OUTLINE

- Names, Aliases, Addresses, Attributes …
  - Bind, lookup, resolve …
  - URL, URI, URN …
  - Service, Resource vs. Address vs. Name
- Name Spaces
  - Finite vs. Infinite
- Name Service Discovery
  - Static, DHCP
  - Broadcast, Multicast, DHT …
- Name Service Examples:
  - NIS, NIS+
  - DNS: Classic DNS, Dynamic DNS, Multicast DNS
  - Directory Services (X500, LDAP …)
  - etc …
NAMES

• What for are the Names used in general ?
• What for are the Names used for in DS ?
• Who guarantees the uniqueness of given names ?
  • How to guarantee uniqueness in general ?
What are Names used for in DS?

- refer to a wide variety of resources such as:
  - Computers, Services, Remote objects, Remote files, Users, etc.

Names identify one entity at most

- Human readable (memorable) identifier
- One-to-One association:
  - name → computer, name → human, name → service
  - computer → name, human → name, service → name
- Associations are not interchangeable:
  - name points to the same entity all the time
- Names can have one or many aliases (many-to-one):
  - Alias1 → name1, Alias2 → name1, etc.
  - www.example.com → myhost.example.com
  - ftp.example.com → myhost.example.com

- Alias can point to another alias
- www.amazon.de → www.cdn.amazon.de → opfwww
Dependent on the design “bind” and “unbind” operations can be allowed to perform by client application:
- PyroNS, CorbaNS, RMIregistry, Dynamic DNS, DHCP ...
- Servers may ask clients to authenticate first

... in case of dynamic clients:
- The client address may change rapidly
- Once the client's address is changed, the name service has to be contacted to update the name association
  - Periodic “bind” updates are performed by clients confirming name association
  - Server may clean the name association in case there was no update from client after certain timeout
Other implementations focused on persistent name indexes (name associations do not change rapidly)

In this case “bind”, “unbind” is prohibited to perform from application side

- The separate authority entity has to be contacted to introduce new name association, modify or remove.

Name indexes in this case are persistent

- No periodic updates required
- Records are not cleaned by server

Example: DNS
In order to guarantee uniqueness the NS relies on specific name space

Name space can be continuous or finite

- Finite name space systems:
  - UNIX user base (32 character limit)
  - Broadcast, Multicast ... (ARP, NetBEUI)
    - Size of MAC address ?
- IPv4, IPv6 (size of IP address ? How may addresses in total ?)
- DHT (Distribute Hash Tables)
  - Tor, Freenet, Torrent, KAD, ...
  - ... just the matter of hash size

Hierarchical name space

Simple way to implement infinite (continuous) name spaces
HIERARCHICAL NAME SPACE

- **Root** ( '/' in UNIX, C: D: - windows, TLD “.” in DNS )
- **Directory tree** (/usr/bin, C:\Python2.7, ut.ee domain)
- **Tree leaves**
  - /usr/bin/python, C:\Python2.7\python.exe, math.ut.ee hostname
- **Relative and Absolute paths**
  - ~/mail and /home/username/mail
- **Local names (context base)**
  - Math server can be referred as “math” from inside of “ut.ee”
  - google.com has particular IPs for each geographical region
- **Alias-names (pointers to some existing) name**
  - Symlinks in file systems, CNAMEs in DNS
DOMAIN

- Part of the name space, which is managed by one logical entity
- It is typical for hierarchical name spaces, where each domain is maintained by specific name server:
  - Easier management
    ✓ Divide and conquer
  - Name space ordering
    ✓ Faster lookups
  - Functional extensions
    ✓ Introducing new services in the scope of domain
- Name space can be described by name components
  - artjom.lind@ut.ee, http://www.ut.ee
  - here we refer to DNS name space in both cases
  - ut.ee is a part of both examples
NAME VS ADDRESS

- Pure name contains only the identifier of the object
- Non-pure name contains other information about the object
  - For example, Address is non-pure name as it contains the location of the object
- Pure names have to be looked up before we can access the object
- Having an address, we can access the object right away
  - But what happens to an address if we relocate the object?
- Address examples:
  - https://courses.cs.ut.ee/2016/ds/fall/Main/Lectures
  - info@ut.ee
ADDRESSING NOTATIONS

**URI: Uniform Resource Identifiers**
- identify resources on the Web, and other Internet resources
- URIs are ‘uniform’ in that their syntax incorporates that of indefinitely many individual types of resource identifiers

