CprE 450/550x
Distributed Systems and Middleware

Naming in Distributed Systems

Yong Guan
3216 Coover
Tel: (515) 294-8378
Email: guan@ee.iastate.edu
March 9, 2004

Readings for Today's Lecture

- References
  - **Chapter 4 of "Distributed Systems: Principles and Paradigms"**

Outline

- Overview: Names, Identifiers, Addresses, Routes, Name Space, Name Resolution, ...
- Implementation of a Naming Service
- Case Studies: DNS, X.500
- Naming and Mobile Entities

Some Terminology: Entities, Names, Addresses

- An **Entity** in a distributed system can be pretty much anything.
- A **Name** is a string of bits used to refer to an entity.
- We operate on an entity through its **Access Point**.
- The **Address** is the name of the access point.

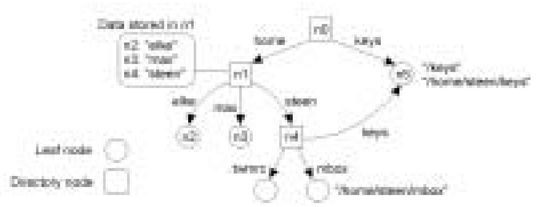
Entities, Names, Addresses: Examples

- Example
  - **Telephone as Access Point to a person.**
  - **The Telephone Number then becomes the address of the person.**
  - **Person can have several telephone numbers.**
  - **Entity can have several addresses.**
- Another Example: Transport-Level Addresses
  - **for servers this can be IP address and port number**
- Entities may change access points over time
  - **telephone numbers, e-mail addresses, IP addresses in mobile systems, ...**

Identifiers are Special Names

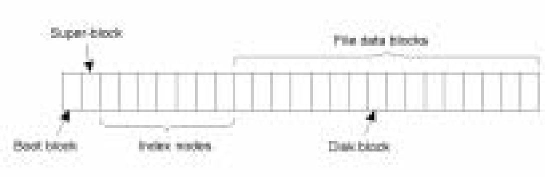
- Can we use addresses of access points as regular name for the associated entity?
  - **access points may change over time**
  - **entities may have several access points**
- **Identifiers** uniquely identify an entity:
  - **An identifier refers to at most one entity.**
  - **Each entity is referred to at most one identifier.**
  - **An identifier always referes to the same entity (never reused)**
- Example:
  - **SSN? Telephone Numbers?**

Name Spaces (1)



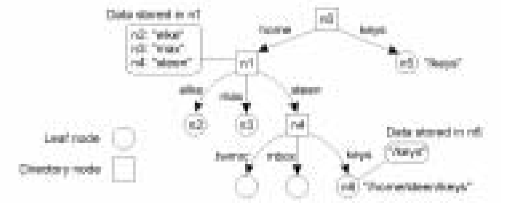
A general naming graph with a single root node.

Name Spaces (2)



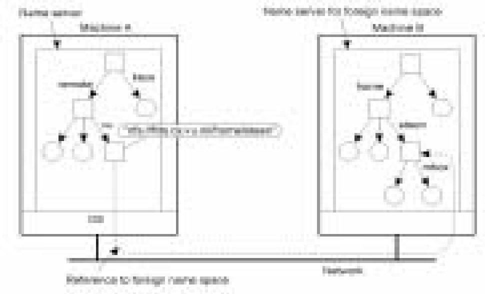
The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.

Linking and Mounting (1)



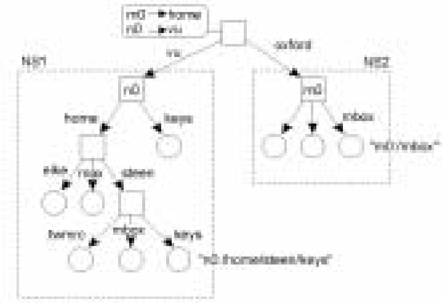
The concept of a symbolic link explained in a naming graph.

Linking and Mounting (2)


Name server
Machine A
Name server for foreign name space
Machine B
remote
keys
vu
"nfs://flits.cs.vu.nl//home/steen"
home
steen
mbox
OS
Network
Reference to foreign name space

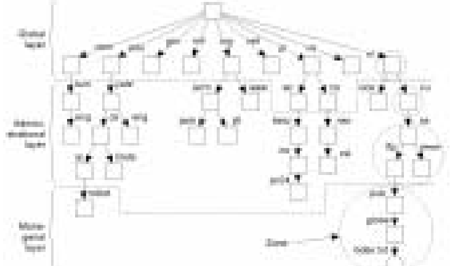
Mounting remote name spaces through a specific process protocol.

