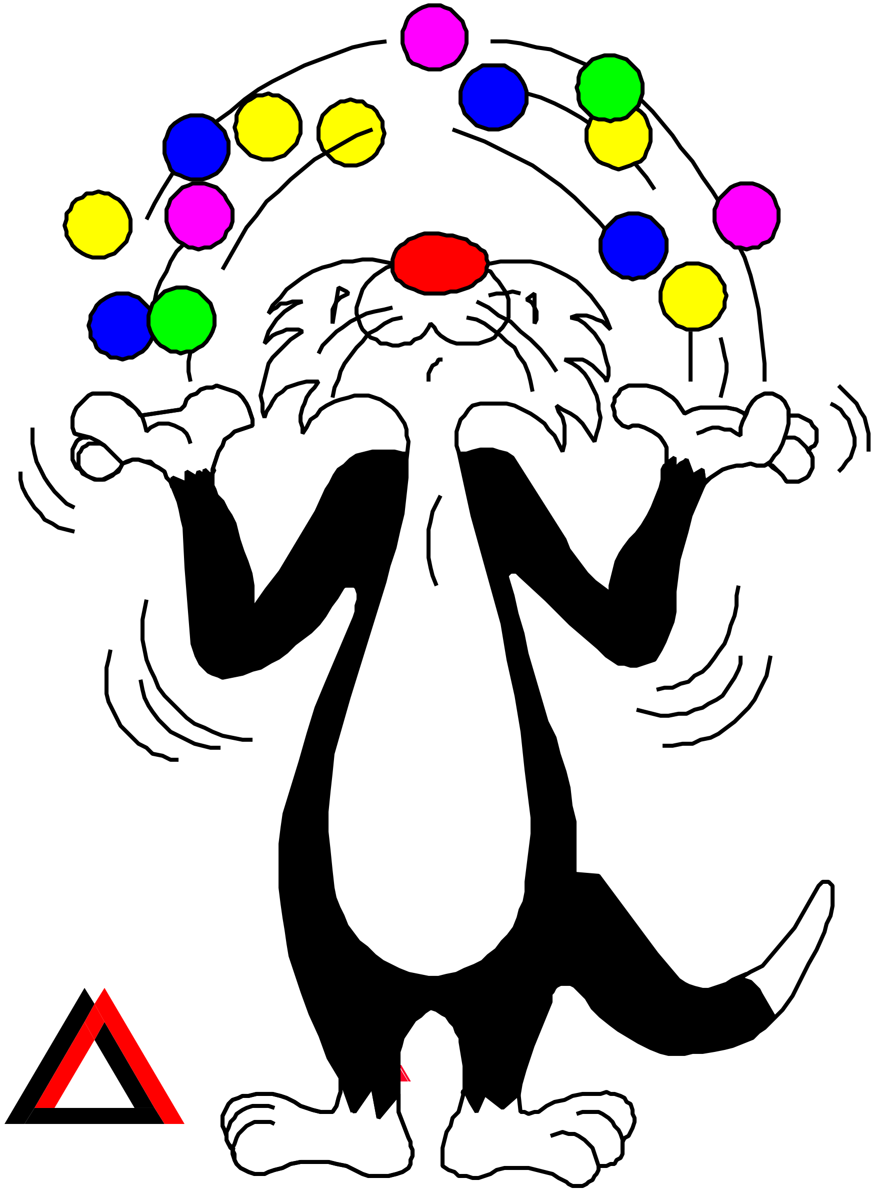


The Mobile Object Workbench

Richard Hayton &

Douglas Donaldson

CEC Technical Review

of FollowMe

10th June 1998

The Mobile Object Workbench

- What is it?
  - Adding mobility to distributed computing
  - Keep distributed computing ideals
    - “Sea of objects”
    - Well defined interfaces
    - Transparency
  - Add the ability to move an object from place to place
- It isn’t
  - An agent architecture - although it forms the basis of one
  - About deciding if, when and where to move

The MOW and FlexiNet

MOW built on top of FlexiNet Open ORB Framework:

- Allows ‘slot in’ middleware enhancements
  - new protocols
  - new abstractions
- Reflection and introspection
  - to keep components modular
  - to allow configuration
- Provides transparent binding
  - For local or remote interconnection

