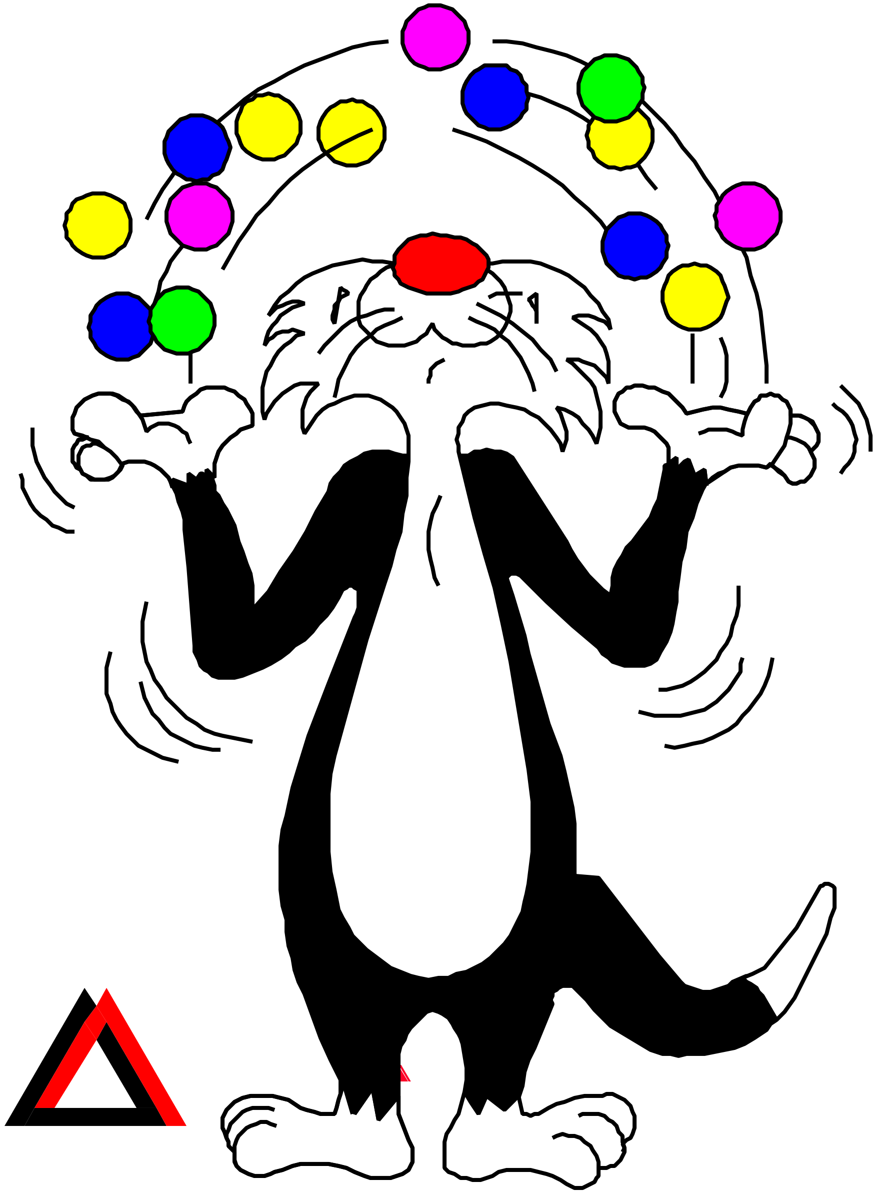


The Mobile Object Workbench

Richard Hayton

ANSA Technical Board

9th January 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

FlexiNet and Kafka

Originally we intended to combine Kafka and FlexiNet in order to active mobility

- It has proved easier to add mobility directly to FlexiNet
- Kafka has a strong security story
  - We will be using ideas from this, and possibly code
- Kafka has investigated Class Loader issues
  - We will build on that experience

