CompSci 131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Naming in DS**

- **Reading assignment:**
  - **Today: Naming, 5.1-5.3**
  - **Next time: Coordination, 6.1-6.2**
    - » **Complete the assignment *before* next class**

Last Lecture Covered

- **More MPI functions, implementation**

Naming *and* locating

- **A name is a string of bits identifying an entity**
  - **Entity can be anything: device, file, connection, address…**
  - **May or may not be human-friendly**
- **Entities are operated on/accessed *via an access point***
  - **Access points are themselves entities!**
    - » **Their name is called an *address (aka location)***
- **A name needs to be resolved to a location**
- **It is good to separate a name and an address!**
  - **Allows migration, re-organization, multiple access points**
- **Need a DS *naming service, which* may be *distributed***
  - **Scalable, but harder to implement**

Types of Names

- **An entity name that is independent from its address is called location independent**
  - **An FTP server vs an address of a host where FTP server resides**
    - » **or mobile phone numbers**
- **A true** (or unique) **identifier**
  - **An identifier the refers to at most one entity**
  - **Each entity is referred to by at most one identifier**
  - **An identifier always refers to the same entity**
    - » **is never reused**
- **An identifier is thus a unique reference to an entity**
  - **An address may or may not be an identifier**

Locating entities

- How does one map a name to a location or an address?
  - Aka resolve a name
- Simplest way – a table of <Ent, Addr> pairs
  - But does not work well in many DS settings, for instance when entities migrate
- There are two main ways of locating entities
  - Using flat names
  - Using hierarchical names

Flat Names

- **Let's start with simple solutions for locating an entity with a flat name**
  - **Using a broadcast**
    - » **ARP is an example for a local net**
  - **Using forwarding pointers**
    - » **Client-server stubs**
    - » **Can short-circuit after first access**

Locating an entity using broadcast

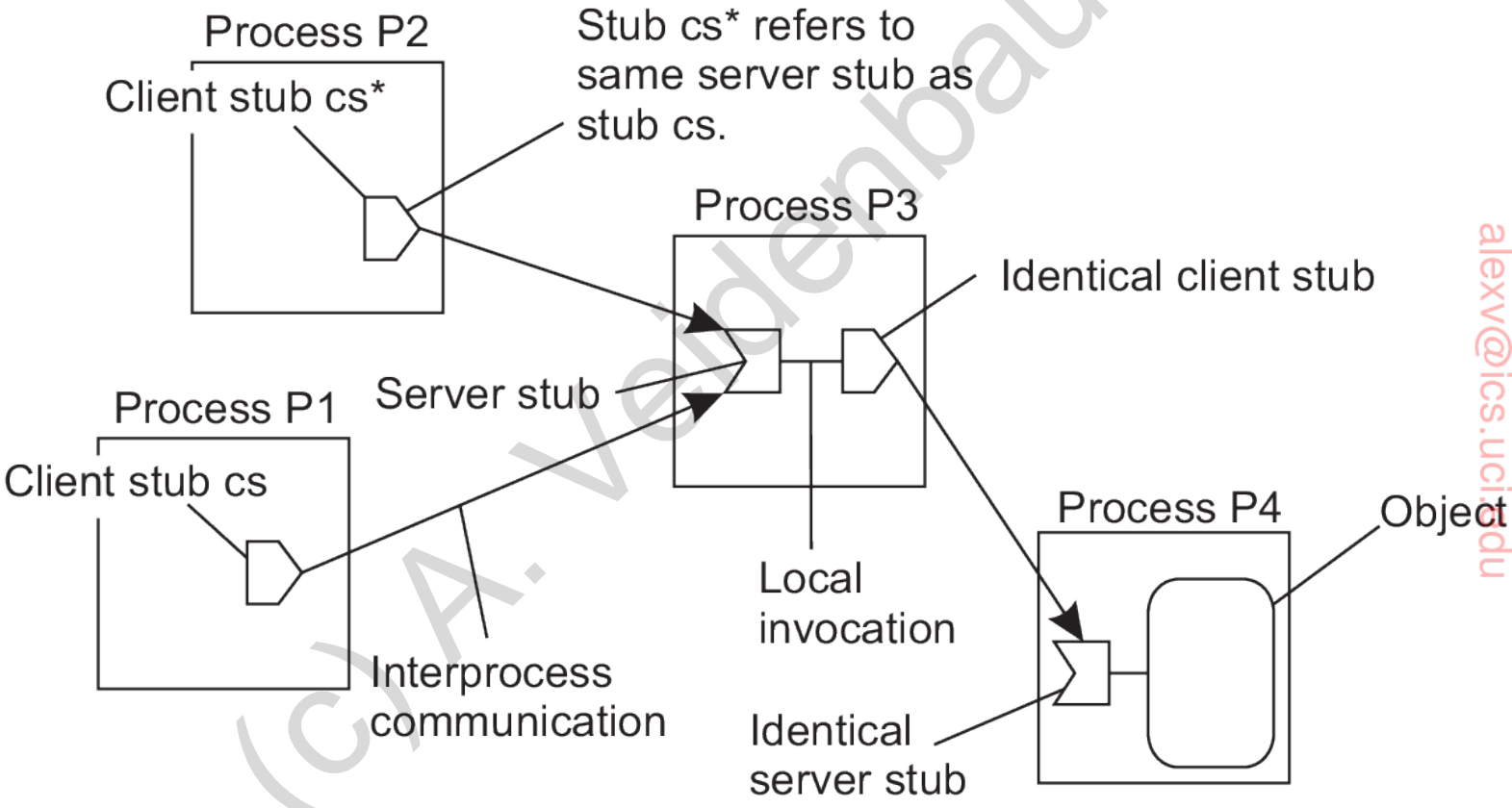
- A local or a wireless network can
  - Use broadcast to send an entity identifier
  - All network nodes check if they have the entity
  - Only nodes that offer an access point reply
- Used in address resolution protocol (ARP)
  - IP address to ethernet address
    - » No routing tables required!
- Can also use restricted multicasting
  - Limited to a group of hosts
    - » Supported in the Internet
- Not a scalable solution

