Unit -5
Naming

- Names plays a very important role in all computer systems. They are used to share resources, to uniquely identify entities, to refer to locations, and more.
- An important issue with naming is that a name can be resolved to the entity it refers to.
- Name resolution thus allows a process to access the named entity. To resolve names, it is necessary to implement a naming system.
- The difference between naming in distributed systems and non-distributed systems lies in the way naming systems are implemented.
- In a distributed system, the implementation of a naming system is itself often distributed across multiple machines. How this distribution is done plays a key role in the efficiency and scalability of the naming system.

5.1 NAMES, IDENTIFIERS, AND ADDRESSES

- A **name** in a distributed system is a string of bits or characters that is used to refer to an **entity**.
- An entity in a distributed system can be practically anything. Typical examples include resources such as hosts, printers, disks, and files.
- Other well-known examples of entities that are often explicitly named are processes, users, mailboxes, newsgroups, Web pages, graphical windows, messages, network connections, and so on.
- Entities can be operated on. For example, a resource such as a printer offers an interface containing operations for printing a document, requesting the status of a print job, and the like.
- Furthermore, an entity such as a network connection may provide operations for sending and receiving data, setting quality-of-service parameters, requesting the status, and so forth.

- To operate on an entity, it is necessary to access it, for which we need an access point. An access point is yet another, but special, kind of entity in a distributed system.
- The name of an access point is called an **address**. The address of an access point of an entity is also simply called an address of that entity.
- An entity can offer more than one access point. As a comparison, a telephone can be viewed as an access point of a person, whereas the telephone number corresponds to an address. Indeed, many people nowadays have several telephone numbers, each number corresponding to a point where they can be reached.
- In a distributed system, a typical example of an access point is a host running a specific server, with its address formed by the combination of, for example, an IP address and port number (i.e., the server's transport-level address).

- An **entity** may change its access points in the course of time.
- For example, when a mobile computer moves to another location, it is often assigned a different IP address than the one it had before.
- Likewise, when a person moves to another city or country, it is often necessary to change telephone numbers as well.
- In a similar fashion, changing jobs or Internet Service Providers, means changing your e-mail address.

- An address is thus just a special kind of name: it refers to an access point of an entity.
- Because an access point is tightly associated with an entity, it would seem convenient to use the address of an access point as a regular name for the associated entity.
- Nevertheless, this is hardly ever done as such naming is generally very inflexible and often human unfriendly.

- In addition to addresses, there are other types of names that deserve special treatment, such as names that are used to uniquely identify an entity.
- A **true identifier** is a name that has the following properties (Wieringa and de Jonge, *1995):*

1. An identifier refers to at most one entity.
2. Each entity is referred to by at most one identifier.
3. An identifier always refers to the same entity (i.e., it is never reused).

- By using identifiers, it becomes much easier to unambiguously refer to an entity.
- For example, assume two processes each refer to an entity by means of an identifier.
- To check if the processes are referring to the same entity, it is sufficient to test if the two identifiers are equal. Such a test would not be sufficient if the two processes were using regular, nonunique, nonidentifying names. For example, the name "John Smith" cannot be taken as a unique reference to just a single person.

- Likewise, if an address can be reassigned to a different entity, we cannot use an address as an identifier.
- Consider the use of telephone numbers, which are reasonably stable in the sense that a telephone number for some time refers to the same person or organization.
- However, using a telephone number as an identifier will not work, as it can be reassigned in the course of time.
- Consequently, Bob's new bakery may be receiving phone calls for Alice's old antique store for a long time.
- In this case, it would have been better to use a true identifier for Alice instead of her phone number.

- Addresses and identifiers are two important types of names that are each used for very different purposes.
- In many computer systems, addresses and identifiers are represented in machine-readable form only, that is, in the form of bit strings.
- For example, an Ethernet address is essentially a random string of 48 bits.
- Likewise, memory addresses are typically represented as 32-bit or 64-bit strings.

- Another important type of name is that which is tailored to be used by humans, also referred to **as human-friendly names.**
- In contrast to addresses and identifiers, a human-friendly name is generally represented as a character string.
- These names appear in many different forms. For example, files in UNIX systems have character-string names that can be as long as 255 characters, and which are defined entirely by the user.
- Similarly, DNS names are represented as relatively simple case-insensitive character strings.