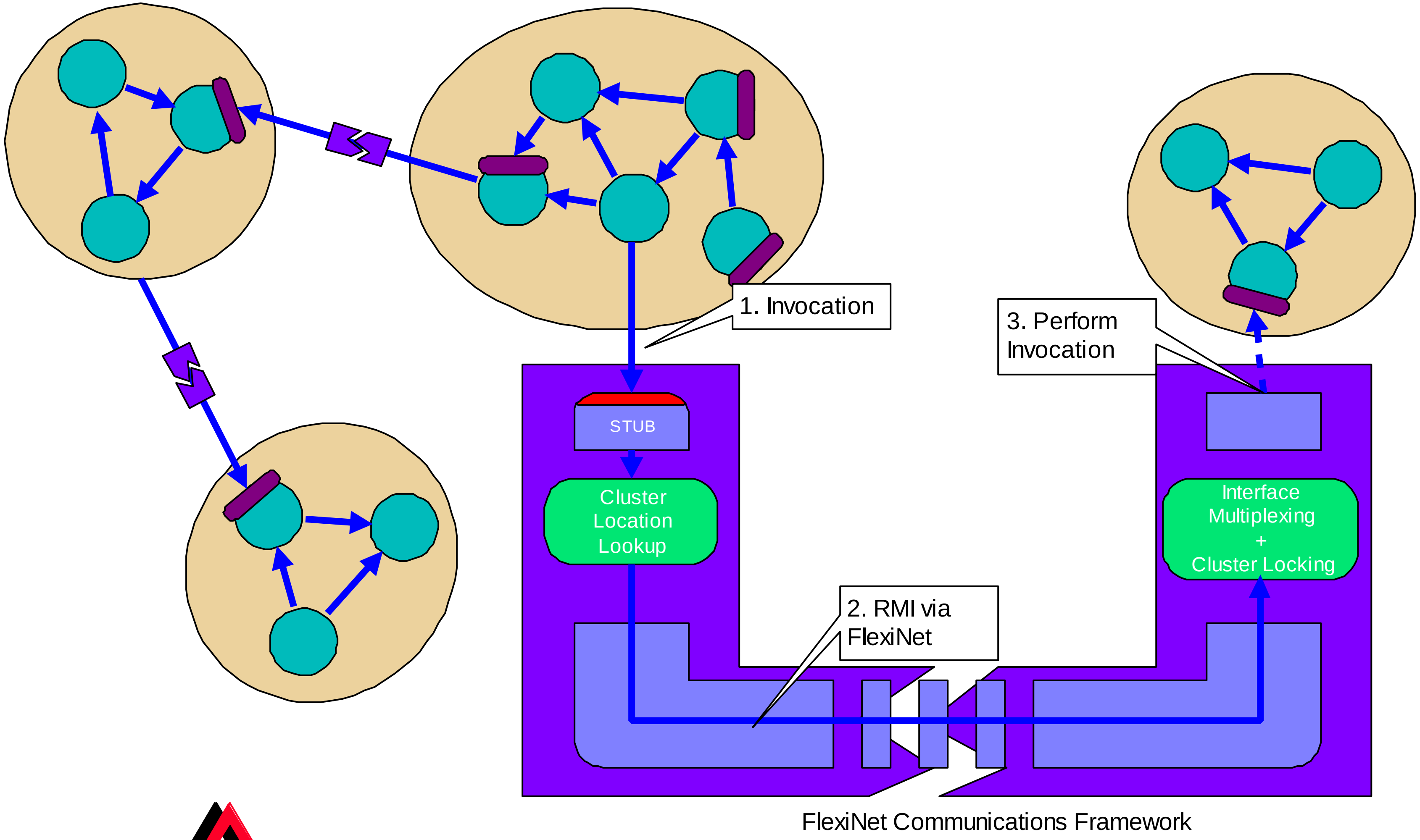
MOW Issues

- Unbinding
  - Removing an object from its execution environment
- Movement
  - Moving the object to a new execution environment
- Rebinding
  - Ensuring that references to the object prior to the move now refer to the newly moved object.

Unbinding

- We must determine what should move
  - (one object or a collection?)
- We must ensure that after the unbind, there are no references to the unbound object(s)
- We cannot break language level references
- Approach
  - Group objects into units for mobility ‘Clusters’
  - Tightly bind objects within a cluster (Java references)
  - Loosely bind clusters (comms. framework)