Using Forwarding Pointers

- A relocated entity leaves a forwarding pointer at original location
  - Very simple
  - Scalable? Not for frequently moving servers!
    - » Can create a very long forwarding chaing
- Example: used with RPC
  - A moving server leaves behind a forwarding stub
    - » Just to send the request to a new location
  - The server at the new location handles the request
- Want to short-cut this for performance
  - Send new location to requestor for update

Example of Forwarding Pointers

- **A relocated entity leaves a forwarding addr**

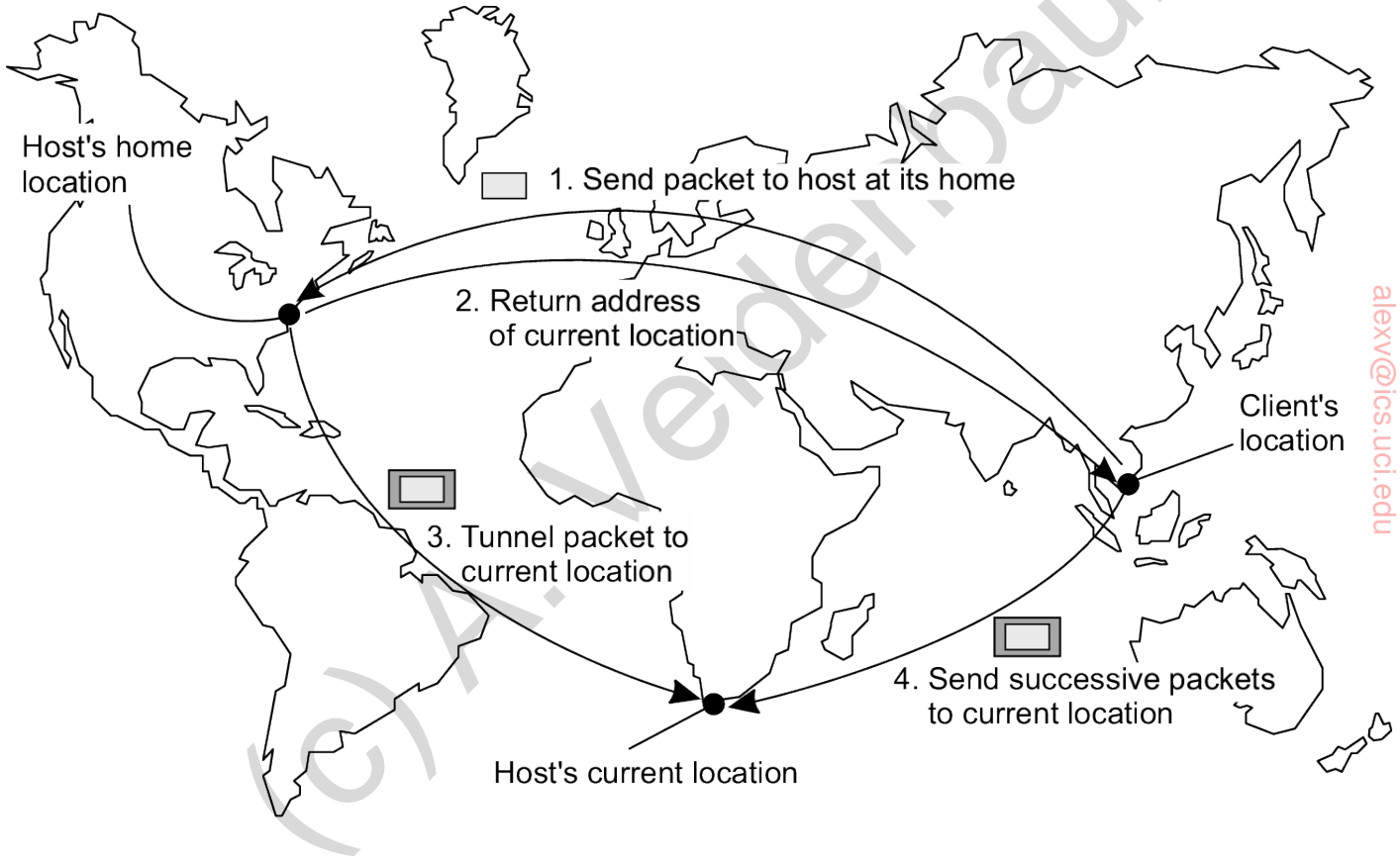


Home-based Approaches

- A home location is unique
  - Can always find an entity starting at its home
- Used for mobile entities in large networks
  - Cell phone roaming
  - Mobile IP, using a "home agent"
- Uses a forwarding pointer as entity moves
- Largely invisible to applications
  - They only refer to original IP address
- Drawback – long delays possible

Home-based Example