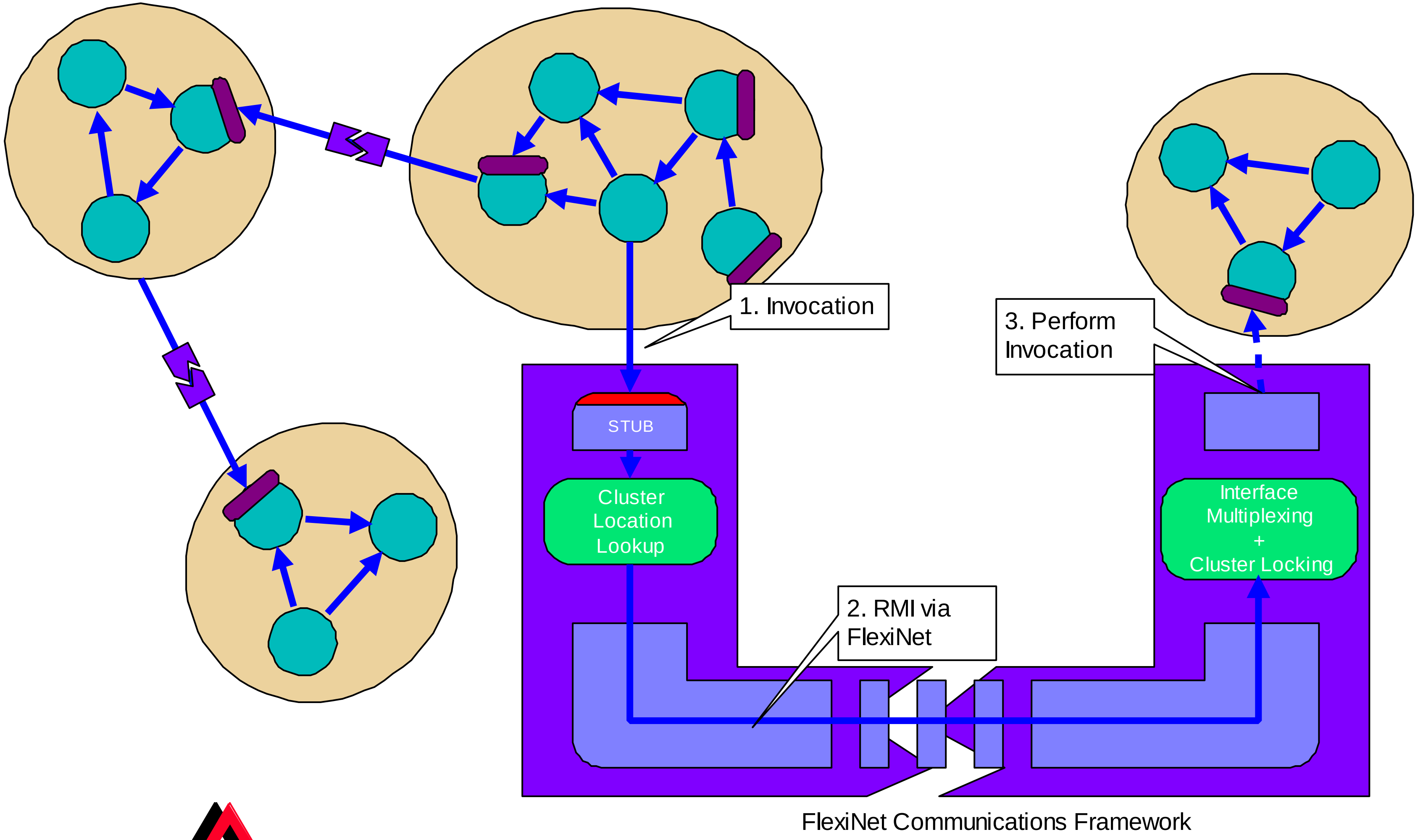
MOW - Basic Concepts

- There are two types of entity, MobileObjects and Places
  - Objects exists “within” Places
  - MobileObjects may move between Places
- Objects are autonomous, they cannot be forced to move, but can be destroyed by their current Place
  - A particular move may be vetoed by either the source or destination Place
- When an Object moves it takes with it its current state

Clusters

- MobileObjects will typically consist of a number of objects
  - These objects should be managed at a unit
  - We introduce the concept of a *cluster* to represent this
- Mobile Clusters
  - When a cluster moves we must fix up references
  - We cannot break language level references
- Approach
  - Tightly bind objects within a cluster (Java references)
  - Loosely bind clusters (using comms. framework)

Clusters


1. Invocation
3. Perform Invocation
STUB
Cluster Location Lookup
Interface Multiplexing + Cluster Locking
2. RMI via FlexiNet
FlexiNet Communications Framework

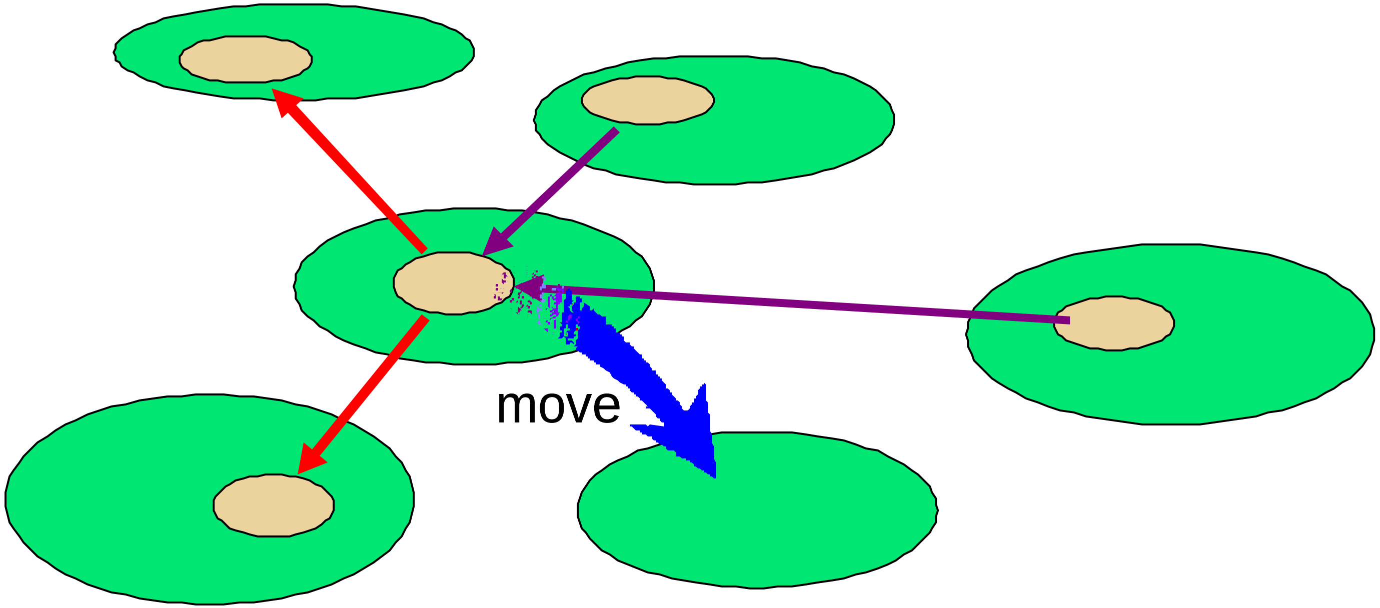
Strong Encapsulation

- We use strong encapsulation to keep clusters separate
  - Objects are always passed by copying
  - Interface references are passed by value
  - No objects are shared between clusters
- De-couple Threads to manage control flow in clusters
  - Each cluster has a thread group
  - Clusters cannot block other clusters
  - MOW can count the number of threads in a cluster
  - MOW can kill all the threads in a cluster
    - In theory - unimplemented in JDK 1.1

MOW Issues

- Unbinding
  - Removing an object from its execution environment
- Movement
  - Moving the object to a new execution environment
- Rebinding
  - For *relocation transparency*
    - Ensuring that references to the object prior to the move now refer to the newly moved object
  - For *migration transparency*
    - Ensuring that references from the object prior to the move still refer to the same objects
- Security