Linking and Mounting (3)



Organization of the DEC Global Name Service

Name Space Distribution (1)



An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

Name Space Distribution (2)

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

A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrational layer, and a managerial layer.

Name Resolution

- Path name

  N:<label-1, label-2, … , label-n>
- Absolute path name: first node in path name is root.
- Relative path name: first node can be any node.
- Global name vs. local name.

- Where to start name resolution? (“Closure”)
- Examples:

  Location of inode of root directory.

  Environment setting (e.g. HOME variable) to refer to home directory.

Implementation of Name Resolution

- Simplified picture:

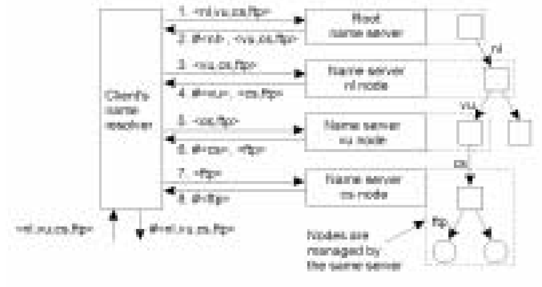
  No replication of name servers

  No client side caching
- Each client has access to local **name resolver**.
- Example: resolve

  root:<edu,iastate,ee,ftp,pub,netex,index.txt>

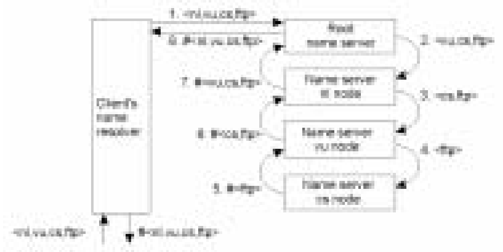
- **Iterative Resolution** *vs.* **Recursive Resolution**

Implementation of Name Resolution (1)



The principle of iterative name resolution.

Implementation of Name Resolution (2)



The principle of recursive name resolution.

Iterative vs. Recursive Name Resolution: Comparison

Iterative

- Stateless

Recursive

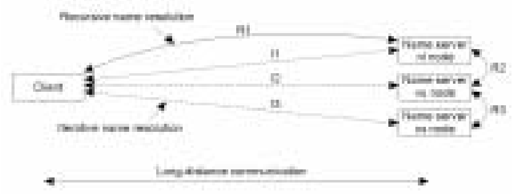
- Higher-level servers need to maintain state about resolutions. (!?)
- Caching is effective.
- Reduced communication costs.

Implementation of Name Resolution (3)

Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	--	--	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
ni	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

Recursive name resolution of <*nl, vu, cs, ftp*>. Name servers cache intermediate results for subsequent lookups.

Implementation of Name Resolution (4)


Recursive name resolution
R1
I1
I2
I3
R2
R3
Client
Name server nl node
Name server vu node
Name server cs node
Iterative name resolution
Long-distance communication

The comparison between recursive and iterative name resolution with respect to communication costs.

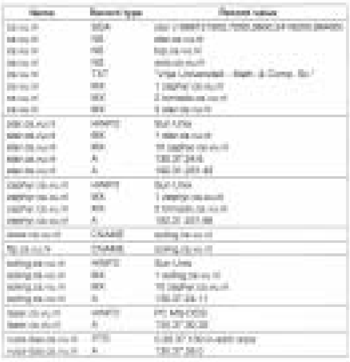
The DNS Name Space

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

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

DNS Implementation (1)

An excerpt from the DNS database for the zone *cs.vu.nl*.



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	NIS	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.

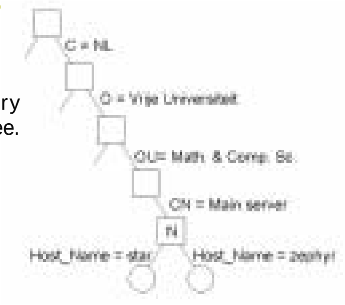
The X.500 Name Space (1)

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	L	Vrije Universiteit
OrganizationalUnit	OU	Math. & Comp. Sc.
CommonName	CN	Main server
Mail_Servers	--	130.37.24.6, 192.31.231,192.31.231.66
FTP_Server	--	130.37.21.11
WWW_Server	--	130.37.21.11

A simple example of a X.500 directory entry using X.500 naming conventions.