- **Messages travel around the world**



Hierarchical Approaches

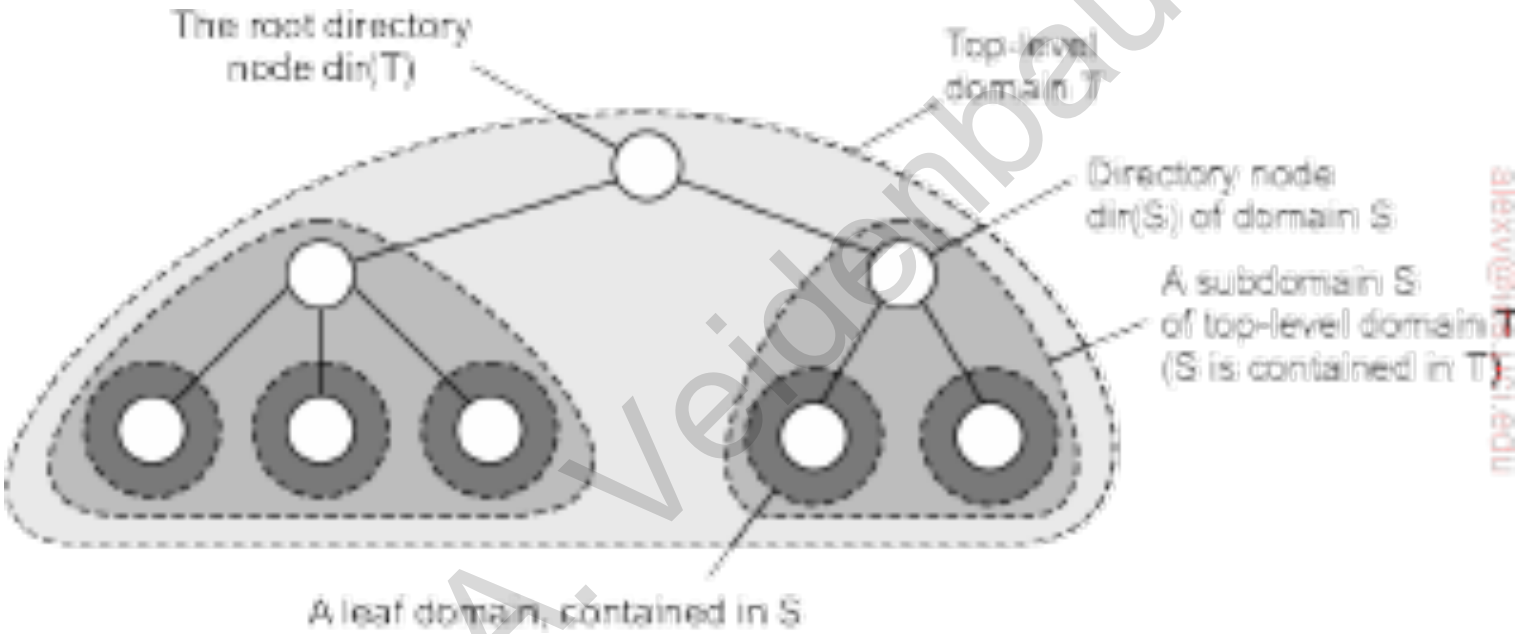
- **A Network is divided into sets of domains**
- **Domains may be 'nested' inside each other**
- **Two node types in a domain (**plus sub-domains**):**
  - **A directory node**
    - » **Contains an entry for each entity in the domain**
  - **A leaf node**
    - » **Typically a local network or a mobile cell**
      - **Contains the entity address**
- **One root node (no predecessors)**
  - **E.g. a directory tree**

Hierarchical Approaches

- **A leaf domain contains:**
  - An *address* for each entity in the domain
- **A directory node contains 'location' records**
  - For each entity in the domain it knows the sub-domain the entity is in
    - » But *not* the address
- **There can be replicas of an entity!**
- **Lookup of entity N in this organization**
  - originates at a leaf and goes up
  - Finds the first directory that contains the entry for N
  - This exploits locality!

