

ICS 362 Distributed Systems

Distributed Systems: Part 11

Lecturer: Toby Daniel

Moving Resources

- Why would a resource move?
- What are the implications for naming?



Moving resources

- If a resource needs to move, then assuming it only moves to a different machine within the same domain then only the local DNS database needs to be changed.
- But if a resource moves to a different domain the global or administrative layer would need to be updated.
 - Updating databases at these levels can be inefficient.
- Adding a link to the new resource is also inefficient as it extends the lookup phase.
- If resources are highly mobile then the problem grows.

Identifiers vs addresses

- When multiple names can point to multiple addresses it makes sense to use an identifier.
 - Any entity can have exactly one identifier, and that identifier can never be reassigned to another entity.

Location Services

- As Identifiers are unique, they can be stored locally, so a naming service can return the appropriate identifier for the resource.
- A location service then accepts the identifier for a resource, and then returns the current address (or addresses) of the entity.

Locating Mobile Entities

- There are several solutions for locating mobile entities;
 - Broadcasting / Multicasting
 - Forwarding Pointers
 - Home Based Approaches
 - Hierarchical Methods

Broadcasting

- Such as with Address Resolution Protocol (ARP).
- When a request is received for a resource, the request is broadcast through the network, along with the resource identifier.
 - All machine are requested to check if they have the identifier, (or rather an access point to the identifier).
 - Machines with an access point reply back across the network.

Multicasting

- An obvious limitation of broadcasting comes from scalability, and the cost of broadcasting a large network.
- Multicasting limits where the broadcast request is sent, so the message is only sent to a specified range of machines.
- Whenever a resource connects to the network they also broadcast to the multicast address(es) to identify themselves and their current location.

Forwarding Pointers

- The forwarding pointers approach simply adds a pointer to a resources new location in its old location.
- Whenever an entity moves it adds a pointer to its new location.
 - One problem with this approach is the possibility of very long pointer chains.
 - Another is vulnerability to losing entities if the pointer chain breaks.

Forwarding Pointers

- When an entity moves from A to B, it leaves behind in A a reference to its new location at B.

Advantages:

