provide a get out

Resolving a set of remote classes



Mobile Security Issues

- Host Integrity
  - Cluster abstraction reduces impact of malicious agents
- Cluster Integrity & Confidentiality
  - Cryptographic techniques can determine if a cluster has been inappropriately modified and hide secrets
- Cluster Authority
  - integrity and confidentiality allows the carrying of passwords
  - replay is still an issue
- Access control and secure communications
  - standard techniques