Hierarchical Domain Example

- **A tree of domains**

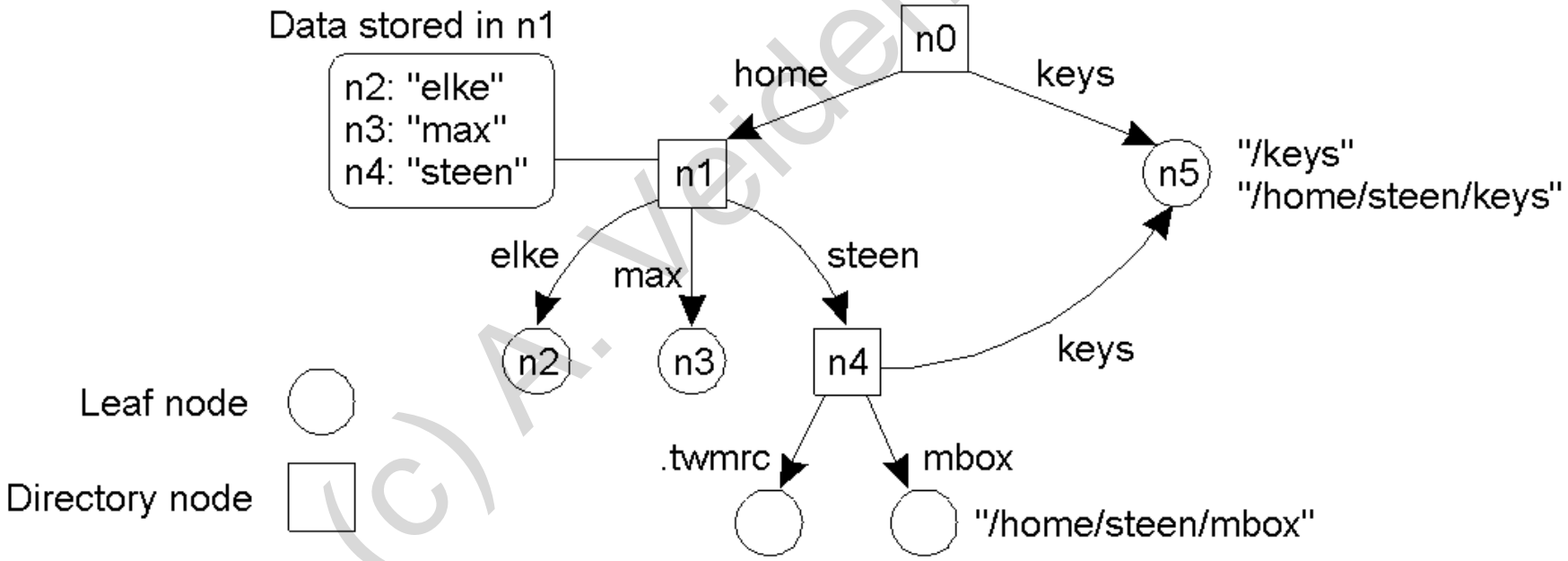


Structured naming

- **Flat names are not human-friendly**
- **Structured names are human readable names**
  - **File names, domain names**
- **Names are organized into *Name Spaces (NS)***
- **NSs are represented as directed graphs**
  - **Leaves are named entities**
  - **Directory nodes have *labeled* outgoing edges**
- **Any node in this graph is itself an entity**
  - **Has an identifier**
  - **Directory nodes have a dir table: (edge,node id)**

An example

- **General, but close to most file systems**
  - **Path name = a sequence of edge labels**
  - **From root: absolute path**
  - **Otherwise a relative path**