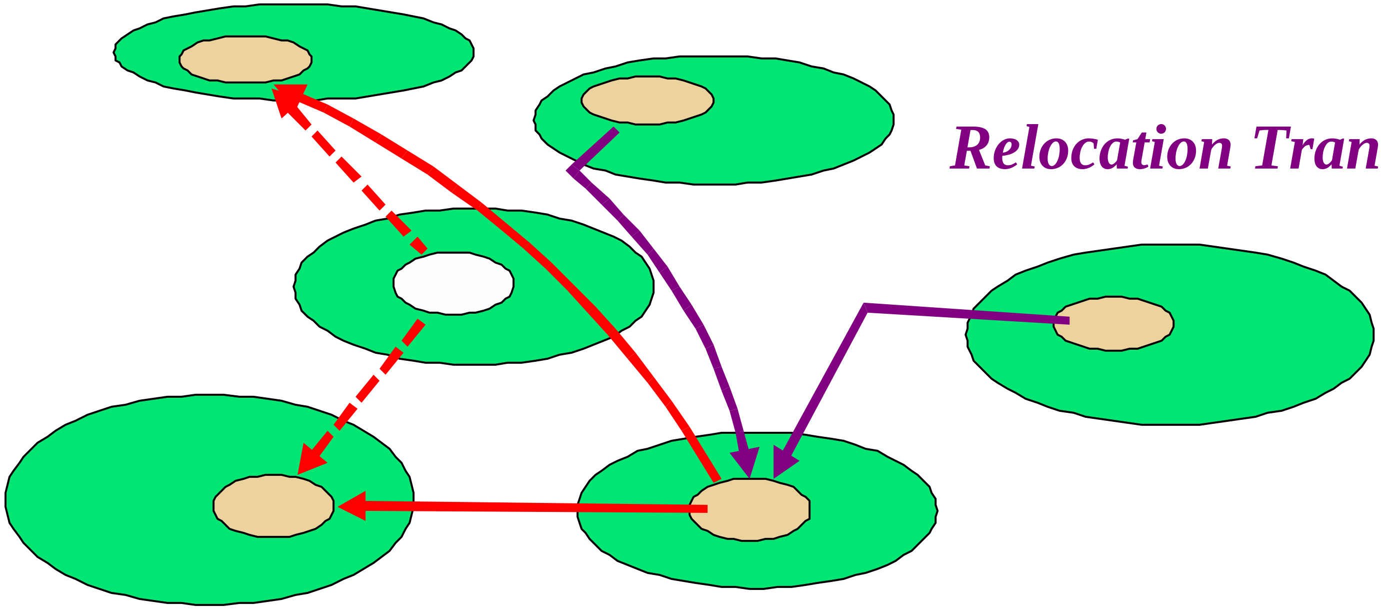
Cluster Mobility



Clusters may move between hosts

References between clusters continue to work.

Maintaining Transparency



Relocation Transparency

Migration Transparency

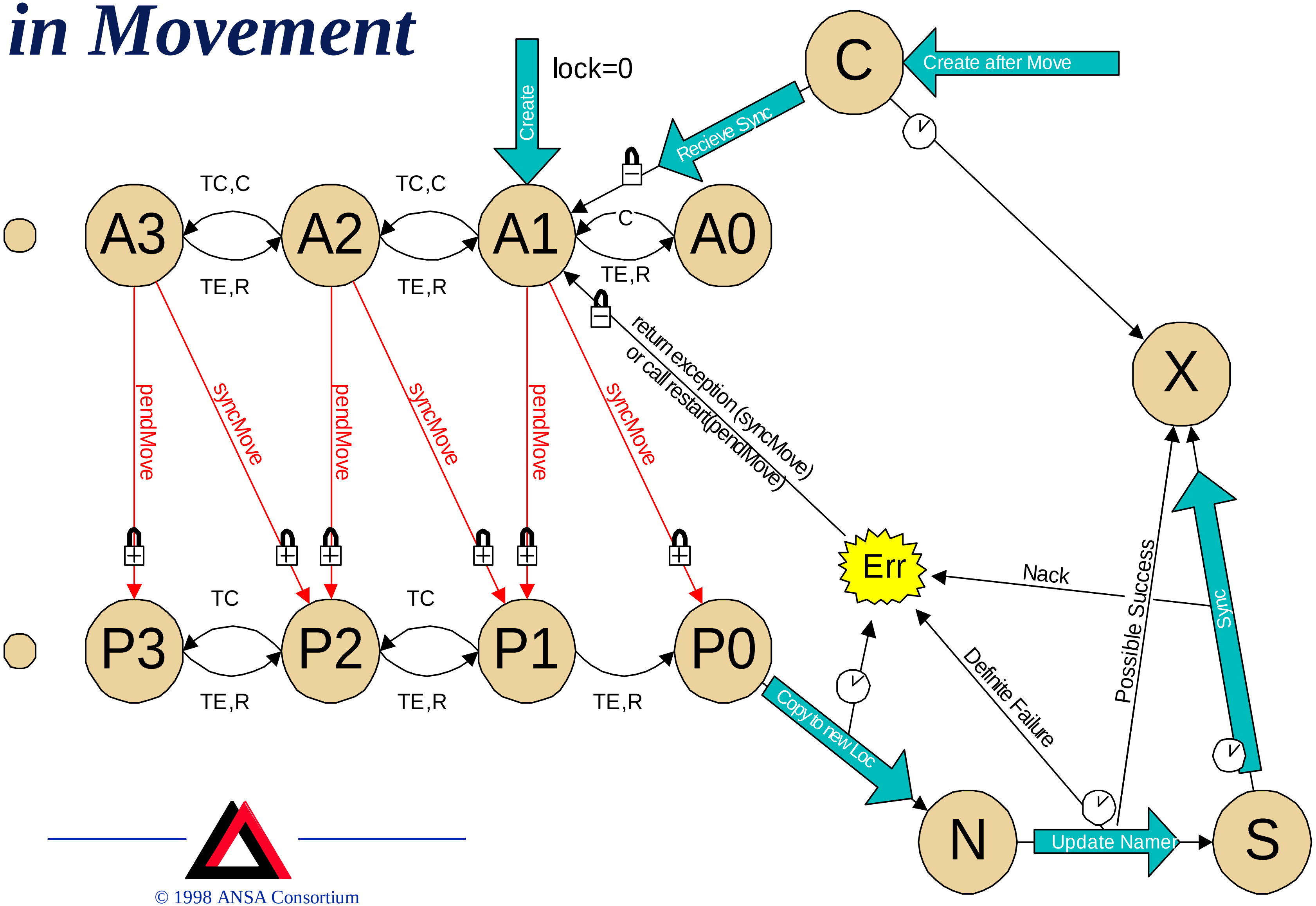
Unbinding

- We must determine what should move
  - One object or a collection? A Cluster.
- We must ensure that after the unbind, there are no references to the unbound object(s)
  - Achieved because Clusters are loosely bound using the communications framework
  - The objects within the Cluster are tightly bound with Java references
- We cannot break language level references