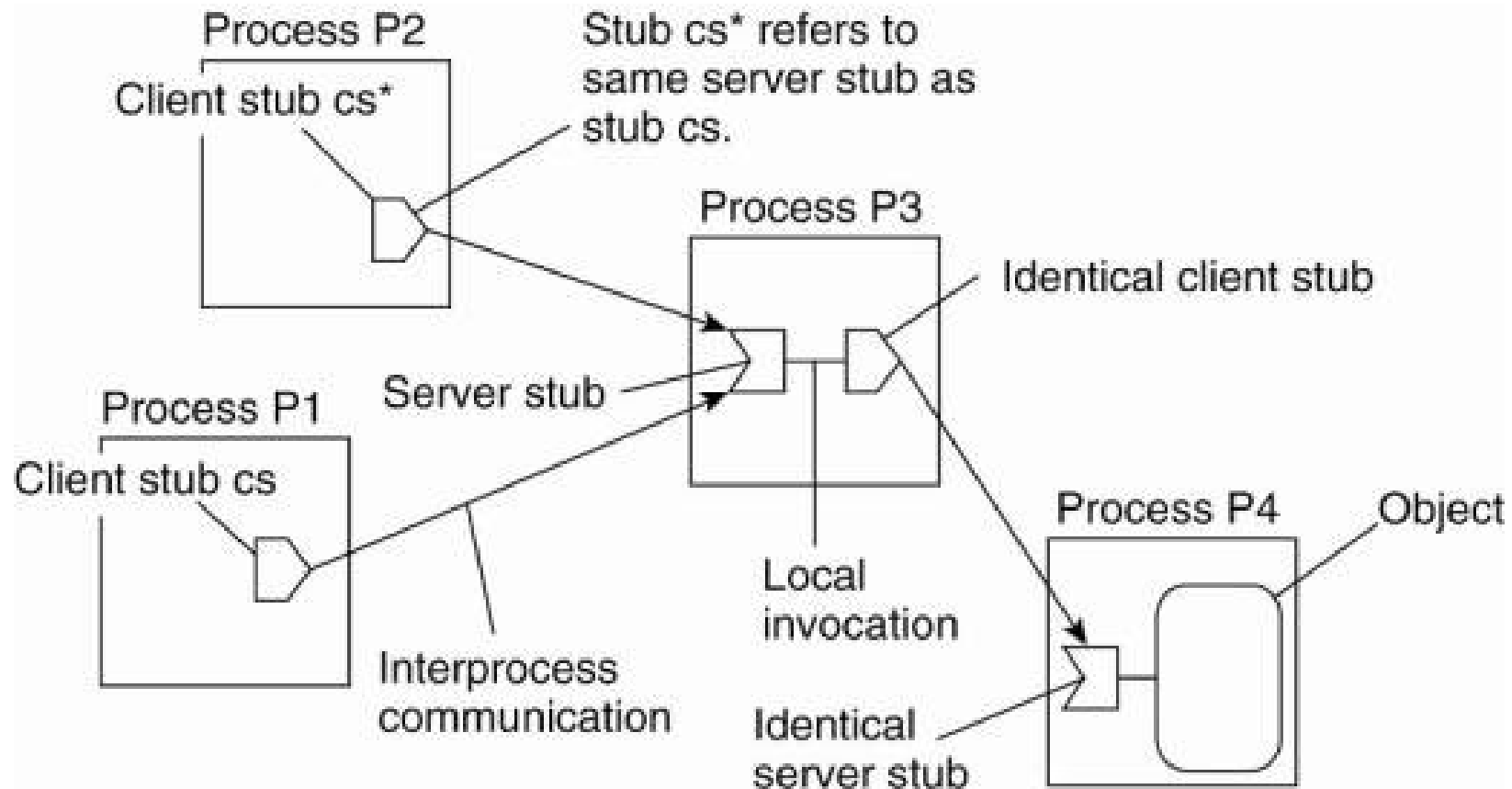
- Simple

Disadvantages

- No special measures are taken, a chain for a highly mobile entity can become so long that locating that entity is prohibitively expensive.
- All intermediate locations in a chain will have to maintain their part of the chain of forwarding pointers as long as needed.
- A drawback is the vulnerability to broken links.