FLAT NAMING

- Above, we explained that identifiers are convenient to uniquely represent entities.
- In many cases, identifiers are simply random bit strings. which we conveniently refer to as unstructured, or flat names.
- An important property of such a name is that it does not contain any information whatsoever on how to locate the access point of its associated entity.

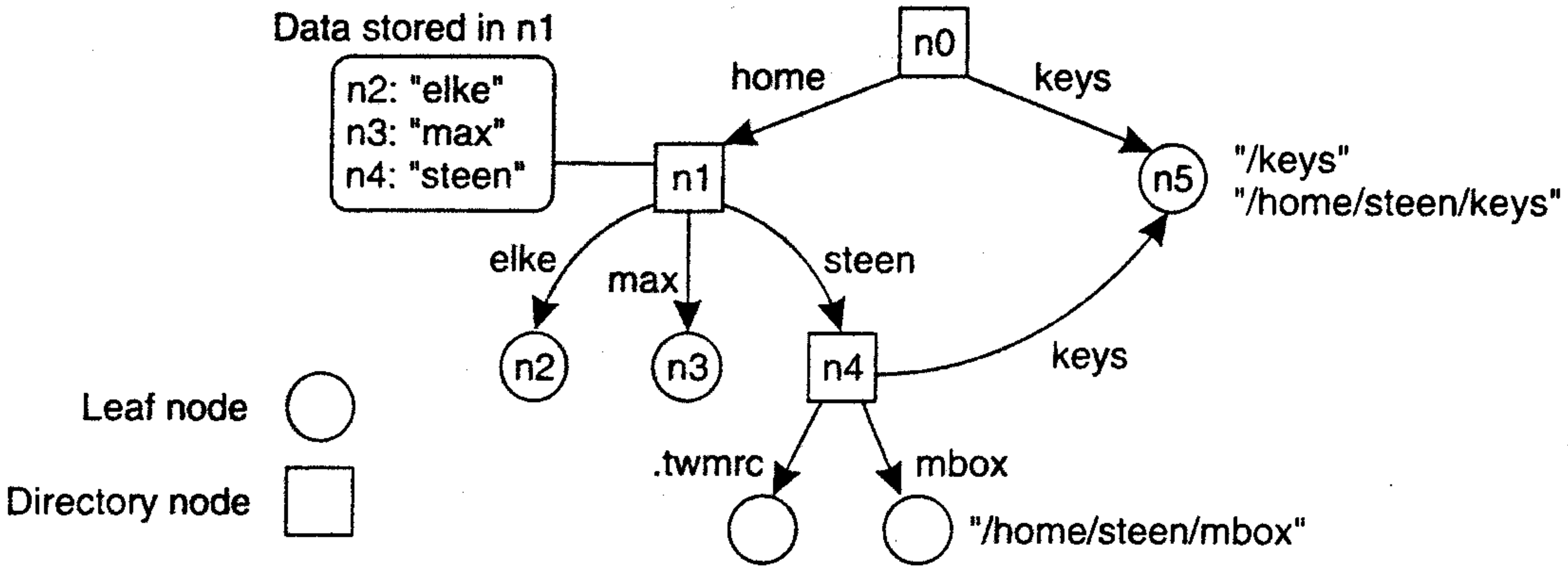
5.2 STRUCTURED NAMING

- Flat names are good for machines, but are generally not very convenient for humans to use.
- As an alternative, naming systems generally support structured names that are composed from simple, human-readable names.
- Not only file naming, but also host naming on the Internet follow this approach.

5.2.1 Name Spaces

- Names are commonly organized into what is called a **name** space.
- Name spaces for structured names can be represented as a labeled, directed graph with two types of nodes.
- A **leaf node** represents a named entity and has the property that it has no outgoing edges.
- A leaf node generally stores information on the entity it is representing-for example, its address-so that a client can access it.
- Alternatively, it can store the state of that entity, such as in the case of file systems in which a leaf node actually contains the complete file it is representing.