Cluster Movement

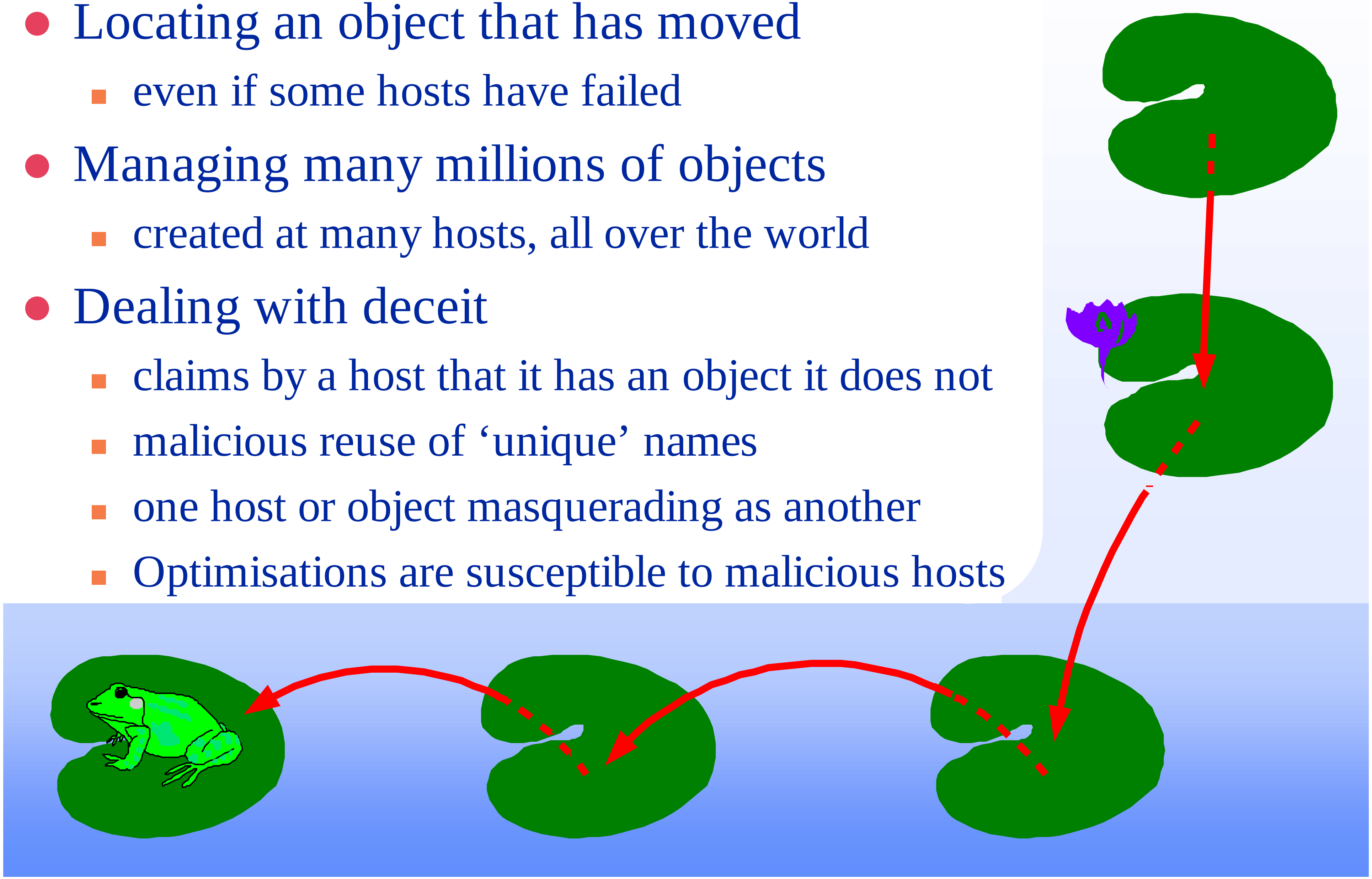
- Is this simply copying an object and discarding the original?
  - Yes, EXCEPT the copy must represent a consistent state
- Only move when:
  - There are no active threads within the cluster
  - This implies there are no calls in progress
- We use locking and thread counting to achieve this

Ensuring Consistency in Movement



Rebinding -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, all over the world
- Dealing with deceit
  - claims by a host that it has an object it does not
  - malicious reuse of 'unique' names
  - one host or object masquerading as another
  - Optimisations are susceptible to malicious hosts

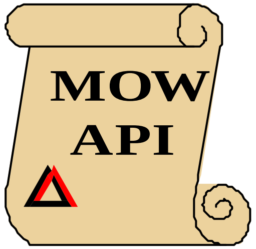


Directory Based Name Resolution

- On cluster creation:
  - choose a directory d but don’t use it yet
  - Name the cluster (d,current address)
- On move
  - update directory d with old address ⇨ new address
- On lookup
  - try the previous address, if it fails contact d

Mobility API

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



Security

- Secure Communication
  - Host to Host
    - Providing a sample implementation of SSL, with infrastructure (key management, certificate management, CA…)
  - Object to Object
    - Requires object identity; assuming public key infrastructure and X509 certificates…
- Objects carrying secrets
  - Untrusted hosts must not be able to decrypt secrets
- Objects must maintain code and data integrity
  - Need to prevent / test for tampering.

Summary So Far

- MOW Release 1.1 Software and Reports delivered
  - Strong encapsulation implemented
  - Movement and copying of Clusters
  - Directory based name relocation service
  - Provides access, location, relocation and migration transparencies
- Tamagotchi demonstration available
- Current effort on
  - Network class loading
  - Scalable relocation service
  - Security implementation

Advanced MOW Issues

- Advanced Name Resolution
  - World-wide naming?
- Class Loading
  - Where does a Place get classes from?
- Security Issues
  - How can mobile objects prove their identity and carry secrets?
- Strong Encapsulation
  - Wrap AWT and other APIs
  - Browser issues

New Name Resolution Scheme

- Designed for a large scale environment with poor reliability and mutual distrust
  - i.e. for FollowMe in a WWW environment

- Implemented as a set of “stages”
  - each is a refinement on the previous stage

- Current status
  - stage one is implemented

Directory Based Name Resolution (repeated)

- On cluster creation:
  - choose a directory d but don't use it yet
  - Name the cluster (d, current address)

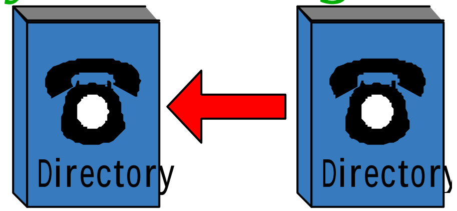
- On move
  - update directory d with old address ⇨ new address
- On lookup
  - try the previous address, if it fails contact d

Analysis

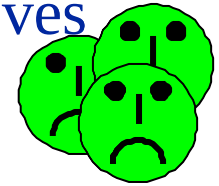
- Security/Integrity
  - High trust in directory
  - Clusters can choose an appropriate directory ☺
  - Hosts cannot fool others into thinking they have a cluster
- Move/Lookup Cost ☺
  - At most two additional calls
  - One may be to a distant host if the directory is ill placed ☹
- Reliability
  - Require access to 1 host out of 1 possible host ☹

Stage Two: Reducing Move/Lookup Cost

When the system decides that a directory is no longer suitable for a particular cluster:



- Pick a more suitable directory d2
  - Update the cluster’s name to (d2, current address)
  - Update the old directory d with (current address ⇨ d2)
    - Tombstoning directories
- Analysis
  - Lookup/Move: 2 calls (directory normally near)
  - Reliability: n+1 hosts out of n+1 after n directory moves