**Service specific names**
- Classic Internet Resource name URI:
  - proto:// DNS : + port/ + path? + args
- Resource in
  - Tor: proto://<160-char hash>.onion: + port/ + path? + args
  - Freenet: <just a big hash> / path? + args
- What about Fidonet, Bitnet?
  - 1:350/201@fidonet.org
  - Is it URI? or what kind of notation is it?
A Typical FidoNet address: 1:350/201@fidonet.org
- Zone: 1
- Region: 17
- Network: 350
- Node: 201
- Net: fidonet.org

Zone: Zone 1 is North America
Region: Part of a Zone (NOT part of the addressing scheme)
Network: A group of local Nodes (unique number in a zone)
Node: An individual BBS (unique number in a net)
Point: A User who is set up like a BBS off your Node
Some URIs contain information that can be used to locate and access a resource; others are pure resource names. The familiar terms

- Uniform Resource Locator (URL) is often used for URIs that provide location information and specify the method for accessing the resource.
  - For example, the URI: `https://courses.cs.ut.ee/2016/ds/fall/Main/Lectures` gives a location of course web page
- Uniform Resource Names (URNs) are URIs that are used as pure resource names rather than locators.
  - For example, the URI: `urn:ISBN:0-201-62433-8` identifies a book
<table>
<thead>
<tr>
<th>Names</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Firstname, Lastname, uid, gid, password hash</td>
</tr>
<tr>
<td>Service</td>
<td>Address, version</td>
</tr>
<tr>
<td>Host</td>
<td>Architecture, OS, address, owner</td>
</tr>
<tr>
<td>Alias</td>
<td>Name</td>
</tr>
<tr>
<td>Group</td>
<td>&lt;name1, name2, ... &gt;</td>
</tr>
<tr>
<td>Directory</td>
<td>&lt;element1, element2, ... &gt;</td>
</tr>
</tbody>
</table>
ATTRIBUTE BASED NAME SPACE

- Idea: attributes are for query specification
  - Any attribute can be used for lookup
- Name by itself is also an attribute of an object
- Example: X.500 and LDAP
  - "$ ldapsearch -x uid=username"
  - "$ ldapsearch -x email=firstname.lastname@domain.xx"
NAME SERVICE EXAMPLES

- **NIS, NIS+**
  - The Network Information Service, or NIS is a client–server directory service protocol for distributing system configuration data such as user and host names between computers on a computer network. Sun Microsystems developed the NIS; the technology is licensed to virtually all other Unix vendors.

- **DNS ...**

- **LDAP**
  - The Lightweight Directory Access Protocol is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network ... allowing the sharing of information about users, systems, networks, services, and applications ... provide any organised set of records, often with a hierarchical structure, such as a corporate email directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.

- **JNDI**
  - The Java Naming and Directory Interface (JNDI) is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name. Like all Java APIs that interface with host systems, JNDI is independent of the underlying implementation.

ACCORDING TO WIKIPEDIA.ORG
The NIS, released by Sun in 1980s, was the first “prime time” administrative database to address these problems;

It was originally called the Sun Yellow Pages, but eventually had to be renamed for legal reasons. Many vendors have licensed Sun’s code, making NIS the most widely-supported network database system;

It is a distributed database system that replaces copies of commonly replicated configuration files with a centralised management facility;

Instead of having to manage each host’s files, you maintain one database for each file on one central server.
NIS

- Network Information System
- SUN old standard for the information exchange between UNIX systems
- Finite name space, 1 domain
- Domain – the machines using same authentication info
- Domain has one primary server and multiple secondary servers
- Data updates occur in the primary server the secondaries hold the copies and share them
- Clients send the getpwent(), getent() … requests to the server over the network
- The NIS protocols rely on SunRPC remote procedure calls
- No security (plain protocol, without encryption)
An NIS server is a machine storing a set of maps that are available to network machines and applications.

NIS master server contains the set of maps that you, the NIS administrator, create and update as necessary.

Each NIS domain must have one, and only one, master server.