Forwarding Pointers

- The principle of forwarding pointers using (client stub, server stub) pairs.



Home Based Approaches

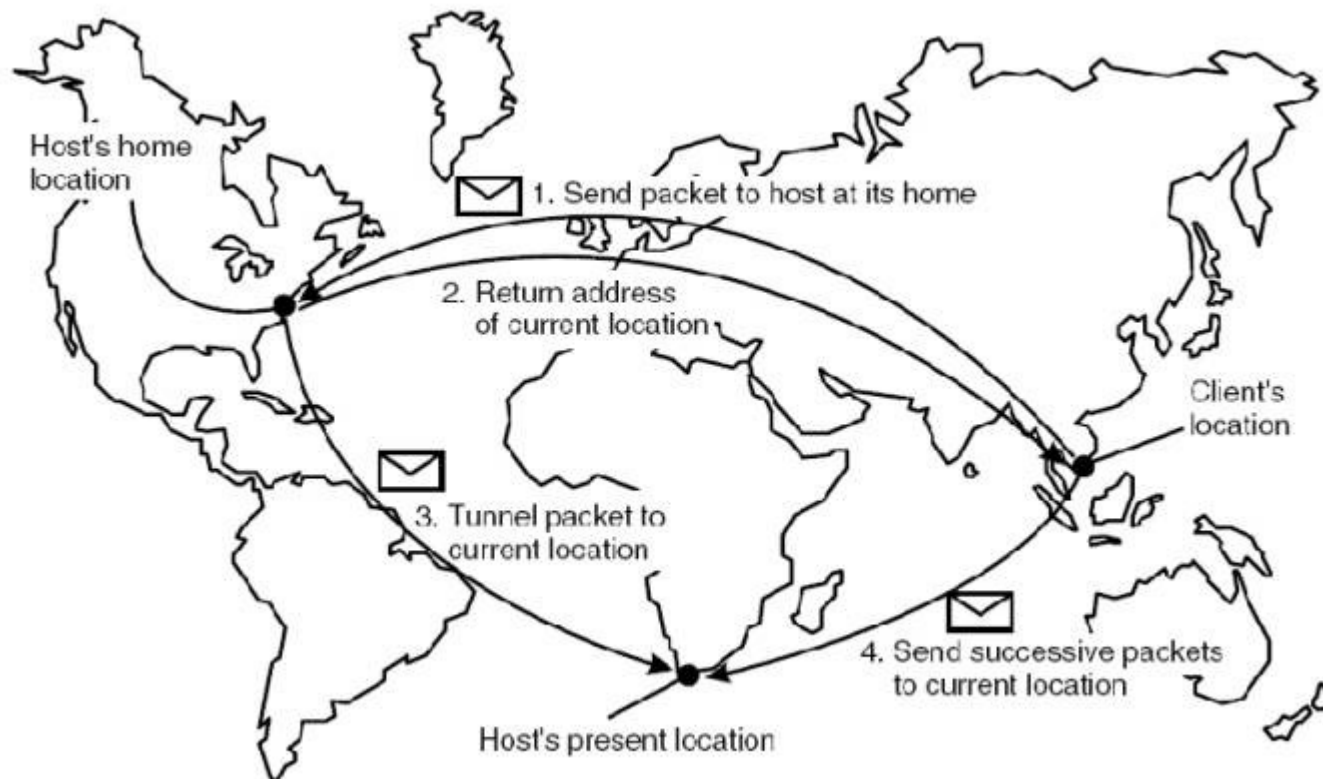
- Mobile IP for example.
- The approach is based on each entity having a home location.
- The home location will store the current location of the entity and forward a message to it.