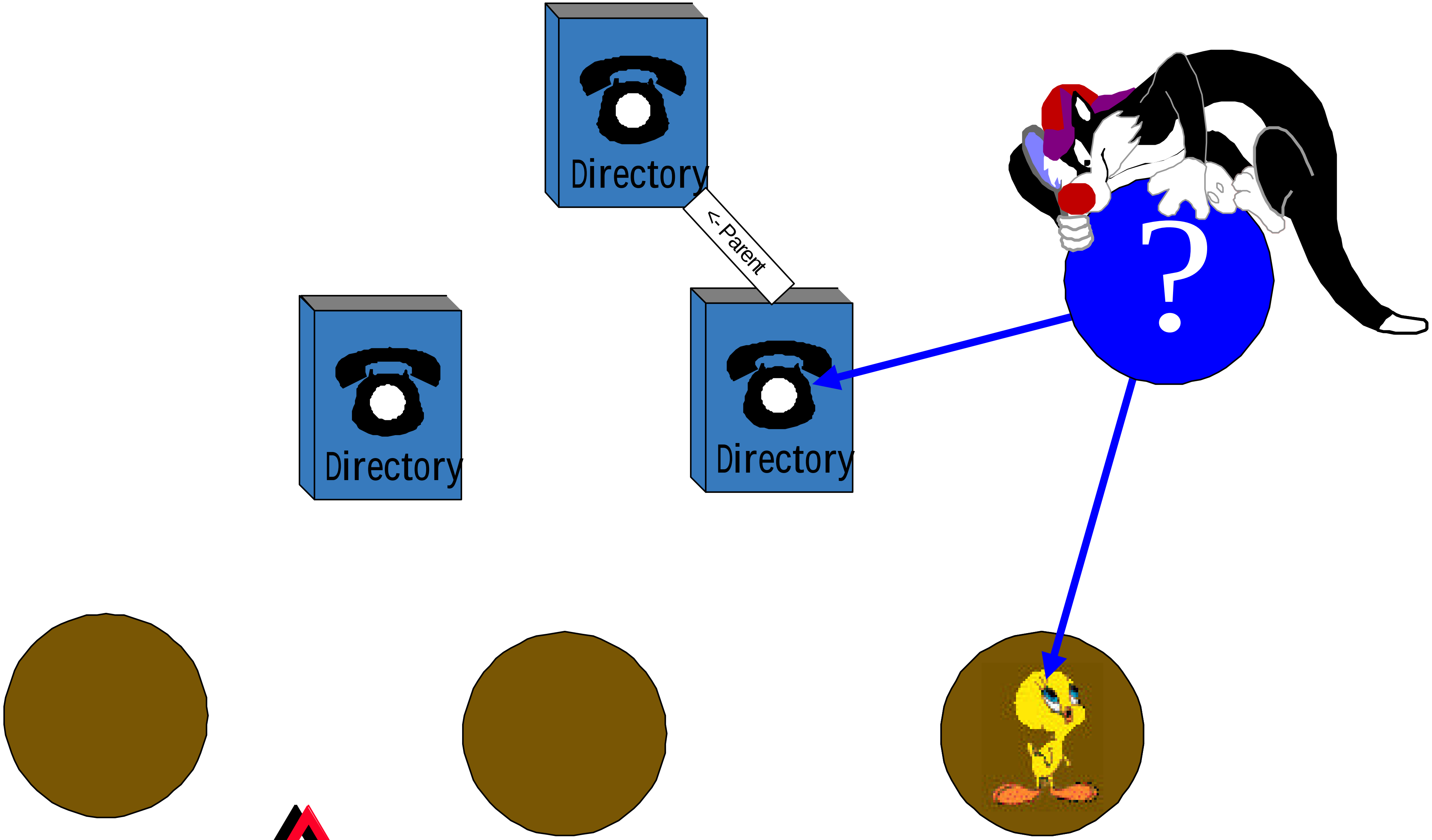


Stage Three: Improving Reliability

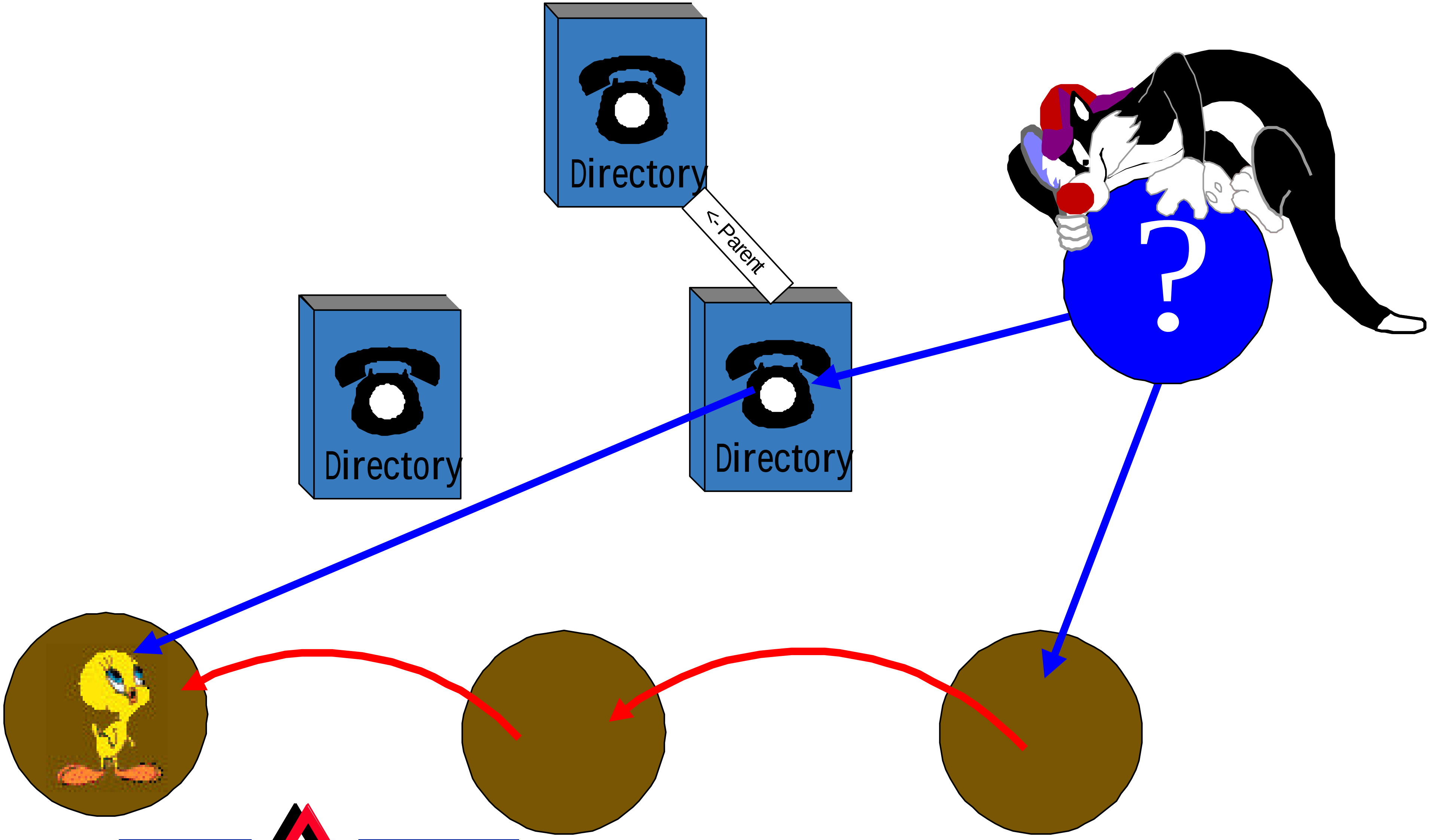
- Each directory is given a well known parent
- A directory may copy any entry to its parent
- If a directory is uncontactable, the parent is asked

- Analysis of reliability:
  - n hosts out of 2n (each tombstone or its parent)
- Analysis of background cost
  - Low - *if we only copy to parent when we create tombstones*