NIS Maps
• NIS maps are essentially two-column tables. One column is the key and the other column is information value related to the key. NIS finds information for a client by searching through the keys. Some information is stored in several maps because each map uses a different key.
NIS TABLES

- passwd.byname, passwd.byuid
- group.byname, group.byuid
- publickey.byname
- hosts.byname, hosts.byaddr
- mail.byaddr, mail.aliases
- services.byname, service.bynumber
- rpc.byname, rpc.bynumber
- protocols.byname, protocols.bynumber
- networks.byname, networks.byaddr
- netmasks.bymask, netmasks.byaddr
- ethers.byname, ethers.byaddr
NIS+

- Next generation of NIS
  - The complete new design
- Domains are hierarchical
- Each domain has again the primary and secondary servers
- The sub-domains can have different administrative division
- Added security
- Access rights can be applied to the domain hierarchy and name tables
- User space name tables are optional (in addition to the system space name tables)
Security maintained by secured version of RPC (SecureRPC) – each query is authenticated and encrypted
- Encryption using DES
- Authentication using Diffie-Hellmann key exchange

Client and Server know each other (using host keys)

Users do login into client host using username and password

Users also have NIS+ password
- Usually same as the login password

DES key is computed from the user's password (using 40bit entropy)

Client requests his Diffie-Hellman key (encrypted) from public NIS+ table

Client decrypts it and gets access to NIS+ server
NIS+ OBJECTS

- Directory – used to form the hierarchical domains
  - org_dir – administrative data directory
  - groups_dir – group membership directory
- Table – stored in directory, used for the data storage
- Group – stored in groups_dir, used to store the group access rights
- Link – used to create direct connection in NIS+ name space
- Record – either table or directory record
- Private – for internal usage
NIS+ ACCESS

- Subjects: users, hosts, groups
- Access rights defined by subject, each record can have own access rights
- World – all the authenticated users
- Nobody – authenticated and unauthenticated users
- Permission can be set separately for owner, group, world or nobody
The Domain Name Service (DNS) is a fundamental component of the Internet: it maps host names to IP addresses (and vice-versa), which allows the use of www.debian.org instead of 5.153.231.4 or 2001:41c8:1000:21::21:4.

DNS Provides:

- Mapping from names to addresses and vice versa
- Mechanism to store and retrieve information in a global data store
- Where to send mail for a domain
- Geographical information
- etc.
Basic DNS tools:

Using the host command:

```
# host ut.ee
AHs-Mac-mini:~ AH$ host ut.ee
ut.ee has address 193.40.5.73
ut.ee mail is handled by 20 frida.it.da.ut.ee.
ut.ee mail is handled by 20 berta.it.da.ut.ee.
```

```
# host www.ut.ee
AHs-Mac-mini:~ AH$ host www.ut.ee
www.ut.ee has address 193.40.5.73
www.ut.ee has IPv6 address 2001:bb8:2002:500::42
```

Using the host IPv6

```
# host 2001:bb8:2002:500::42
2.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.5.0.2.0.0.2.8.b.b.0.1.0.0.2.ip6.arpa domain name pointer www.ut.ee.
```
Hierarchical Internet name system

General requests:
- Host name based IP lookup
- Domain mailserver lookup
- IP address based host name resolution

Zone – the subtree in hierarchical name space

Specific domains for IP based hierarchy
- in-addr.arpa
- ip6.arpa
DNS

- DNS built:
  - Components:
    - Name space
    - Servers making that name space available
    - DNS database
    - Forms a tree structure
DNS

• DNS built:

• DNS is hierarchical

• DNS administration is shared

• This distribution of administration is called “delegation”
DNS

• How DNS works

1. "I NEED DIRECTION TO WWW.UT.EE"

2. "I CANNOT FIND THE DOMAIN WWW.UT.EE IN MY DATABASE, I’LL CHECK ANOTHER DNS SERVER"

3. DNS SERVER

4. DNS SERVER

5. "YES I HAVE IT IN MY CACHE IT IS MAPPED TO THIS IP ADDRESS: 172.167.36.44"

6. "THANKS I GOT IT I CAN ACCESS NOW"

7. "AWESOME I WILL CACHE IT TOO FOR WHILE IN CASE SOMEONE ASK FOR IT AGAIN"