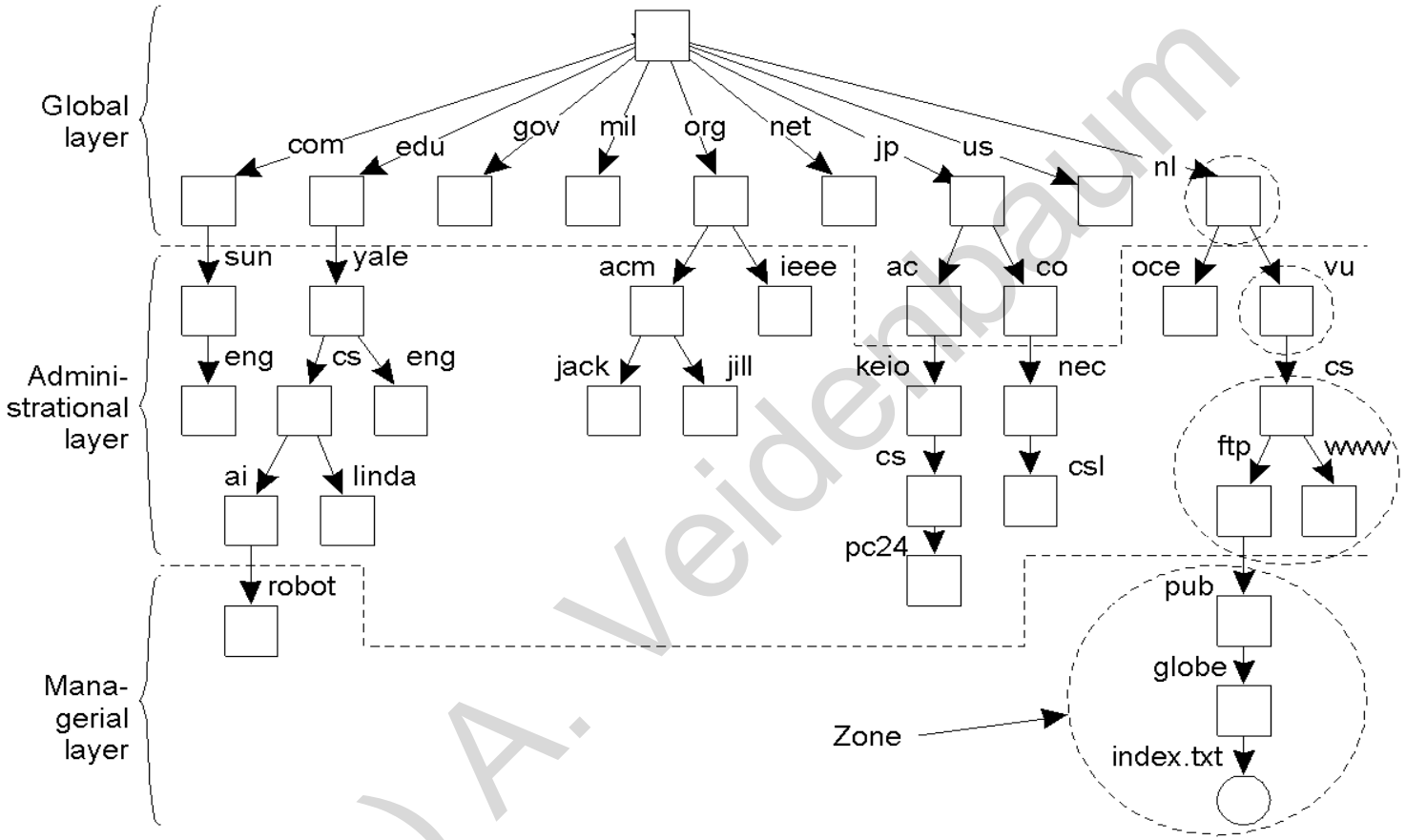


Name Resolution

- **The process of looking up a name**
  - **Go through edges in naming graph, until node found**

- **Closure mechanism: where and how to start**
  - **i.e. where is root?**
    - » **Hard-coded as inode0 in Unix**
  - **Local/global names**
    - » **HOME environment label**

Name Space Distribution (DNS)



- **Global layer: "stable" nodes (non-overlapping "zones")**
- **Administration layer: a node per one organization, relatively stable**

Properties

Item	Global	Administrational	Managerial
Geographical scale of network	**Worldwide**	**Organization**	**Department**
Total number of nodes	**Few**	**Many**	**Vast numbers**
Responsiveness to lookups	**Seconds**	**Milliseconds**	**Immediate**
Update propagation	**Lazy**	**Immediate**	**Immediate**
Number of replicas	**Many**	**None or few**	**None**
Is client-side caching applied?	**Yes**	**Yes**	**Sometimes**