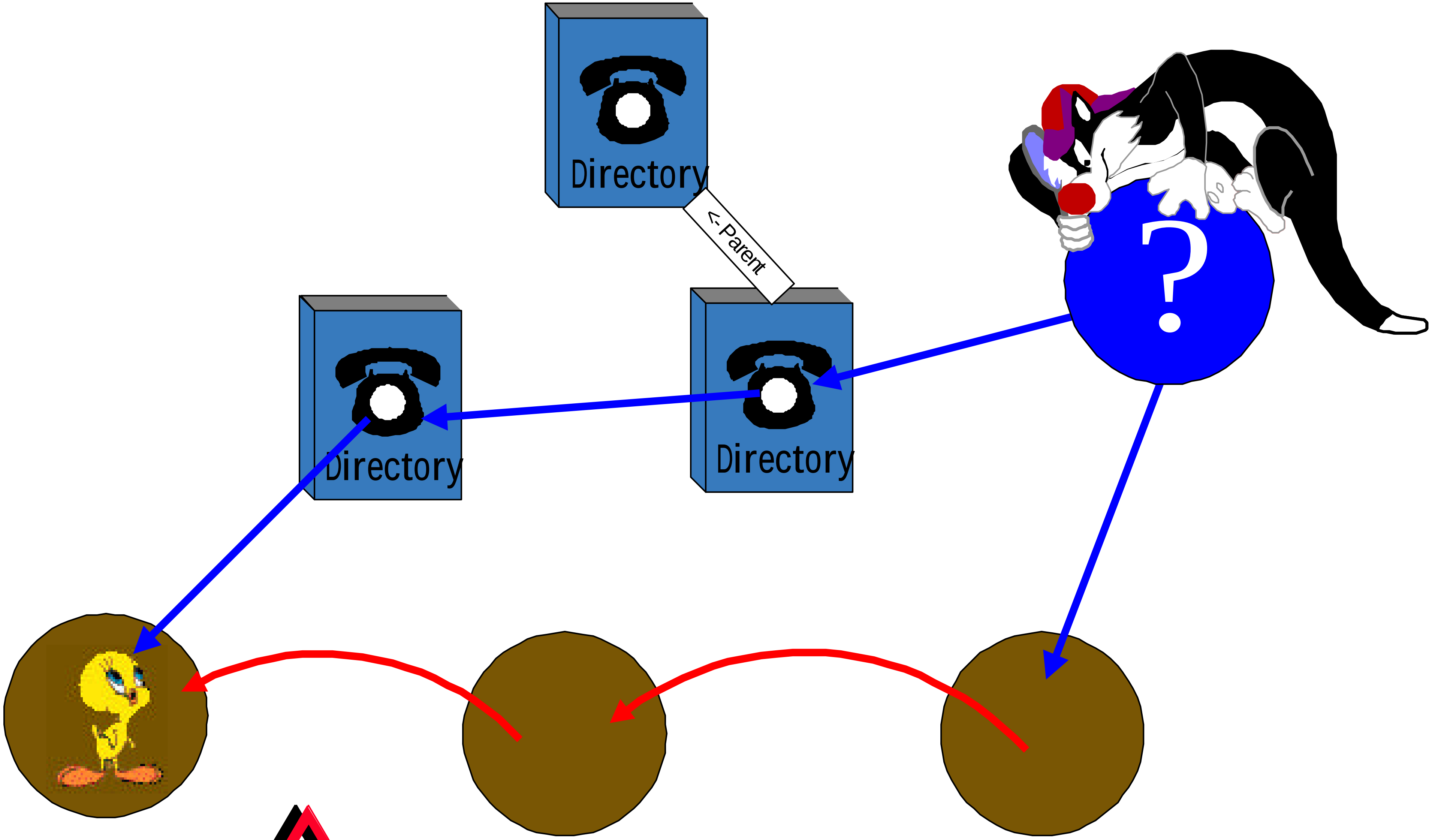
Catch the Birdie…...