Clusters



STUB

Cluster Location Lookup

Interface Multiplexing + Cluster Locking

1. Invocation

2. RMI via FlexiNet

3. Perform Invocation

FlexiNet Communications Framework

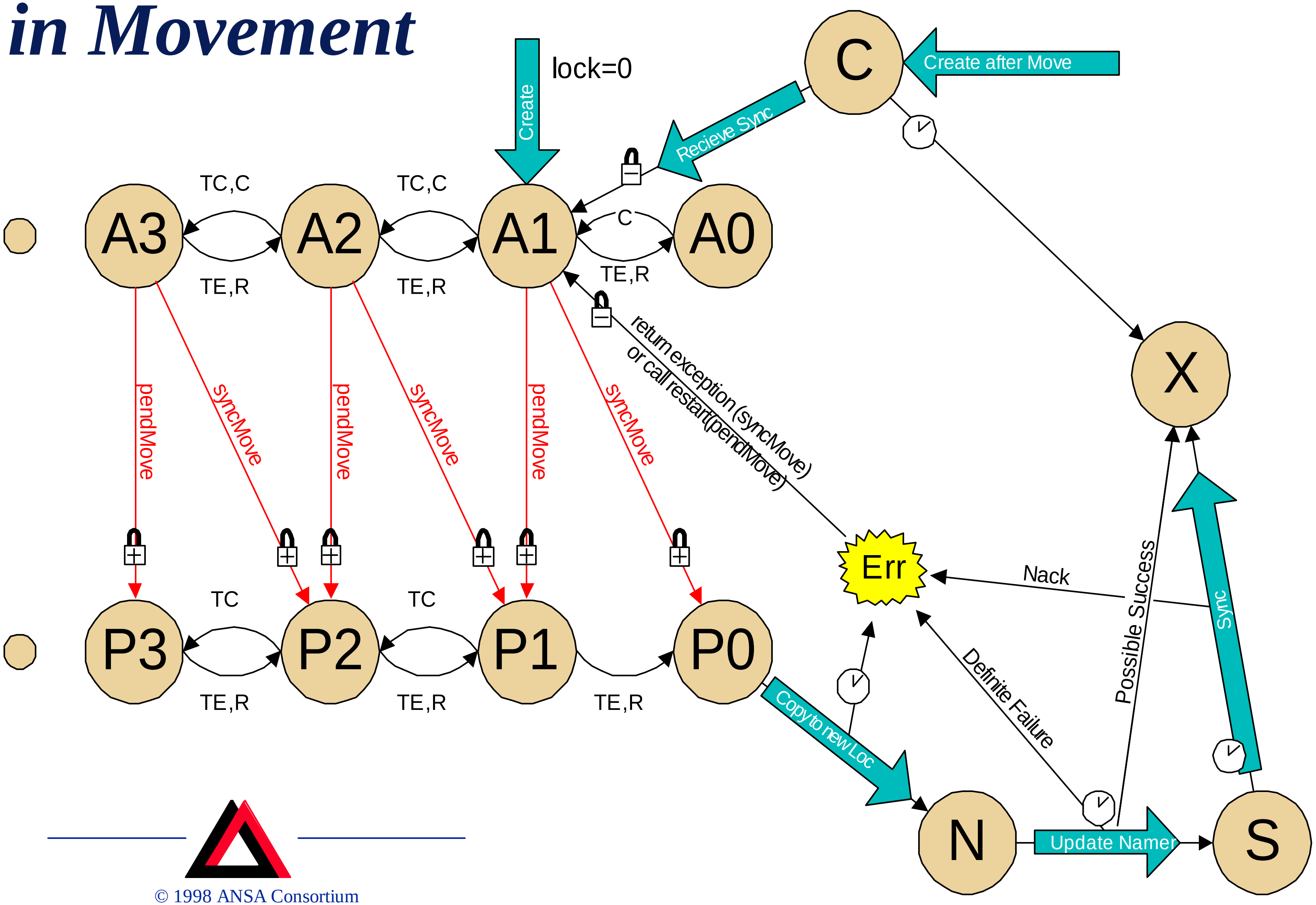
Strong Encapsulation

- 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

Cluster Movement

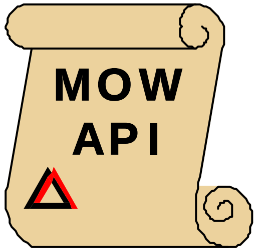
- Is this simply copying an object and discarding the original?
  - Yes. EXCEPT the copy must represent a consistent state
- Only move when:
  - There are no active threads within the cluster
  - This implies there are no calls in progress
- We use locking and thread counting to achieve this