RESOLVER / QUERY
DNS

• Query detail with tcpdump

```bash
$ sudo -s
passwd:
# tcpdump -s1500 -n port 53
```

```
AHs-Mac-mini:~ AH$ sudo tcpdump -s1500 -n port 53
tcpdump: data link type PKTAP
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on pktap, link-type PKTAP (Packet Tap), capture size 1500 bytes
09:02:43.982601 IP 172.17.164.117.58374 > 193.40.5.39.53: 15716+ A? dr-eu.skype-cr.akadns.net. (43)
09:02:43.982719 IP 172.17.164.51108 > 193.40.5.39.53: 3184+ AAAA? dr-eu.skype-cr.akadns.net. (43)
09:02:43.983816 IP 193.40.5.39.53 > 172.17.164.117.58374: 15716 8/10/7 A 40.127.143.123, A 40.127.180.92, A 104.41.212.225, A 137.135.247.208, A 138.91.55.206, A 23.102.43.250, A 40.113.91.50, A 40.127.103.135 (507)
09:02:43.996398 IP 193.40.5.39.53 > 172.17.164.117.51108: 3184 0/1/0 (109)
```
DNS

- Query detail and analysis:
  - you can use www.wireshark.org
DNS

- Resolver configuration:
  - How does your computer which DNS server to send the query to get information:

  - location “/etc/resolv.conf”
  - if you check it you will find:
    - nameserver a.b.c.d
    - nameserver ip:v6:ad:dr:es:ss

    IS THE IP/IPV6 OF A FUNCTIONING DNS SERVER.
DNS SERVER REPLICATION

- The servers of each domain is replicated

- Primary server – the one where the data can be changed
- Secondary server
- Zone replication
- notify
- Dynamic updates
- Buffering servers
- Response global TTL (time-to-live)
# RESOURCE RECORD TYPES

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>marks Start Of a zone of Authority</td>
</tr>
<tr>
<td>NS</td>
<td>NameServer record</td>
</tr>
<tr>
<td>A</td>
<td>IPv4 Address record</td>
</tr>
<tr>
<td>AAAA</td>
<td>IPv6 Address record</td>
</tr>
<tr>
<td>CNAME</td>
<td>Canonical name (i.e. an alias)</td>
</tr>
<tr>
<td>MX</td>
<td>Mail Exchanger record</td>
</tr>
<tr>
<td>SRV</td>
<td>Service Location record</td>
</tr>
<tr>
<td>PTR</td>
<td>Pointer (most commonly for reverse DNS)</td>
</tr>
<tr>
<td>TXT</td>
<td>Text record (free form text with no semantics)</td>
</tr>
<tr>
<td>NAPTR</td>
<td>Naming Authority Pointer Record</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSIG</td>
<td>Transaction Signature (RFC 2845)</td>
</tr>
<tr>
<td>TKEY</td>
<td>Transaction Key (RFC 2930) - estab secret keys</td>
</tr>
<tr>
<td>AXFR</td>
<td>Zone Transfer</td>
</tr>
<tr>
<td>IXFR</td>
<td>Incremental Zone Transfer (RFC 1995)</td>
</tr>
<tr>
<td>OPT</td>
<td>Opt pseudo RR (RFC 2671 - EDNS0)</td>
</tr>
</tbody>
</table>
Zone: example.com

@ 3600 IN SOA master.example.com. hostmaster.example.com. ( 1001514808 ; serial 10800 ; refresh (3 hours) 3600 ; retry (1 hour) 604800 ; expire (1 week) 3600 ; minimum (1 hour) )

86400 IN NS ns1.example.com.
86400 IN NS ns2.example.com.
86400 IN MX 10 mail1.example.com.
86400 IN MX 20 mail2.example.com.
ns1 86400 IN A 10.1.1.1
ns2 86400 IN A 10.1.1.2
www 900 IN A 10.1.2.2
mail1 3600 IN A 10.3.3.3
mail2 3600 IN A 10.3.3.4
DNS ISSUES

- DNS Cache Poisoning
  - Forgery: respond before the intended nameserver
  - Redirection of a domain's nameserver
  - Redirection of NS records to another target domain

- DNS Hijacking
  - Response to non-existent domains
  - Rogue DNS servers

- In particular
  - MX hijacking
  - Entire domain redirection
  - Take a large .COM offline
  - Complete spoofing of a bank's DNS info

DNS ISSUES

DNSSEC

- Protocol extension granting security of DNS responses
- Zone signing and record signatures
- Protection against fake responses
- No encryption implemented
  - Does not protect against DoS
- Trusted anchor (anchors), certificate sequence between client and trusted entity, which controllable by the resolver
DNSSEC

- Data authenticity and integrity by signing the Resource Records Sets with private key
  - Public DNSKEYs published, used to verify the RRSIGs
- Children sign their zones with their private key
  - Authenticity of that key established by signature/checksum by the parent (DS)
- Repeat for parent...
- Not that difficult on paper
  - Operationally, it is much more complicated
DNSSEC RESOURCES TYPES