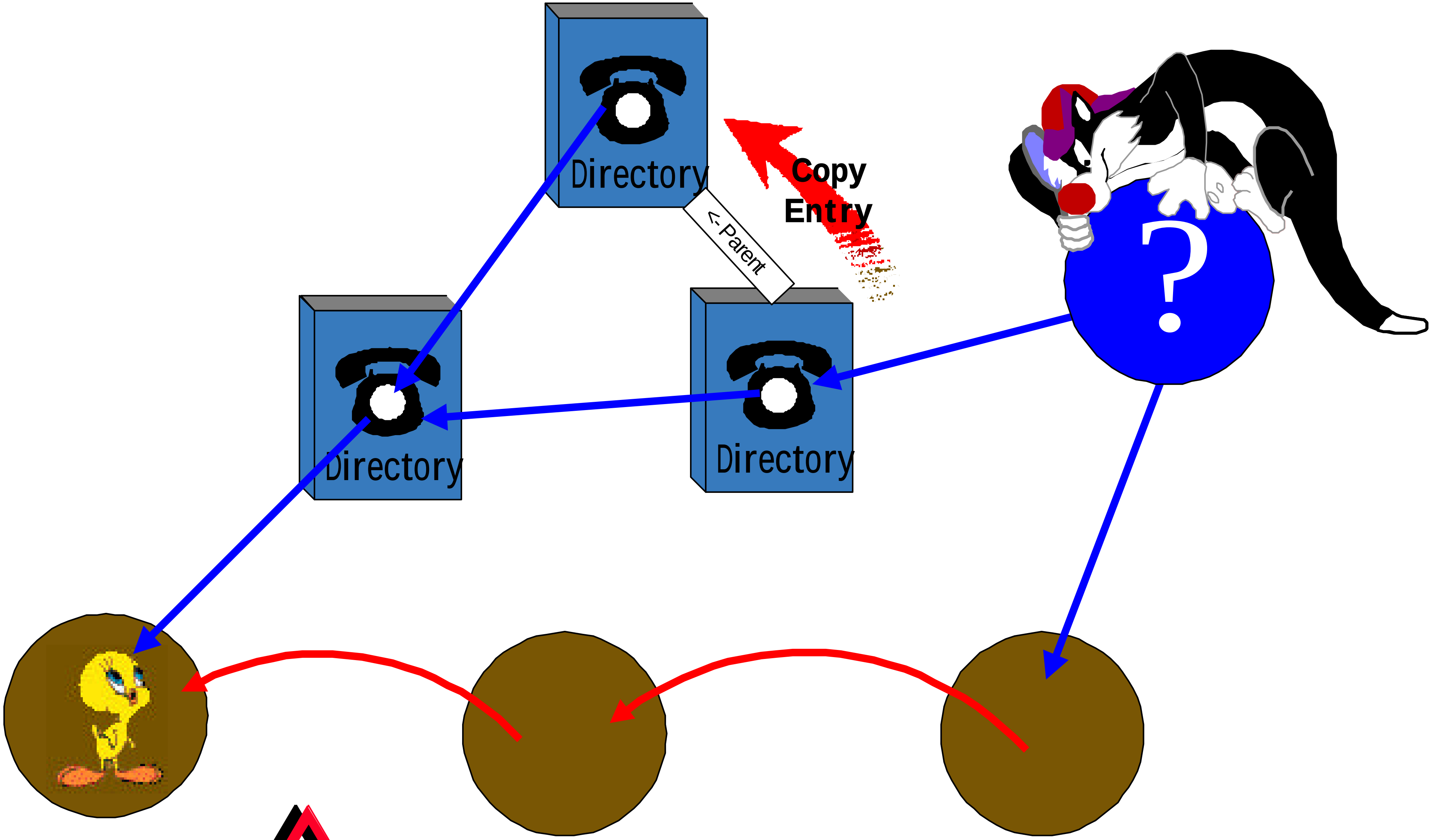
Catch the Birdie……



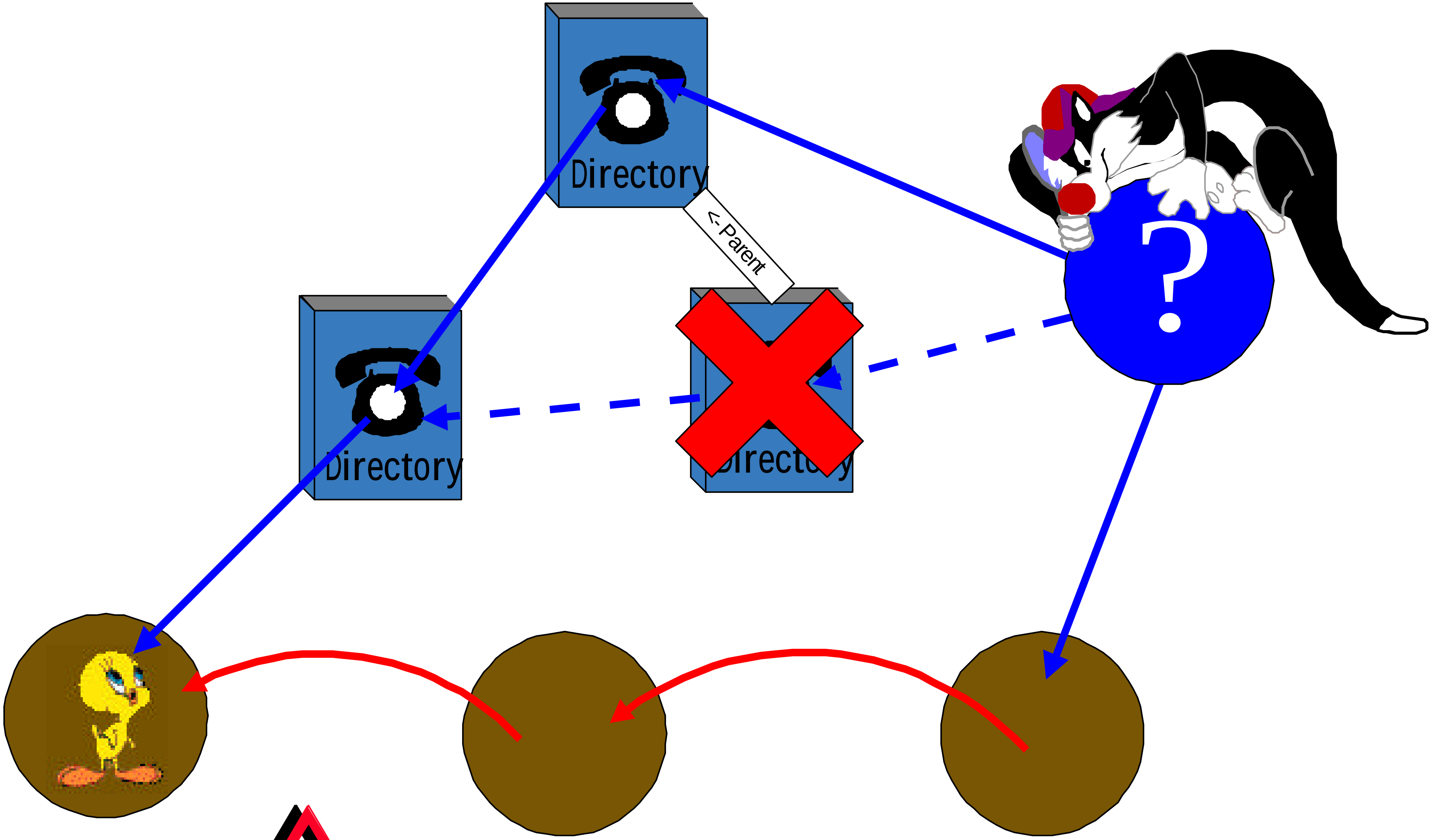
Catch the Birdie…...



Catch the Birdie…...



Catch the Birdie…...



© 1998 ANSA Consortium

Stage Four: Reduce Garbage Accumulation

In the current scheme a directory can never forget an object that has not been deleted, even if it is ‘long gone’

- Solution:
  - A directory may copy an entry to its parent, and delete the local reference
  - When a client requests a lookup of an unknown name, the directory bounces the request to its parent
  - NB. There must be a short chain of parents or invalid names will take a long time to return definite failure on lookup
- Stage Five: mobile places…...