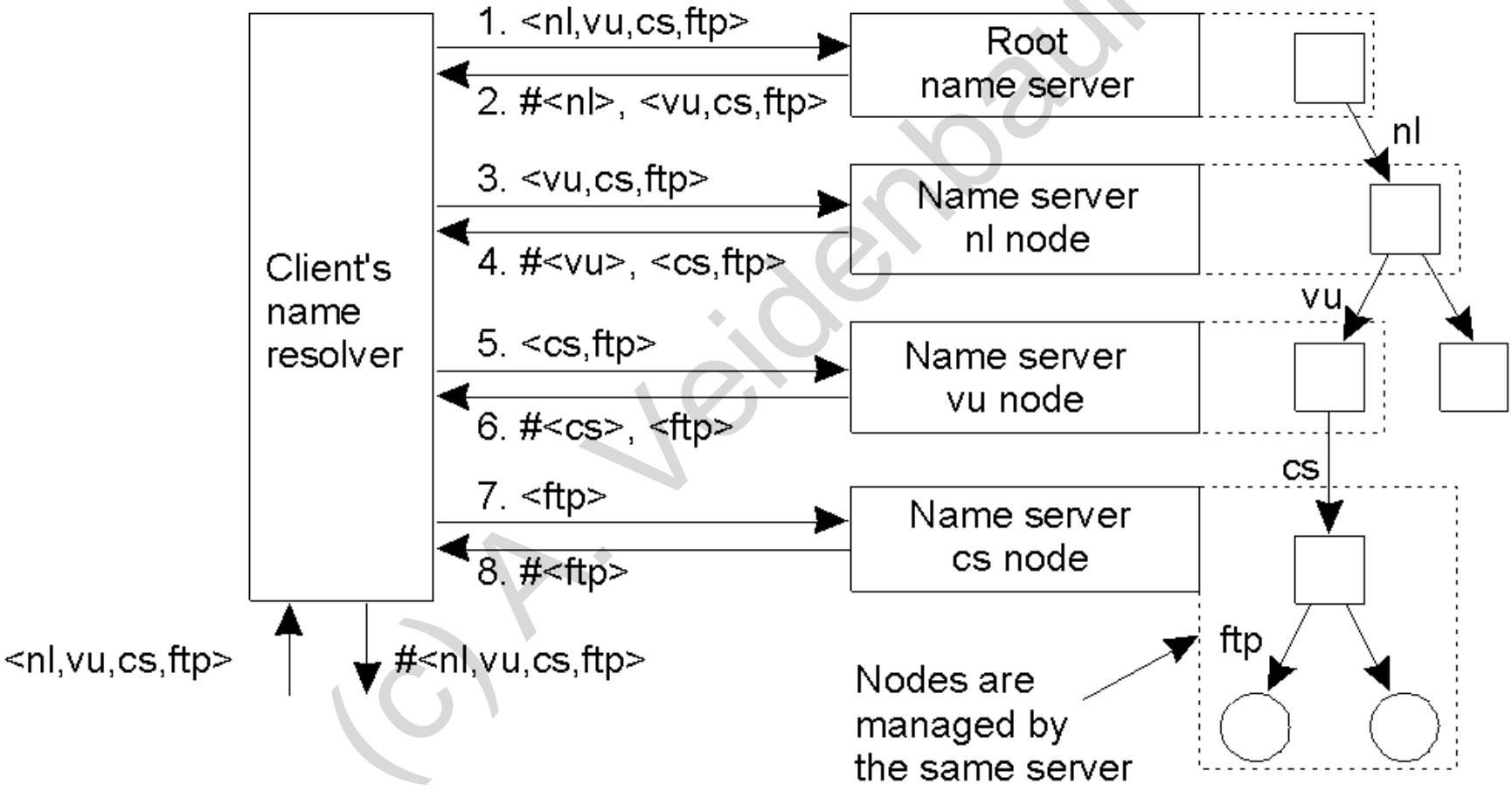
- **Used in DNS, but is generally applicable**
- **Properties affect availability and performance**
  - **Caching, replication**
    - » **Caching allows faster resolution, but creates consistency problems**

Name Resolution

- DS have "name resolvers"
  - A client has access to a local name resolver
- Two ways of resolving:
  - Iterative
  - Recursive