- We return to the contents of nodes below.
- In contrast to a leaf node, a **directory node** has a number of outgoing edges, each labeled with a name, as shown in Fig. below.
- Each node in a naming graph is considered as yet another entity in a distributed system, and, in particular, has an associated identifier.
- A directory node stores a table in which an outgoing edge is represented as a pair *(edge label, node identifier).* Such a table is called a directory table.


Data stored in n1
n2: "elke"
n3: "max"
n4: "steen"
n0
home
keys
n1
n5
"/keys"
"/home/steen/keys"
elke
max
steen
n2
n3
n4
keys
Leaf node
Directory node
.twmrc
mbox
"/home/steen/mbox"

- The naming graph shown in Fig. above has one node, namely *no,* which has only outgoing and no incoming edges. Such a node is called **the root (node**) of the naming graph. Although it is possible for a naming graph to have several root nodes, for simplicity, many naming systems have only one.
- Each path in a naming graph can be referred to by the sequence of labels corresponding to the edges in that path, such as

  Nt-clabel-I, label-2, ..., label-n>

  where *N* refers to the first node in the path.
- Such a sequence is called a path name. If the first node in a path name is the root of the naming graph, it is called an "**absolute"** path name. Otherwise, it is called a **relative** path name.

Name Resolution

- Name spaces offer a convenient mechanism for storing and retrieving information about entities by means of names.
- More generally, given a path name, it should be possible to look up any information stored in the node referred to by that name.
- The process of looking up a name is **called name resolution**.

- To explain how name resolution works, let us consider a path name such as

  Ni<label v.label g, ...label;».

- Resolution of this name starts at node *N* of the naming graph, where the name *label}* is looked up in the directory table, and which returns the identifier of the node to which *label}* refers.
- Resolution then continues at the identified node by looking up the name *label-.* in its directory table, and so on. Assuming that the named path actually exists, resolution stops at the last node referred to by *label.;* by returning the content of that node.

Domain Name System(DNS)

- The **Domain Name System** is a name service design whose main naming database is used across the Internet.
- This original scheme was soon seen to suffer from three major shortcomings:

1. It did not scale to large numbers of computers.

2. Local organizations wished to administer their own naming systems.

3. A general name service was needed – not one that serves only for looking up computer addresses.

- **Domain names •** The DNS is designed for use in multiple implementations, each of which may have its own name space.
- In practice, however, only one is in widespread use, and that is the one used for naming across the Internet.
- The Internet DNS name space is partitioned both organizationally and according to geography.
- The names are written with the highest-level domain on the right.

- The original top-level organizational domains (also called *generic domains*) in use across the Internet were:
- *com* – Commercial organizations
- *edu* – Universities and other educational institutions
- *gov – US governmental agencies*
- *mil – US military organizations*
- *net* – Major network support centres
- *org* – Organizations not mentioned above
- *int – International organizations*

New top-level domains such as *biz* and *mobi* have been added since the early 2000s. A full list of current generic domain names is available from the Internet Assigned Numbers Authority [www.iana.org I]. In addition, every country has its own domains:

- *us – United States*
- *uk – United Kingdom*
- *fr* – France
- ... – ...

DNS - The Internet Domain Name System

 A distributed naming database (specified in RFC 1034/1305)

 Name structure reflects administrative structure of the Internet

 Rapidly resolves domain names to IP addresses

– exploits caching heavily

– typical query time ~100 milliseconds

 Scales to millions of computers

– partitioned database

– caching

 Resilient to failure of a server

– Replication

Basic DNS algorithm for name resolution (domain name -> IP number)