Deployment of Directories

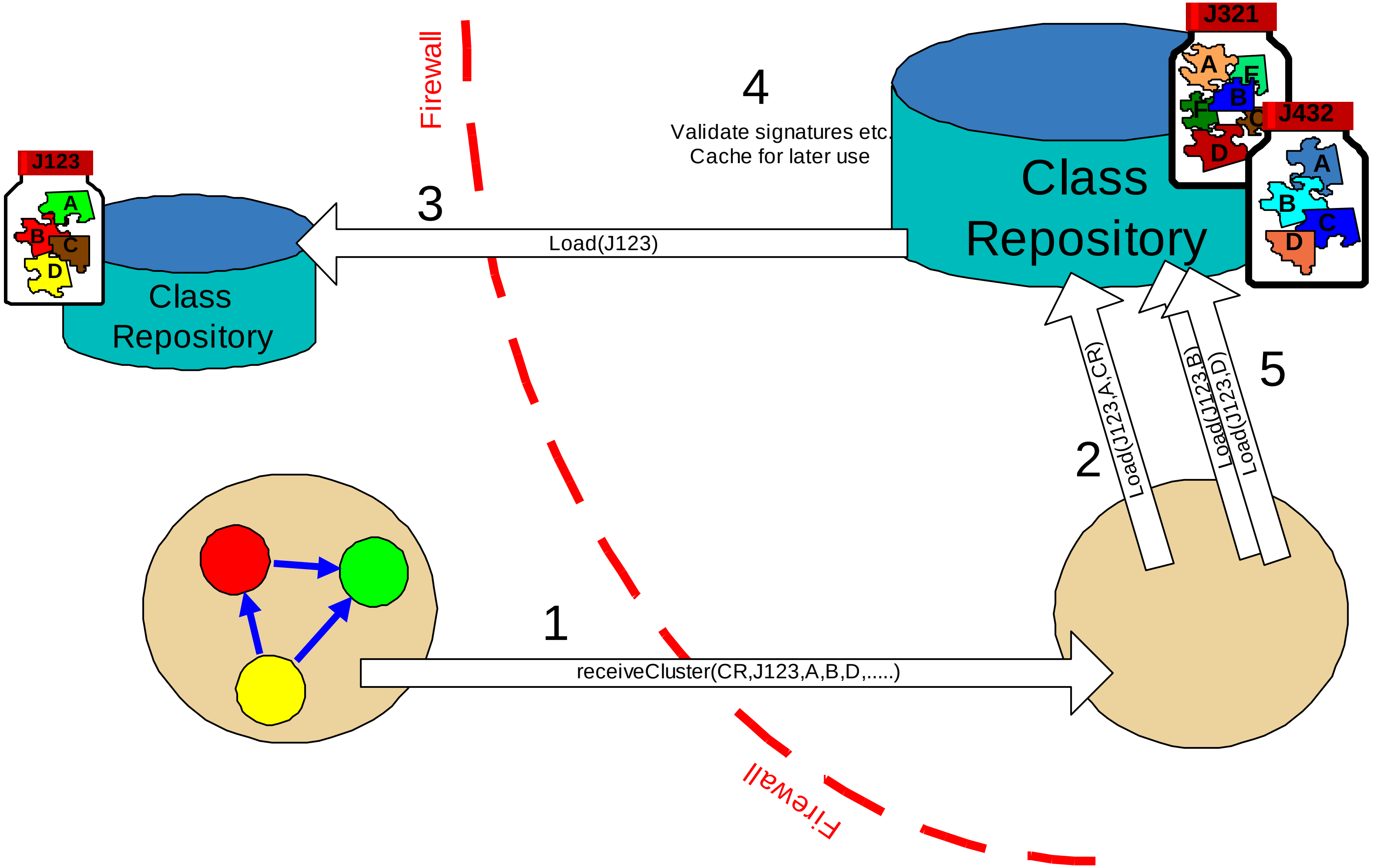
- Level 1 directories:
  - in unreliable hosts (e.g. browsers, client places etc.)
  - have parents at level 2
- Level 2 directories
  - On servers. Approx. 1 per LAN
  - have parents at level 3
- Level 3 directories
  - Backup servers. Approx. 1 per LAN
  - no parents

Class Loading Issues

- Where do we load classes from?
  - Mobile agents come from ordinary hosts
  - The class may already be available locally
- 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?

These issues apply equally to Agents and Applets

Resolving a set of remote classes



Security Problems

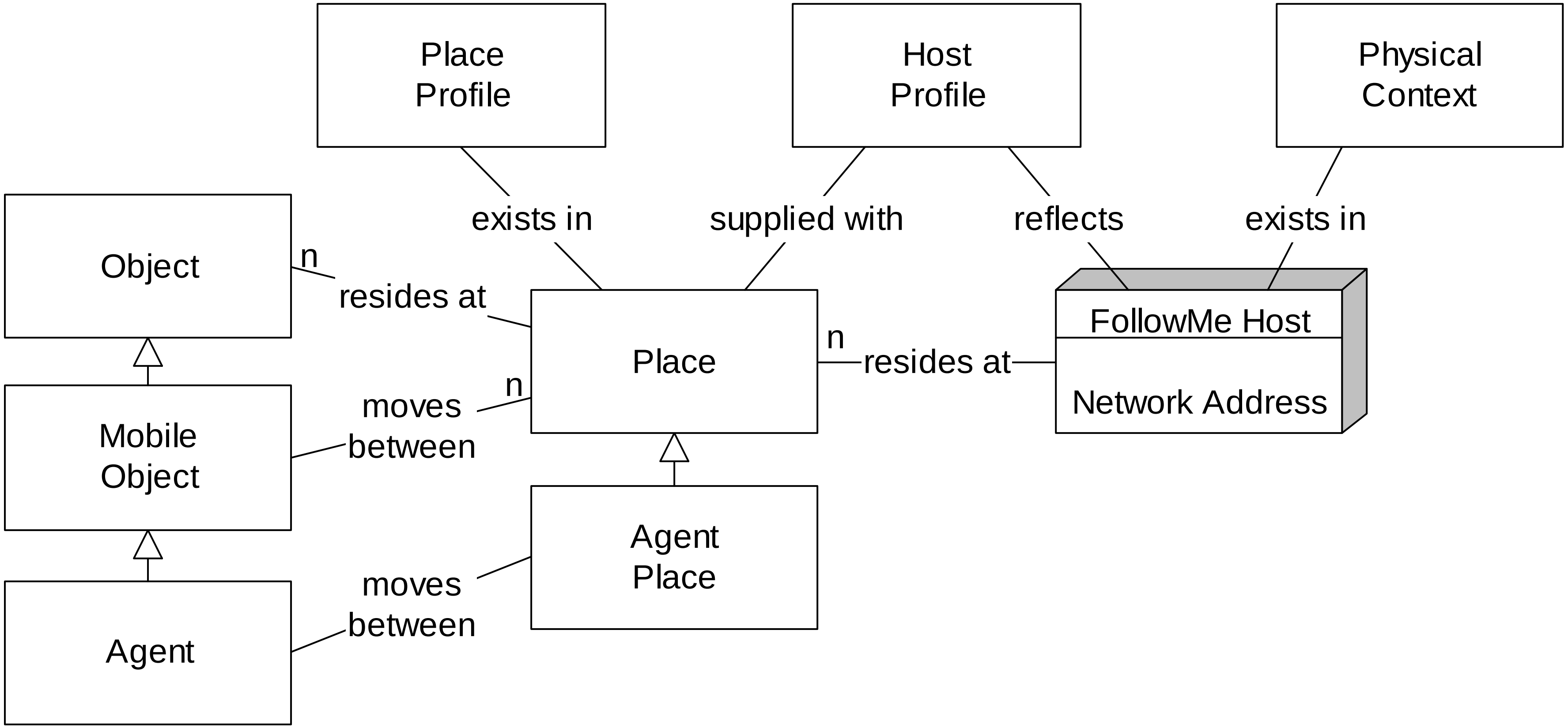
- A Place needs to be protected from its Clusters
  - But current Java Security Managers cannot prevent a Cluster from abusing its ThreadGroup
- A Cluster needs to be protected from untrusted Places
  - They need to carry secrets
- The Name Relocation Service can be fooled
  - Cluster hijacking or host masquerading

Encryption and Authentication is being integrated.

Stronger Encapsulation

- System services may violate encapsulation
  - AWT Threads may enter AWT Components through a back door
- Such services need to be isolated.

Places, Objects and Hosts


Place Profile
Host Profile
Physical Context
exists in
supplied with
reflects
exists in
Object
n
resides at
Place
n
resides at
FollowMe Host
Network Address
Mobile Object
n
moves between
Agent Place
Agent
moves between