The X.500 Name Space (2)

Part of the directory information tree.

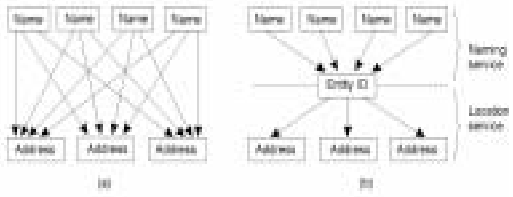


The X.500 Name Space (3)

Attribute	Value	Attribute	Value
Country	NL	Country	NL
Locality	Amsterdam	Locality	Amsterdam
Organization	Vrije Universiteit	Organization	Vrije Universiteit
OrganizationalUnit	Math. & Comp. Sc.	OrganizationalUnit	Math. & Comp. Sc.
CommonName	Main server	CommonName	Main server
Host_Name	star	Host_Name	zephyr
Host_Address	192.31.231.42	Host_Address	192.31.231.66

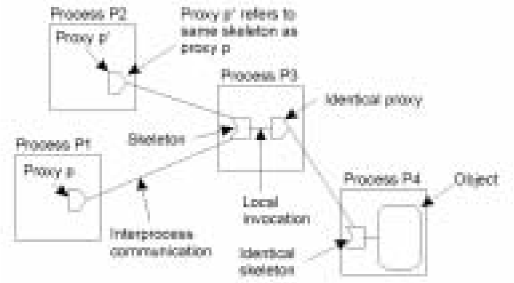
Two directory entries having *Host_Name* as RDN.

Naming versus Locating Entities



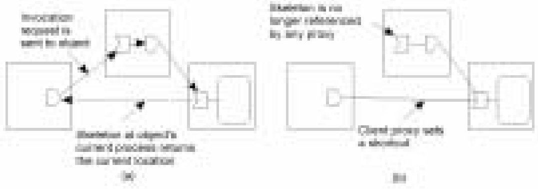
a) Direct, single level mapping between names and addresses.
b) T-level mapping using identities.

Forwarding Pointers (1)



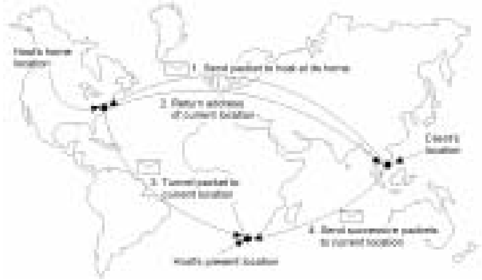
The principle of forwarding pointers using (*proxy, skeleton*) pairs.

Forwarding Pointers (2)


Invocation request is sent to object
Skeleton at object's current process returns the current location
(a)
Skeleton is no longer referenced by any proxy
Client proxy sets a shortcut
(b)

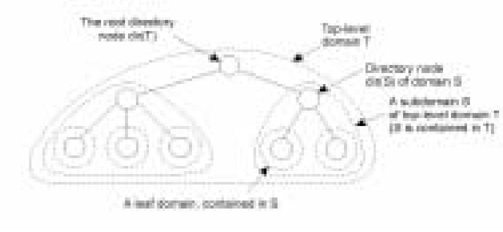
Redirecting a forwarding pointer, by storing a shortcut in a proxy.

Home-Based Approaches


Host's home location
1. Send packet to host at its home
2. Return address of current location
Client's location
3. Tunnel packet to current location
4. Send successive packets to current location
Host's present location

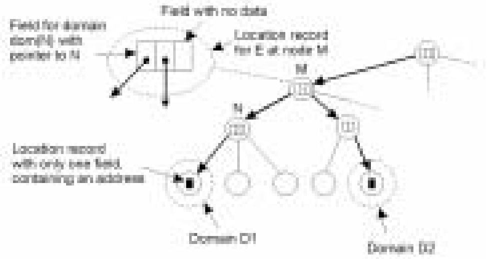
The principle of Mobile IP.

Hierarchical Approaches (1)


The root directory node dir(T)
Top-level domain T
Directory node dir(S) of domain S
A subdomain S of top-level domain T (S is contained in T)
A leaf domain, contained in S

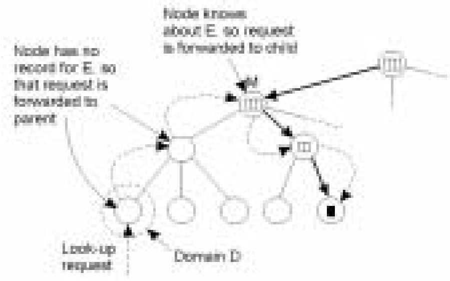
Hierarchical organization of a location service into domains, each having an associated directory node.