Ensuring Consistency in Movement



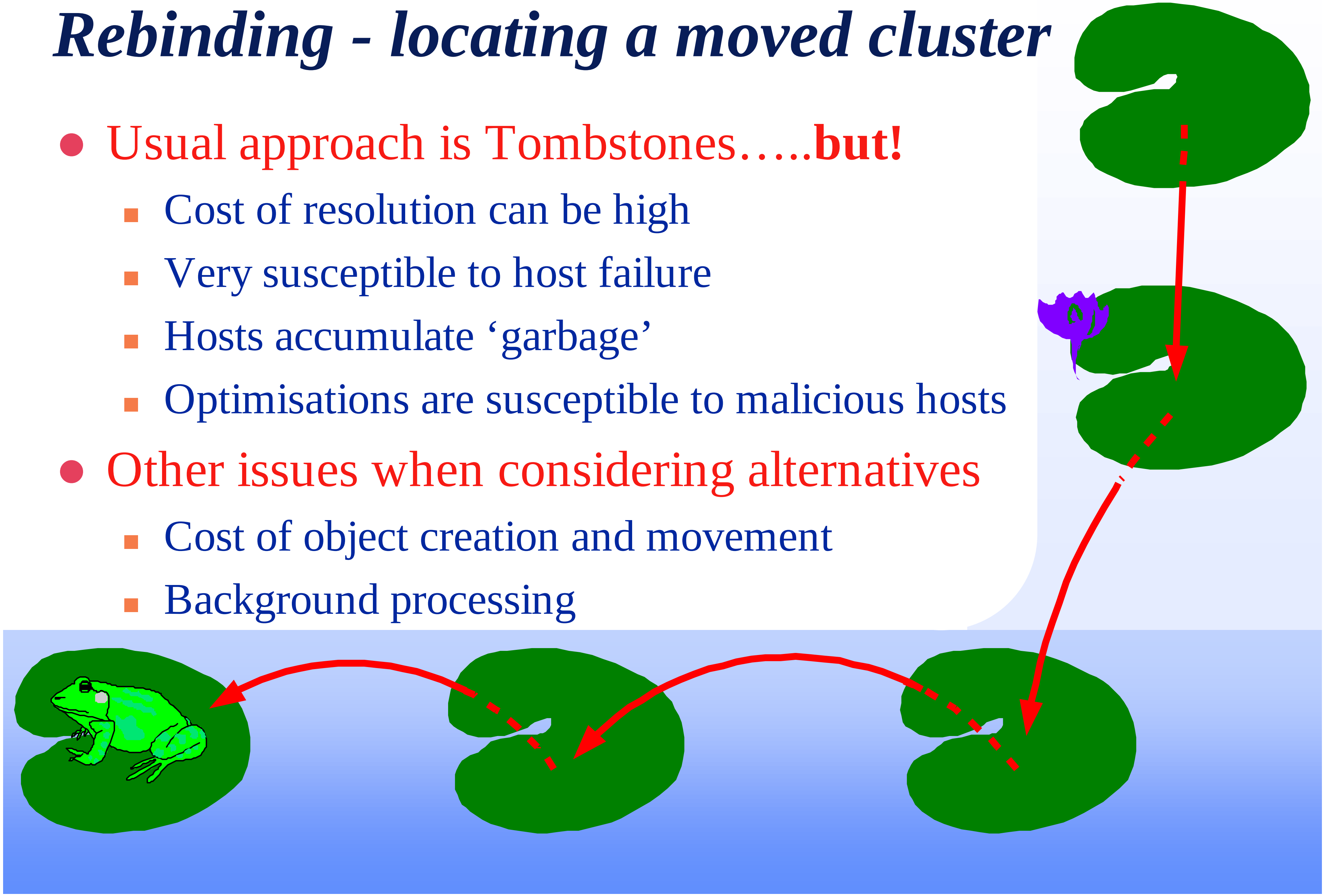
Mobility API

```
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);
}
```



Rebinding - locating a moved cluster

- Usual approach is Tombstones…..**but!**
  - Cost of resolution can be high
  - Very susceptible to host failure
  - Hosts accumulate ‘garbage’
  - Optimisations are susceptible to malicious hosts
- Other issues when considering alternatives
  - Cost of object creation and movement
  - Background processing



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

Stage One: Directory Based

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

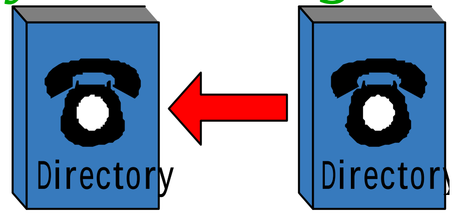
- On move
  - update directory d with old address⇨new address
- On lookup
  - try the previous address, if it fails contact d

Analysis

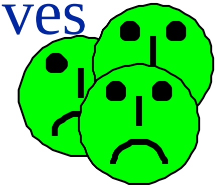
- Security/Integrity
  - High trust in directory
  - Clusters can choose an appropriate directory ☺
  - Hosts cannot fool others into thinking they have a cluster
- Move/Lookup Cost ☺
  - At most two additional calls
  - One may be to a distant host if the directory is ill placed ☹
- Reliability
  -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