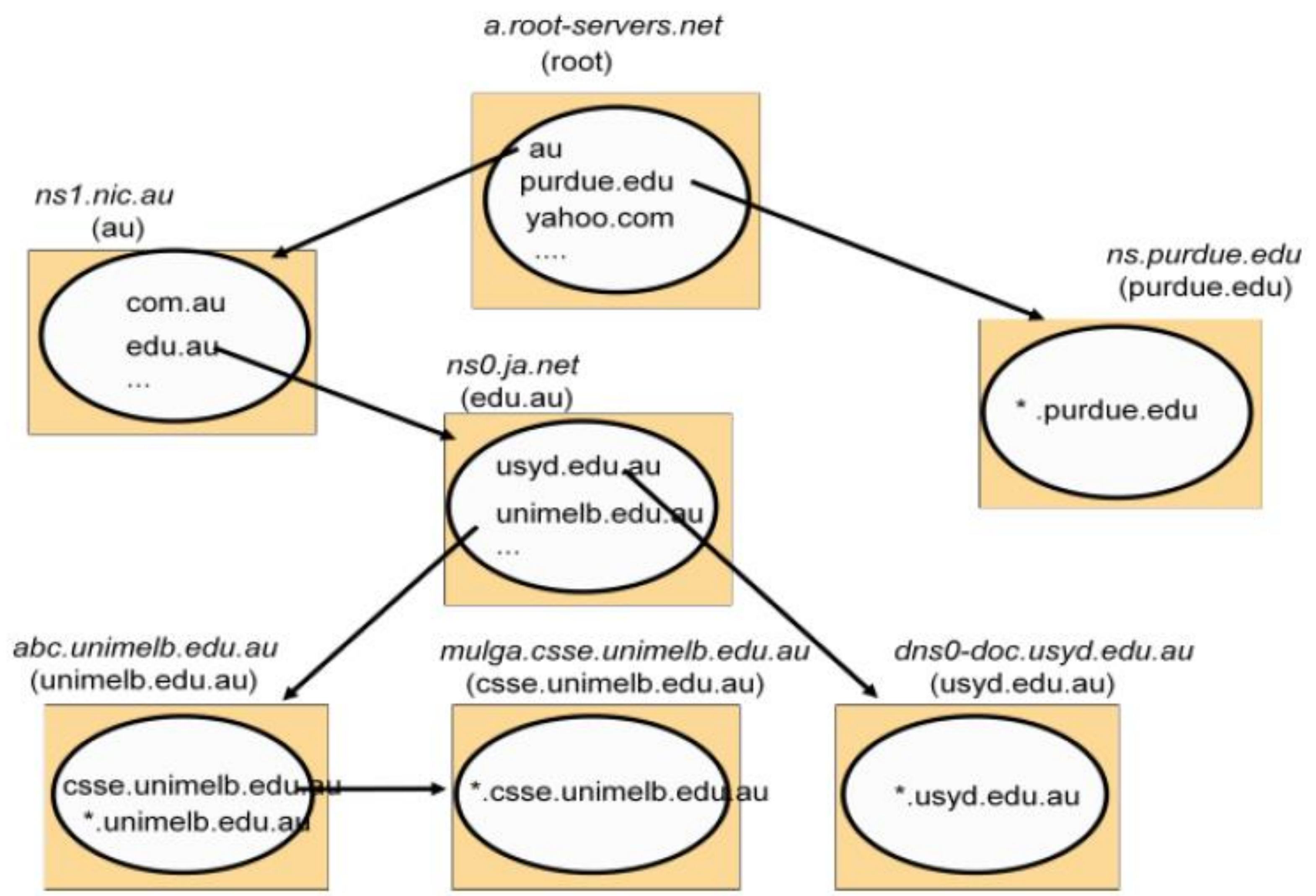
- Look for the name in the local cache
- Try a superior DNS server, which responds with:

– another recommended DNS server

– the IP address (which may not be entirely up to date)

DNS name servers: Hierarchical organization

Note: Name server names are in italics, and the corresponding domains are in parentheses. Arrows denote name server entries


a.root-servers.net
(root)
au
purdue.edu
yahoo.com
....
ns1.nic.au
(au)
com.au
edu.au
...
ns.purdue.edu
(purdue.edu)
* .purdue.edu
ns0.ja.net
(edu.au)
usyd.edu.au
unimelb.edu.au
...
abc.unimelb.edu.au
(unimelb.edu.au)
csse.unimelb.edu.au
*.unimelb.edu.au
mulga.csse.unimelb.edu.au
(csse.unimelb.edu.au)
*.csse.unimelb.edu.au
dns0-doc.usyd.edu.au
(usyd.edu.au)
*.usyd.edu.au

DNS server functions and configuration

 Main function is to resolve domain names for computers, i.e. to get their IP addresses

– caches the results of previous searches until they pass their 'time to live'

 Other functions:

– get *mail host* for a domain

– reverse resolution - get domain name from IP address

– Host information - type of hardware and OS

– Well-known services - a list of well-known services offered by a host

– Other attributes can be included (optional)

DNS issues

 Name tables change infrequently, but when they do, caching can result in the delivery of stale data. (***Stale data** is an artifact of caching, in which an object in the cache is not the most recent version committed to the **data** source. To avoid **stale data**, implement an appropriate cache locking strategy.)