Hierarchical Approaches (2)


Field with no data
Field for domain dom(N) with pointer to N
Location record for E at node M
M
N
Location record with only one field, containing an address
Domain D1
Domain D2

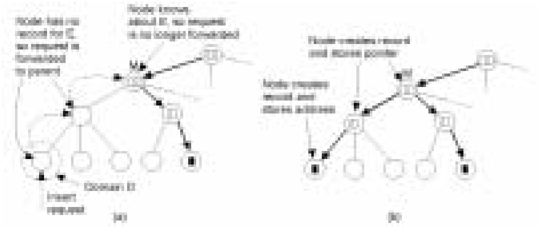
An example of storing information of an entity having two addresses in different leaf domains.

Hierarchical Approaches (3)


Node knows about E, so request is forwarded to child
Node has no record for E, so that request is forwarded to parent
M
Look-up request
Domain D

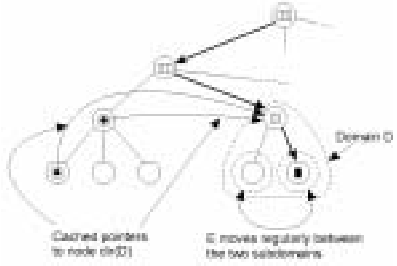
Looking up a location in a hierarchically organized location service.

Hierarchical Approaches (4)


Node has no record for E, so request is forwarded to parent
Node knows about E, so request is no longer forwarded
M
Domain D
Insert request
(a)
Node creates record and stores pointer
Node creates record and stores address
M
(b)

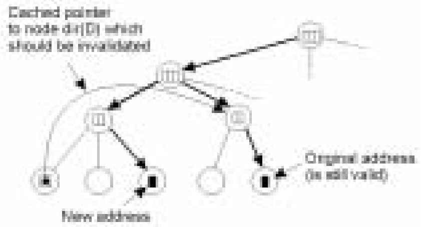
a) An insert request is forwarded to the first node that knows about entity *E*.
b) A chain of forwarding pointers to the leaf node is created.

Pointer Caches (1)


Domain D
Cached pointers to node dir(D)
E moves regularly between the two subdomains

Caching a reference to a directory node of the lowest-level domain in which an entity will reside most of the time.

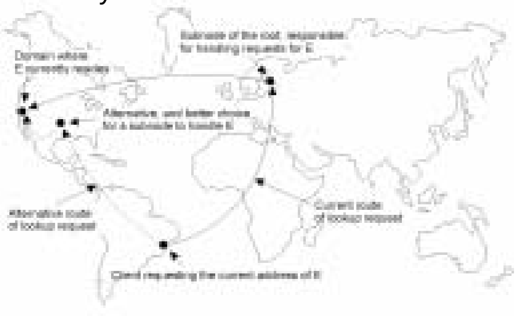
Pointer Caches (2)


Cached pointer to node dir(D) which should be invalidated
Original address (is still valid)
New address

A cache entry that needs to be invalidated because it returns a nonlocal address, while such an address is available.

37

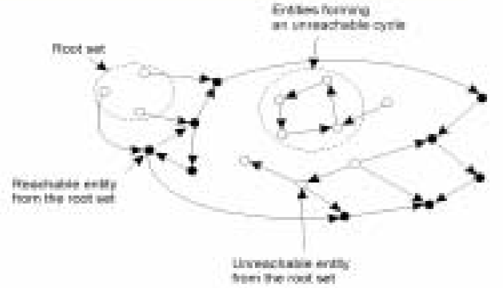
Scalability Issues


Subnode of the root, responsible
for handling requests for E
Domain where
E currently resides
Alternative, and better choice
for a subnode to handle E
Alternative route
of lookup request
Current route
of lookup request
Client requesting the current address of E

The scalability issues related to uniformly placing subnodes of a partitioned root node across the network covered by a location service.

38

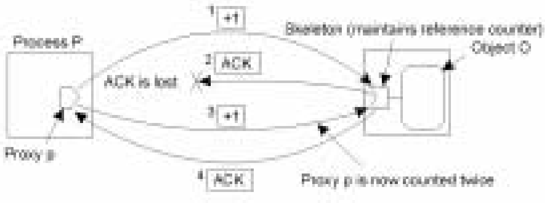
The Problem of Unreferenced Objects



An example of a graph representing objects containing references to each other.