- **CERT** – certificate
- **SIG, PRSIG** – signatures
- **(KEY), DNSKEY** – keys
- **TSIG, TKEY** – transaction signature and key
- **DS** – signer entity delegation
- **NSEC, NSEC3** - “next secure”, for proofing an absence of record
MULTICAST DNS

- For internal usage in local networks with no external connectivity
- For locating the hosts
- For locating the services
- So called zero configuration
- Each host has a DNS server (UDP port 5353)
- Domain local
- Each host can publish information about the services of this host

Example queries
  - _workstation._tcp.local
  - _services._dns-sd._udp.local

Example responses PTR rhn
  - [00:03:47:a4:64:d5]._workstation._tcp.local
MULTICAST DNS AVAILABILITY

- **Multicast DNS client**
  - Mac OS 9.2
  - Mac OS X
  - iPhone & iPod touch
  - Linux
  - Bonjour for Windows

- **Lookup**
  - "laserwriter.local." into your Web browser
  - "ssh mymac.local." into a terminal window
**LDAP**

- **X.500 names**
  - `dc=ut,cd=ee,ou=People,cn=usrename`
  - `/C=EE/O=AS Sertifitseerimiskeskus/CN=Juur-SK`
- **Client can search by any attribute**
- **LDAP schema describes used object classes and attributes**
- **Examples:**
  - UNIX user info
  - Windows user info (Active Directory info)
  - X.509 certificates
- **Replication**
- **Client – Server channel protected by SSL encryption**
JNDI

- Java Naming and Directory Interface
- No separate name service, but Java interface (javax.naming) to some particular name service implementation
  - LDAP, DNS, NIS, NDS, CORBA, COS RMI
  - Windows register, file system
- Name contexts, sub-contexts
  - Something similar to LDAP tree
  - Context, SubContext
- All names are context relative
  - InitialContext
JNDI API

- `void bind(String stringName, Object object)`
- `void rebind(String stringName, Object object)`
- `Object lookup(String stringName)`
- `void unbind(String stringName)`
- `void rename(String stringOldName, String stringNewName)`
- `NamingEnumeration listBindings(String stringName)`
- `NamingEnumeration list(String stringName)`
Dynamic DNS (DDNS) is a method of automatically updating a name server in the Domain Name System (DNS), often in real time, with the active DNS configuration of its configured hostnames, addresses or other information.

Implemented in Apple's CUPS printing system for automatic printers discovery in local network.

According to wikipedia.org
Example 1 - Registering a Service

```java
import com.apple.dnssd.*;

class MyRegistrar implements RegisterListener {
    void registerWebService() {
        DNSSDRegistration reg = DNSSD.register("Me!","_http._tcp",80,this);
    }

    public void serviceRegistered(
        DNSSDRegistration reg,
        int flags, String serviceName,
        String regType, String domain) {
        System.out.println("Registered service " + serviceName);
    }
}
```
Example 2 - Discovering Services

```java
import com.apple.dnssd.*;

class MyBrowser implements BrowseListener {
    void browseForPrinters() {
        DNSSDSERVICE browser = DNSSD.browse("_ipp._tcp", this);
    }

    public void serviceFound(
        DNSSDSERVICE browser, int flags,
        int ifIndex, String serviceName,
        String regType, String domain) {
        System.out.println("Found service "+serviceName);
    }
}
```
import com.apple.dnssd.*;

class MyResolver implements ResolveListener {
    void resolveService(String service,
        String type, String domain) {
        DNSSD.resolve(0,0,service,type,domain,this);
    }

    public void serviceResolved(
        DNSSDSERVICE resolver,
        int flags, int ifIndex,
        String fullName, String hostName,
        int port, TXTRecord txtRecord) {
        System.out.println("Resolved service to "
            + hostName);
    }
}
QUESTIONS
"QUIZZ

Password: DNSDNS "
HOME READING

— Chapter 15 Coordination and agreement
— Chapter 16 Transactions and Concurrency Control
— Chapter 17 Distributed Transactions
— Questions:
  — How multicast based reliable delivery is related to coordination?
  — What does it mean “network partition” issue?
  — What is the idea behind transactions?
  — What does it mean “serial equivalence”?
  — What is the difference between “flat” and “nested” transactions?