– Clients are responsible for detecting this and recovering

 Its design makes changes to the structure of the name space difficult. For example:

– merging previously separate domain trees under a new root

– moving subtrees to a different part of the structure (e.g. if Scotland became a separate country, its domains should all be moved to a new country-level domain.)

5.3 Attribute based naming

- Flat and structured names generally provide a unique and location-independent way of referring to entities.
- Moreover, structured names have been partly designed to provide a human-friendly way to name entities so that they can be conveniently accessed. In most cases, it is assumed that the name refers to only a single entity.
- However, location independence and human friendliness are not the only criterion for naming entities. In particular, as more information is being made available it becomes important to effectively search for entities.
- This approach requires that a user can provide merely a description of what he is looking for.

- There are many ways in which descriptions can be provided, but a popular one in distributed systems is to describe an entity in terms of *(attribute, value)* pairs, generally referred to as attribute-based naming.
- In this approach, an entity is assumed to have an associated collection of attributes. Each attribute says something about that entity.
- By specifying which values a specific attribute should have, a user essentially constrains the set of entities that he is interested in.
- It is up to the naming system to return one or more entities that meet the user's description.

- **Directory Services**
- Attribute-based naming systems are also known as directory services, whereas systems that support structured naming are generally called naming systems.
- With directory services, entities have a set of associated attributes that can be used for searching. In some cases, the choice of attributes can be relatively simple.
- For example, in an e-mail system, messages can be tagged with attributes for the sender, recipient, subject, and so on.
- However, even in the case of e-mail, matters become difficult when other types of descriptors are needed, as is illustrated by the difficulty of developing filters that will allow only certain messages (based on their descriptors) to be passed through.

- In most cases, attribute design has to be done manually.
- Even if there is consensus on the set of attributes to use, practice shows that setting the values consistently by a diverse group of people is a problem by itself, as many will have experienced when accessing music and video databases on the Internet.

- To alleviate some of these problems, research has been conducted on unifying the ways that resources can be described.
- In the context of distributed systems, one particularly relevant development is the resource description framework (RDF).
- Fundamental to the RDF model is that resources are described as triplets consisting of a subject, a predicate, and an object.
- For example, *(Person, name,Alice)* describes a resource *Person* whose *name* is *Alice.*
- In RDF, each subject, predicate, or object can be a resource itself. This means that *Alice* may be implemented as reference to a file that can be subsequently retrieved.
- In the case of a predicate, such a resource could contain a textual description of that predicate.
- Of course, resources associated with subjects and objects could be anything. References in RDF are essentially URLs.

Hierarchical Implementations: LDAP

- LDAP (Lightweight Directory Access Protocol) is a software protocol for enabling anyone to locate data about organizations, individuals and other resources such as files and devices in a network -- whether on the public internet or on a corporate intranet.
- LDAP allows a user to search for an individual without knowing where they're located (although additional information will help with the search).
- A common approach to tackling distributed directory services is to combine structured naming with attribute-based naming.

https://youtu.be/SK8Yw-CiRHk

- This approach has been widely adopted, for example, in Microsoft's Active Directory service and other systems.
- Many of these systems use, or rely on the lightweight directory access protocol commonly referred simply as LDAP.
- The LDAP directory service has been derived from OS1's X.500 directory service.
- As with many OSI services, the quality of their associated implementations hindered widespread use, and simplifications were needed to make it useful. Detailed information on LDAP can be found in Arkills (2003).

- Conceptually, an LDAP directory service consists of a number of records, usually referred to as directory entries.
- A directory entry is comparable to a resource record in DNS.
- Each record is made up of a collection of *(attribute. value)* pairs, where each attribute has an associated type.
- A distinction is made between single-valued attributes and multiple-valued attributes. The latter typically represent arrays and lists.