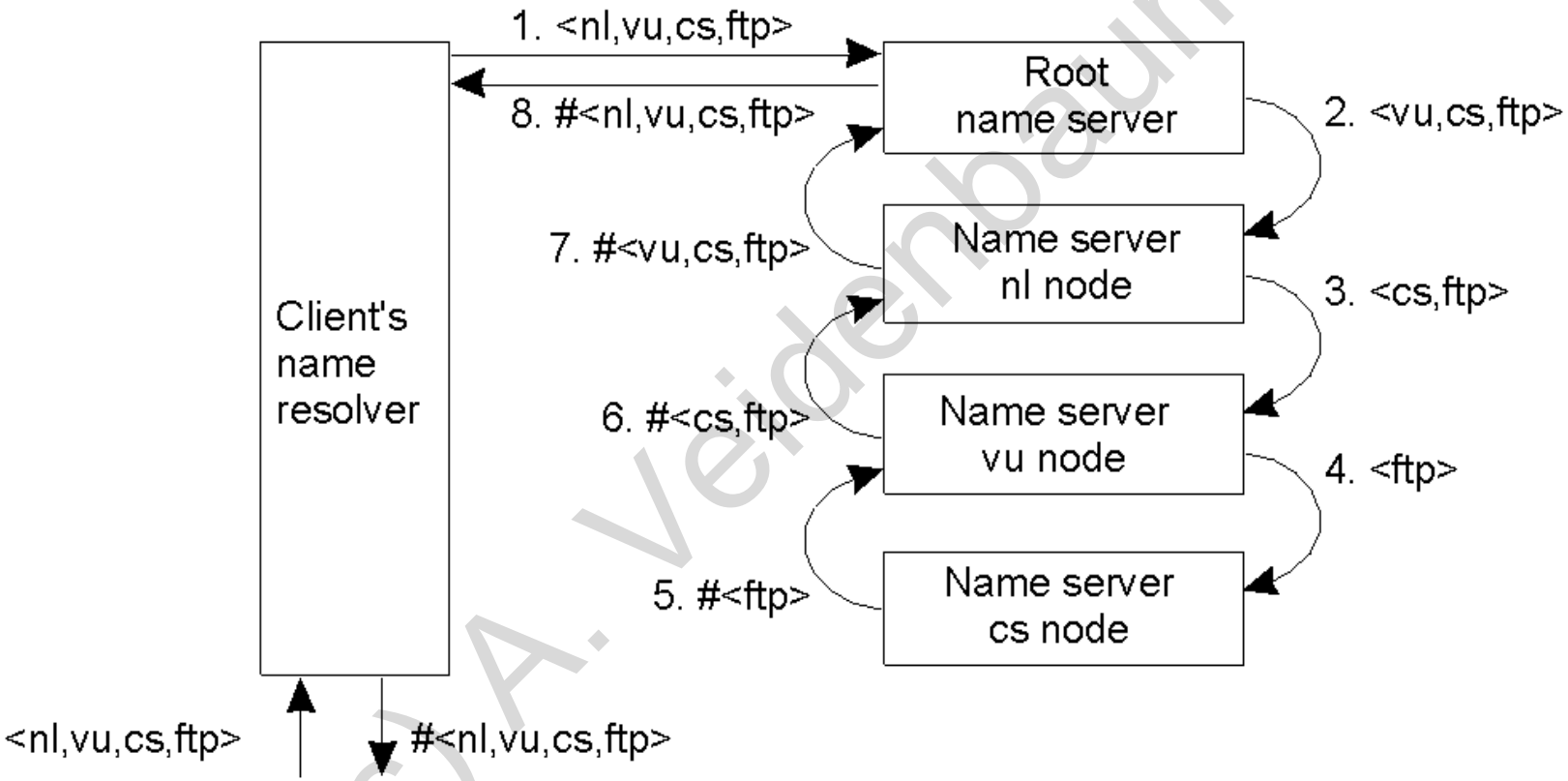
Iterative Name Resolution

- **Each name in the path is resolved by new client-server exchange**

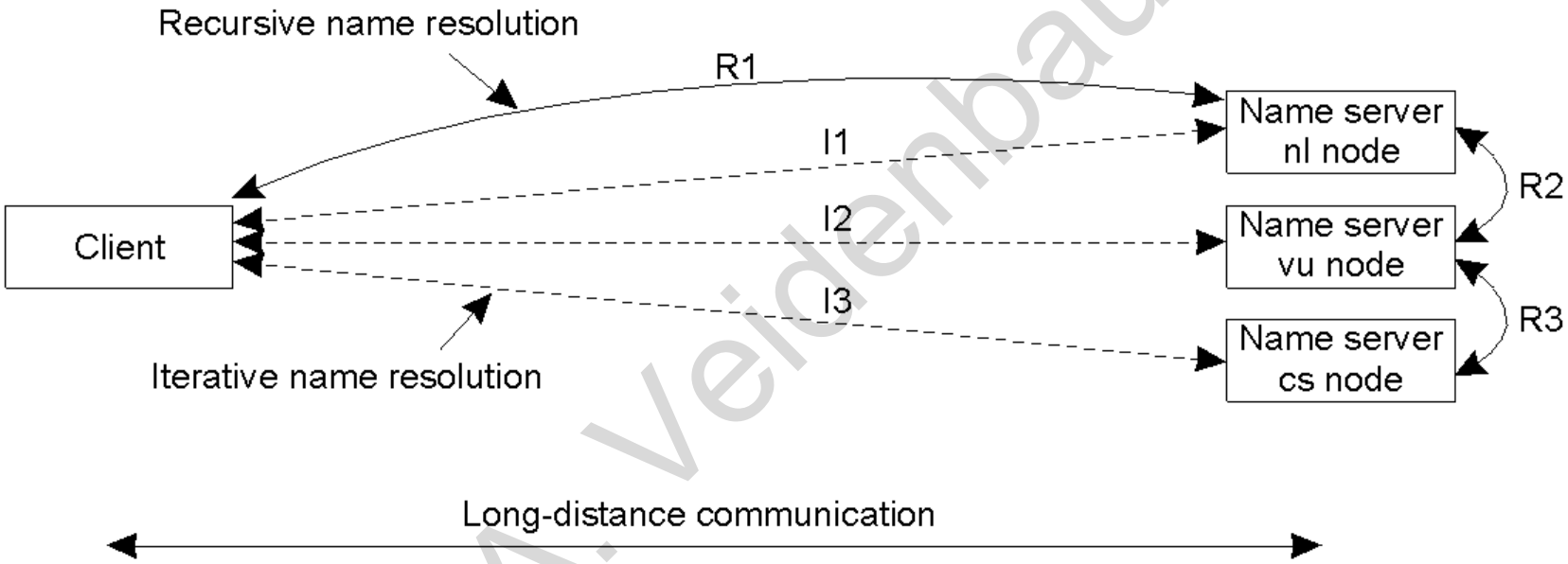


Recursive Name Resolution

- **Only result returns to client, name servers communicate**



Implementation of Name Resolution (4)



- **What is an advantage of one vs the other?**

Domain Name System

- **Name to IP address resolution**
  - **Primarily for hosts and mail servers**
- **30 years old, but working just fine**
- **Main reason: design simplicity**

The DNS Name Space

- **The most important types of resource records forming the contents of nodes in the DNS name space.**

Type of record	Associated entity	Description
SOA	**Zone**	**Holds information on the represented zone**
A	**Host**	**Contains an IP address of the host this node represents**
MX	**Domain**	**Refers to a mail server to handle mail addressed to this node**
SRV	**Domain**	**Refers to a server handling a specific service**
NS	**Zone**	**Refers to a name server that implements the represented zone**
CNAME	**Node**	**Symbolic link with the primary name of the represented node**
PTR	**Host**	**Contains the canonical name of a host**
HINFO	**Host**	**Holds information on the host this node represents**
TXT	**Any kind**	**Contains any entity-specific information considered useful**