Home-Based Approaches

An entity's home address is registered at a naming service.

The home registers the foreign address of the entity.

Clients always contact the home first, and then continues with the foreign location



Hierarchical Methods

- In Hierarchical methods larger networks of entities are separated into domains, broken down into a tree like structure.
- Each node stores a record for every sub domain.
 - When locating an entity the root node can be checked as it stores records for every sublevel node.
 - If the root node contains the entity it can forward the request to the appropriate sub domain node, until the entity is reached.

Hierarchical Methods

- If the root node is expected to know about every sublevel entity there are scalability considerations.
 - By space, the root node only need store a small point to lower level nodes, so space isn't a primary concern.
 - By requests, as the root node would need to handle every request for every sublevel entity.

Unreferenced Entities



Unreferenced Entities

- Naming and Location services are useful for dealing with entities, but we also need to consider what to do when an entity is no more.
 - Removing unnecessary resources is important, while it can be easy in a nondistributed system, for the last user of a resource to delete it, in a distributed system we may not know if the resource will be needed again.
 - Leaving a resource available just because we don't know if it might be needed again is unacceptable as it is a waste of resources.
- Consider the number of broken links on the web.

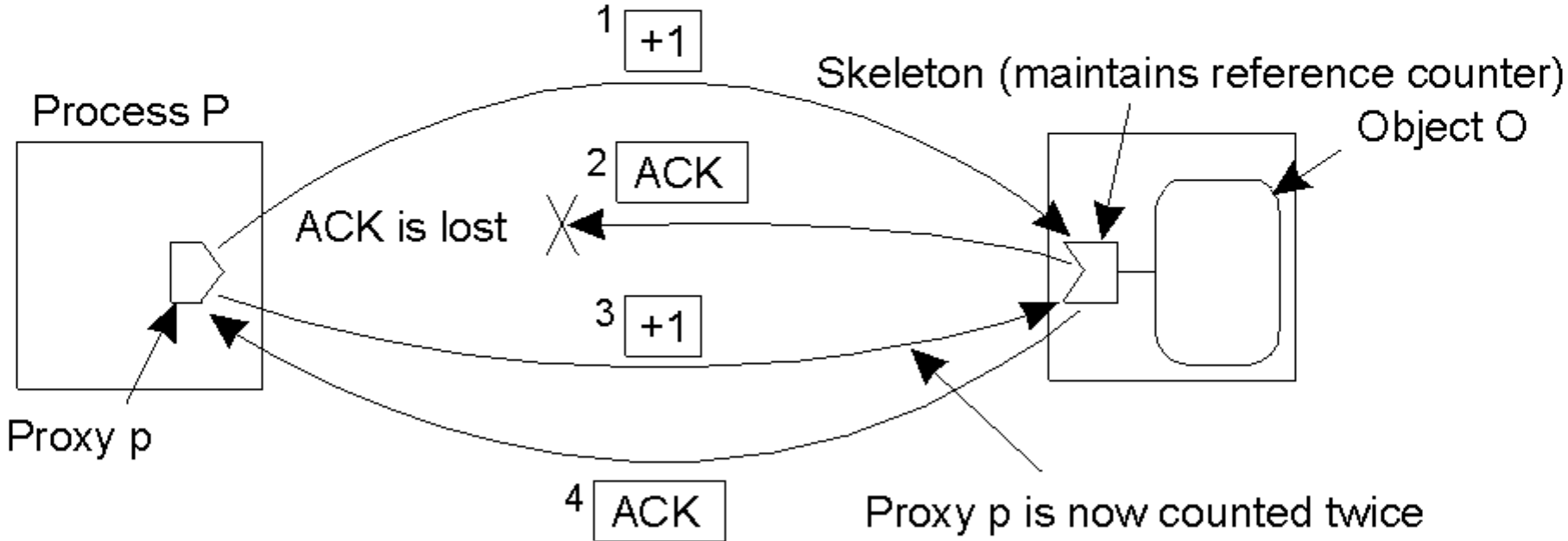
Unreferenced Objects

- Unreferenced objects cause problems as they are wasting resources.
 - Objects are manipulated through calls to their interface.
 - If there is no remote reference to the object then it can't be accessed or used.
- With single processor systems, unreferenced objects are easier to identify and remove, through garbage collection techniques.

Generic Solution: Reference Counting

- Increment a counter when an object is referenced.
- Decrement a counter when an object reference is no longer needed.
- Delete the object when the reference count is zero.
- Leads to a number of problems, mainly due to unreliable communications systems.

Reference Counting Problems



Adding Robustness

- Lost acknowledgements are easy to detect and deal with (a problem that has been solved by many other networking technologies).
- Duplicates can also be handled.
- A number of reliable enhancements to simple reference counting exist, but suffer from performance and scalability problems (they are also complex):
 - Weighted Reference Counting
 - Generation Reference Counting

Enhancements to Counting

- **Reference Listing:** an reference count is not maintained. Instead, as list of proxies that point to the object is maintained by the object.
- The list has some important properties: if a proxy is already in the list, adding it again *does not change the list*. Also, if a proxy is not in the list, removing it from the list *does not change the list*.
- Reference Listing is said to be “idempotent” – an operation can be repeated any number of times without affecting the end result. So a proxy can keep adding/removing itself from the list until an ACK is returned.
- **Key point:** duplicates are OK, and reliable communications is NOT required.

Naming...

- Names refer to entities, which are organised into name-spaces.
 - *Address*: an entities access point.
 - *Identifier*: one-to-one mapping to an entity.
 - *Name*: human friendly descriptor.
- Distributed systems must support mobile entities, so traditional naming systems like DNS aren't enough.

Naming...

- Four approaches to finding/naming mobile entities:
 - Broadcasting/multicasting: only works on LAN's.
 - Forwarding pointers: large chains cause problems.
 - Home based systems: e.g. Mobile-IP.
 - Hierarchical, dynamic domains.
- Removal of “no longer needed” entities is important.
- Distributed systems garbage collection technologies are organised around:
 - Simple reference counting systems.
 - Reference tracing.

Midterm Exam

- Anything from Chapters 1 – 4 of Tanenbaum's *Distributed Systems Principles and Paradigms*
 - Key requirements
 - Hardware concepts
 - Topography
 - Software concepts
 - Client server model
 - Protocols
 - RPC
 - MOC
 - Streaming
 - Processes and threads
 - Software Agents
 - Naming concepts