39

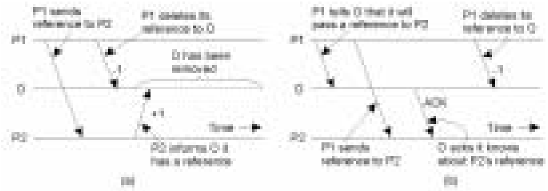
Reference Counting (1)


1 +1
Skeleton (maintains reference counter)
Process P
Object O
2 ACK
ACK is lost
3 +1
Proxy p
4 ACK
Proxy p is now counted twice

The problem of maintaining a proper reference count in the presence of unreliable communication.

40

Reference Counting (2)

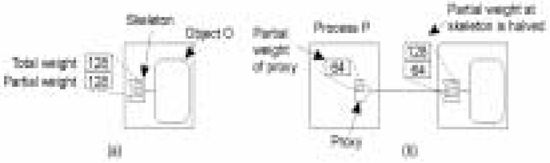


a) Copying a reference to another process and incrementing the counter too late

b) A solution.

41

Advanced Referencing Counting (1)

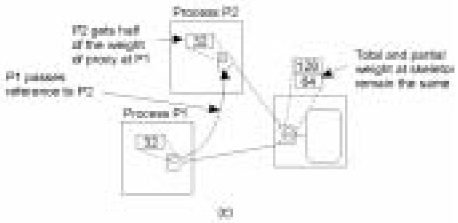


a) The initial assignment of weights in weighted reference counting

b) Weight assignment when creating a new reference.

42

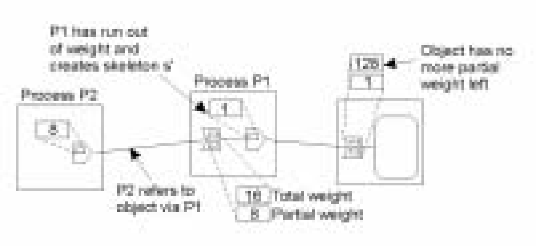
Advanced Referencing Counting (2)



c) Weight assignment when copying a reference.

43

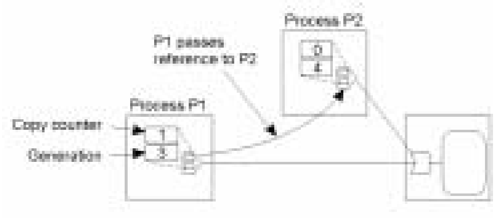
Advanced Referencing Counting (3)



Creating an indirection when the partial weight of a reference has reached 1.

44

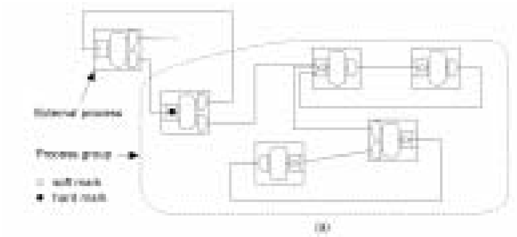
Advanced Referencing Counting (4)



Creating and copying a remote reference in generation reference counting.

45

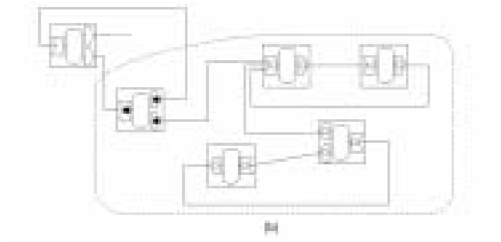
Tracing in Groups (1)



Initial marking of skeletons.

46

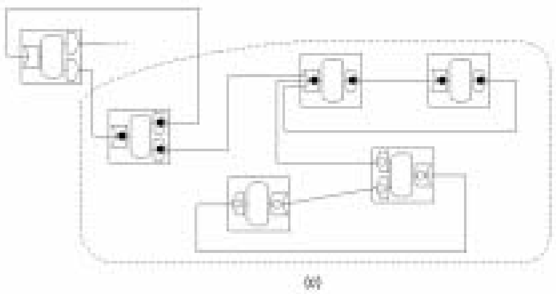
Tracing in Groups (2)



After local propagation in each process.

47

Tracing in Groups (3)



Final marking.

48

Any Questions?

See you next time.