DNS database for the zone *cs.vu.nl*.

- 3 name servers
- 3 mail servers
  - Diff. priorities
- www, ftp servers
- Laser printers

Name	Record type	Record value
cs.vu.nl	SOA	star (1999121502,7200,3600,2419200,86400)
cs.vu.nl	NS	star.cs.vu.nl
cs.vu.nl	NS	top.cs.vu.nl
cs.vu.nl	NS	solo.cs.vu.nl
cs.vu.nl	TXT	"Vrije Universiteit - Math. & Comp. Sc."
cs.vu.nl	MX	1 zephyr.cs.vu.nl
cs.vu.nl	MX	2 tornado.cs.vu.nl
cs.vu.nl	MX	3 star.cs.vu.nl
star.cs.vu.nl	HINFO	Sun Unix
star.cs.vu.nl	MX	1 star.cs.vu.nl
star.cs.vu.nl	MX	10 zephyr.cs.vu.nl
star.cs.vu.nl	A	130.37.24.6
star.cs.vu.nl	A	192.31.231.42
zephyr.cs.vu.nl	HINFO	Sun Unix
zephyr.cs.vu.nl	MX	1 zephyr.cs.vu.nl
zephyr.cs.vu.nl	MX	2 tornado.cs.vu.nl
zephyr.cs.vu.nl	A	192.31.231.66
www.cs.vu.nl	CNAME	soling.cs.vu.nl
ftp.cs.vu.nl	CNAME	soling.cs.vu.nl
soling.cs.vu.nl	HINFO	Sun Unix
soling.cs.vu.nl	MX	1 soling.cs.vu.nl
soling.cs.vu.nl	MX	10 zephyr.cs.vu.nl
soling.cs.vu.nl	A	130.37.24.11
laser.cs.vu.nl	HINFO	PC MS-DOS
laser.cs.vu.nl	A	130.37.30.32
vucs-das.cs.vu.nl	PTR	0.26.37.130.in-addr.arpa
vucs-das.cs.vu.nl	A	130.37.26.0

DNS Implementation (2)

Name	Record type	Record value
cs.vu.nl	**NS**	**solo.cs.vu.nl**
solo.cs.vu.nl	**A**	**130.37.21.1**

- **Part of the description for the *vu.nl* domain which contains the *cs.vu.nl* domain.**