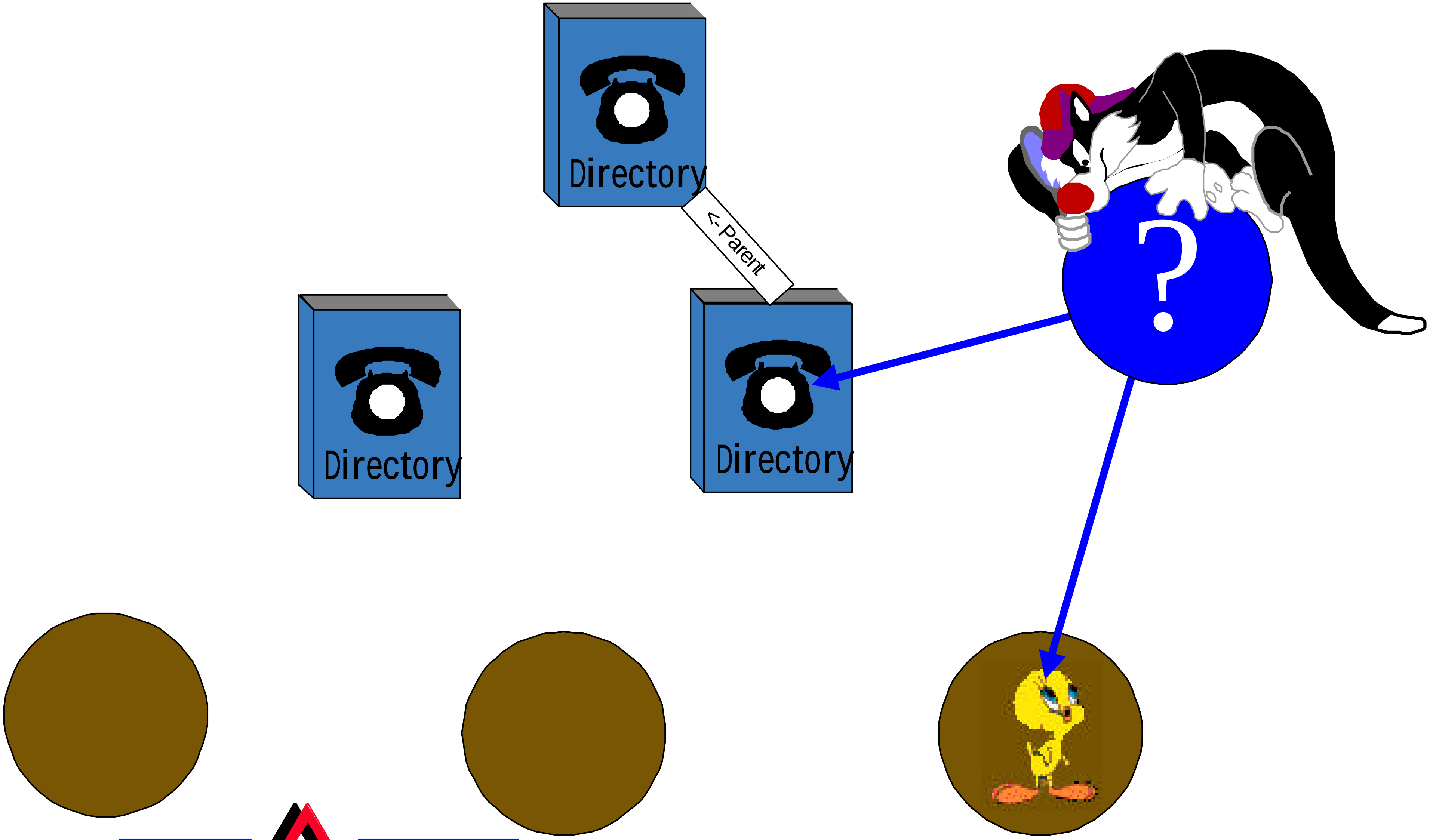
Stage Three: Improving Reliability

- 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

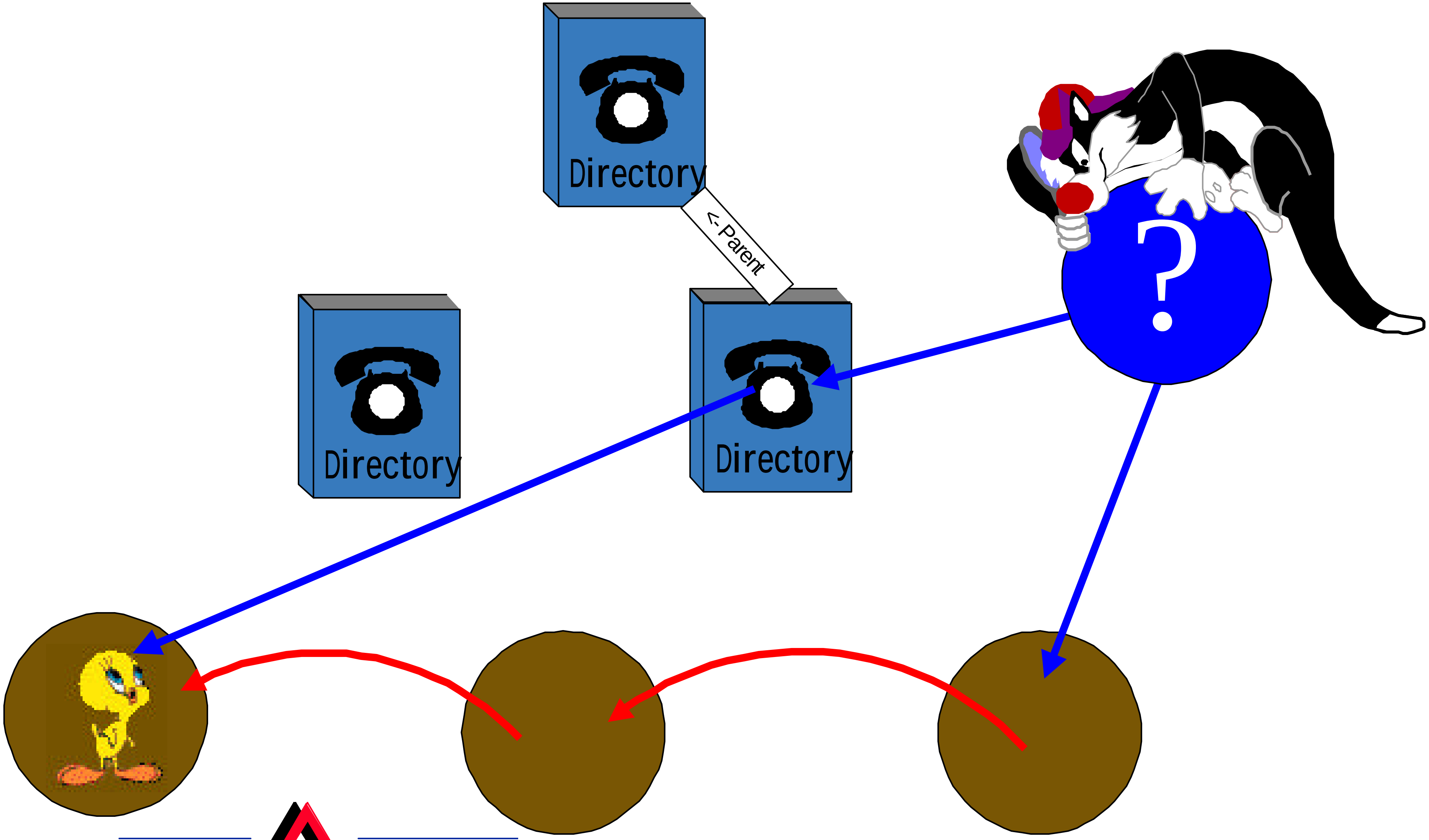
- Analysis of reliability:
  - n hosts out of 2n (each tombstone or its parent)

- Analysis of background cost
  - Low - *if we only copy to parent when we create tombstones*

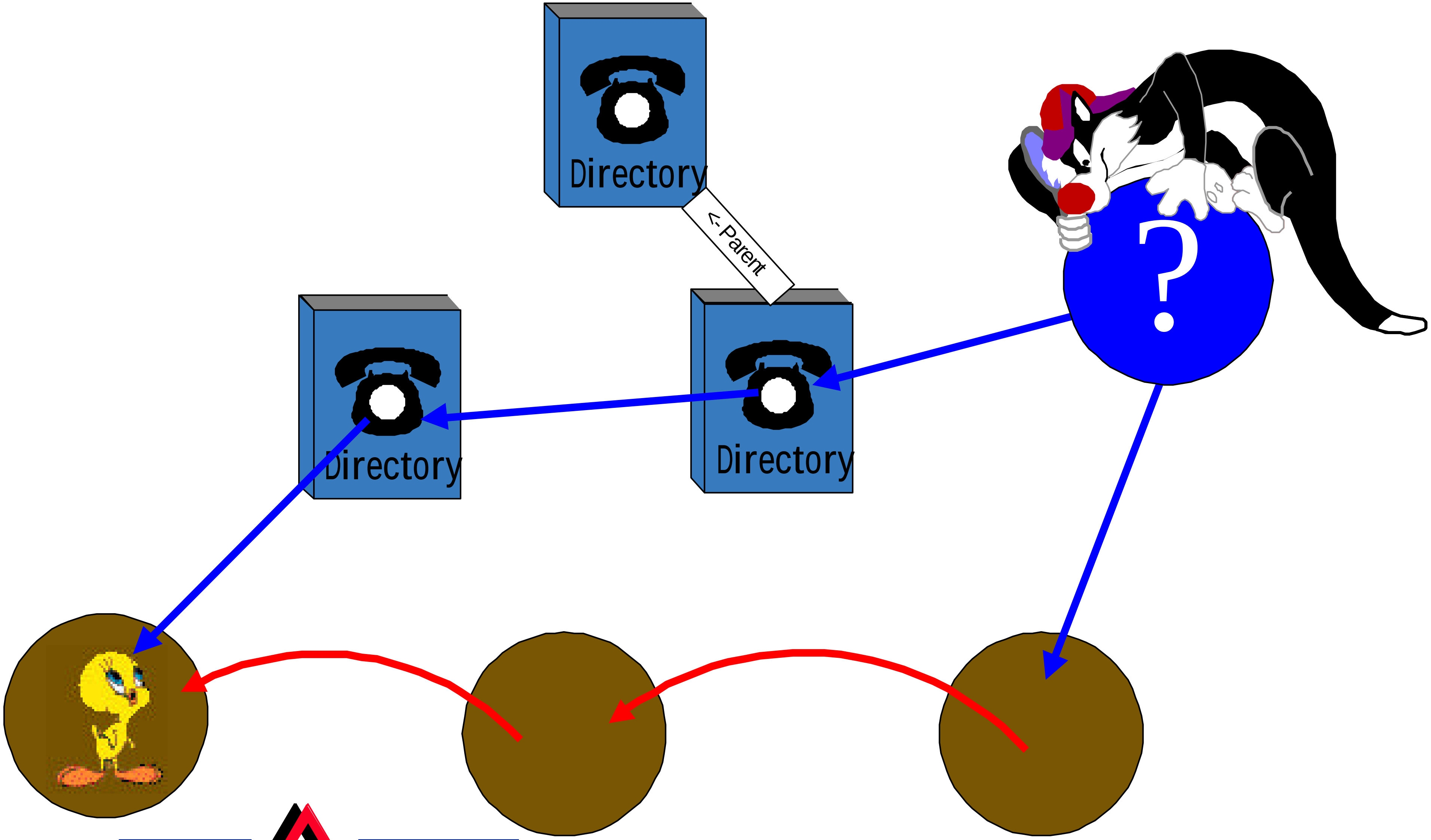
Catch the Birdie......



Catch the Birdie……

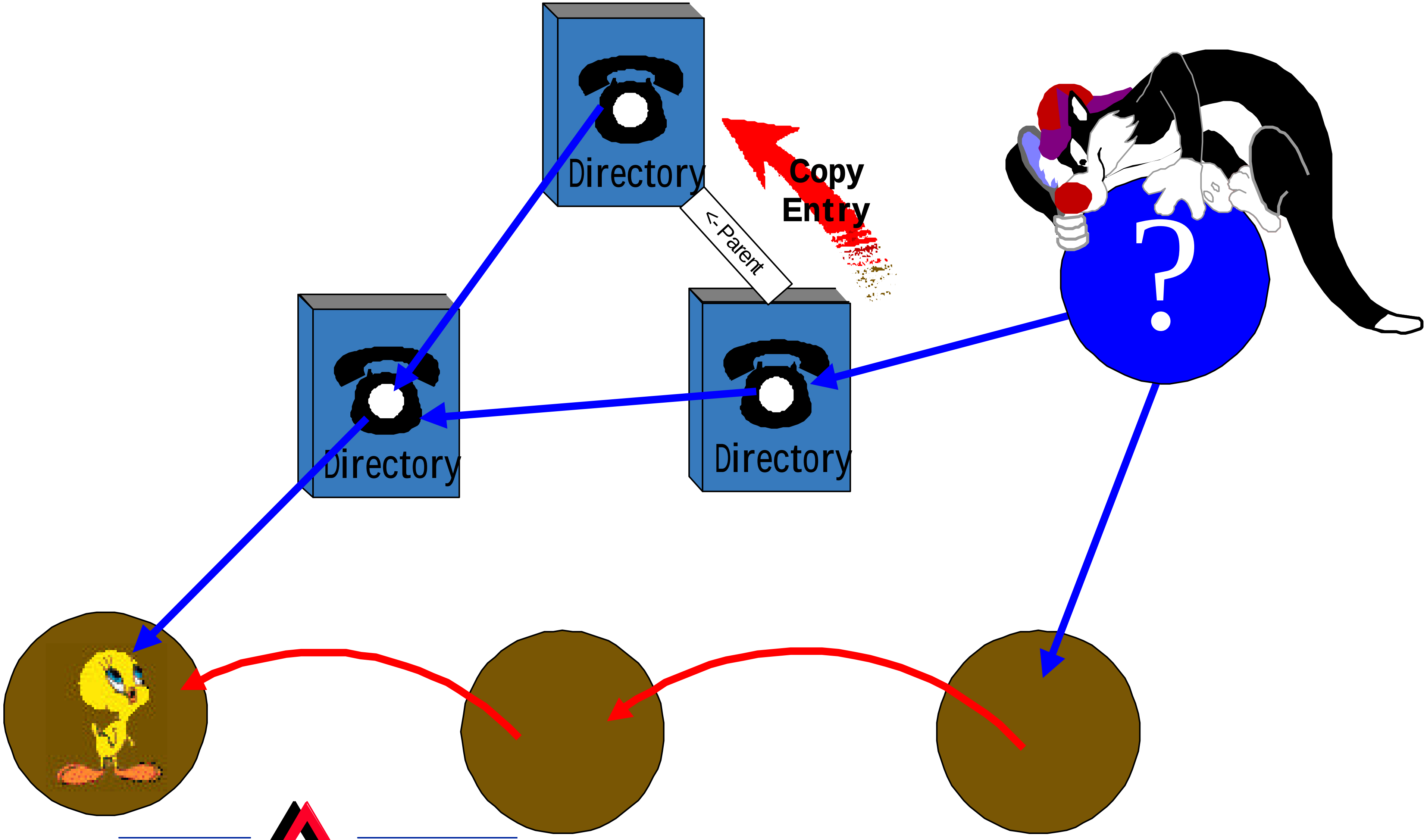


Catch the Birdie…...



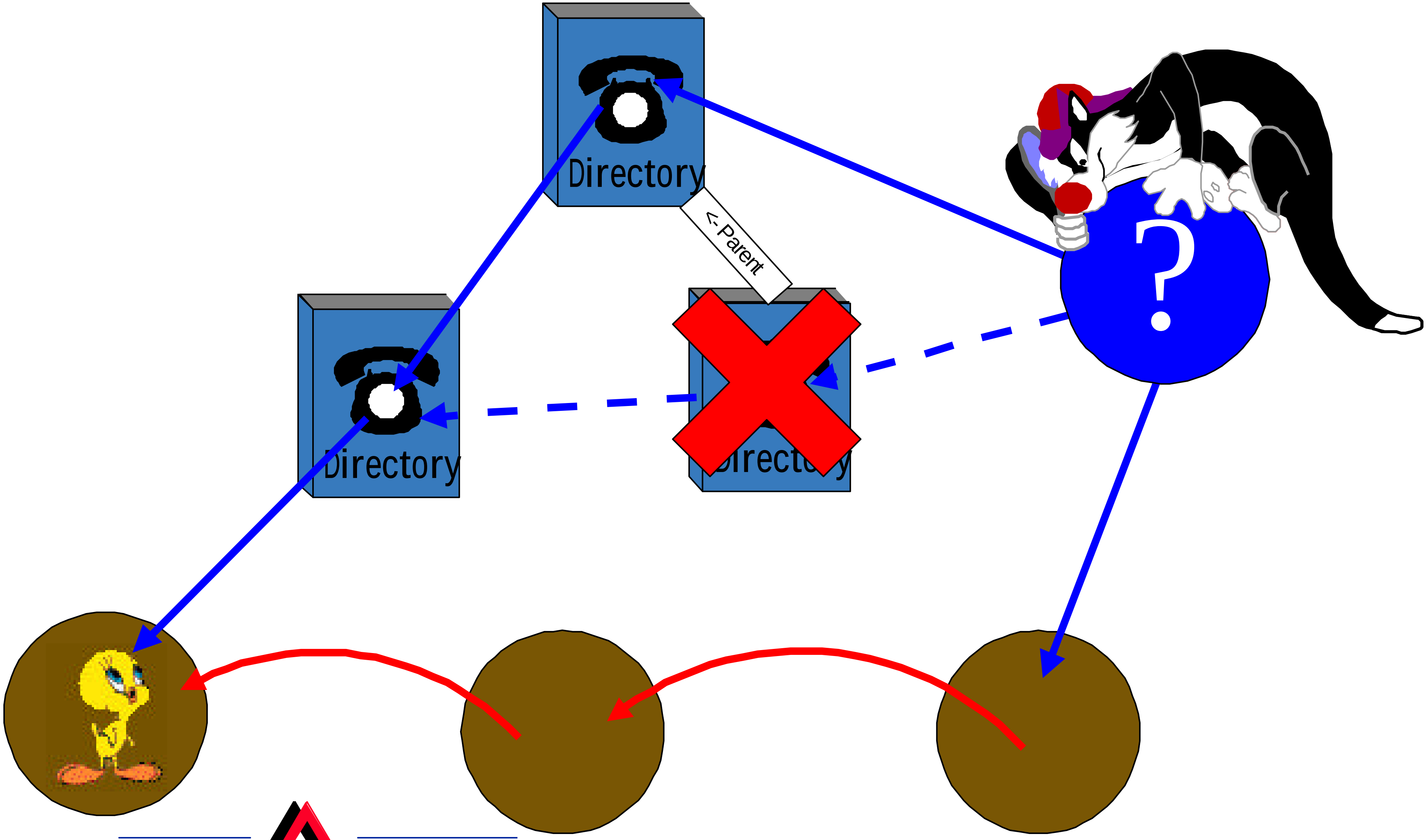
© 1998 ANSA Consortium

Catch the Birdie…...



© 1998 ANSA Consortium

Catch the Birdie…...



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

Status

- MOW Release 0.1 available. (Release 1.0 at end of Jan)
  - Strong Encapsulation Implemented
  - Movement and Copying of Clusters
  - Location/Movement transparent communications
- Name Relocation Service:
  - Level 1 implemented
  - Design work continuing

Next

- Security Issues
  - How can mobile objects prove their identity and carry secrets?
  - Preventing malicious disruption of services?
- Strong Encapsulation
  - Wrap AWT and other APIs
  - Investigate implementation options for servers
  - Browser issues