Distributed Operating Systems

Name no more precise →

Interesting/advanced Topics in Operating Systems

- **scalability**
- **systems security**
- **modeling**

Some overlap with „Distributed Systems“ (Prof Schill)

In some cases no written material.

Several lectures presented by research-group members.

Strongly requested: register for mailing list

Most Questions: mail to mailing list

Distributed OS

Hermann Härtig

Scalability in Computer Systems

DNS/BIND as a first case study

13/04/15

TECHNISCHE UNIVERSITÄT DRESDEN

Outline and Goal of Lectures on Scalability

Outline:

- scalability: terminology, problems
- basic approaches
- case studies

Goal:

- understand some of the important principles how to build scalable systems

Outline and Goal of today's Lecture

Outline:

- scalability ...
- names in Distributed Systems:
  purposes of naming, terminology
- application of scalability approaches on name resolution

Goal:

- understand some of the important principles how to build scalable systems
- ...using DNS as an example

More Case Studies

- memory consistency
- Locks
- Advanced synchronization (Paul Mc Kenney)
- file systems
- load balancing (Mosix) and HPC

General Definition: Scalability

Scalability:

- the ease with which a system or component can be modified to fit the problem area
  http://www.sei.cmu.edu/str/indexes/glossary/

Dimensions of Scalability:

- size (more CPUs)
- other resources (memory)
- software (versions, better libs, etc.)
- heterogeneity (different hardware / SW = portability)

More specific: Scalability in Computer Systems

- **A system is described as scalable** if it remains effective when there is a significant increase in the number of resources and the number of users. (Coulouris, Dollimore, Kindberg: Distributed Systems)

- **Scalability** [in telecommunication and software engineering] indicates the capability of a system to increase performance under an increased load when resources (typically hardware) are added (Wikipedia)

Scaling down

- a system is scalable if it works well for very large and very small numbers

Definition(Wang, Xu 98):

- A computer system (HW + SW) is called *scalable* if it can *scale up* (improve its resources) to accommodate ever increasing performance and functionality demand and / or *scale down* (decrease resources) to reduce cost.

A SW engineering aspect of scalability

Not subject of the course

Prepare for change in functionality

- software engineering
- choose sufficiently large logical resources
- provide hooks for extension

Problems for Scalability in Distrib./Par. Systems

- performance bottlenecks / Amdahl's Law
- failures / abuse
- administration

Amdahl's Law

- f: fraction of computation that can be enhanced
- Speedup: $\frac{\text{original execution time}}{\text{enhanced execution time}}$
- S: speedup factor for f
- $\text{Speedup}(f,S) = \frac{1}{\left(1 - f + \frac{f}{S}\right)}$

Consequences: Amdahl's Law

- attack the common case
- if S becomes VERY large, speedup approaches $\frac{1}{(1-f)}$
- interpretation for parallel systems:
  - P: section that can be parallelized
  - 1-P: serial section
  - N: number of CPUs
  - Speedup(P,N) $= \frac{1}{\left(1-P+\frac{P}{N}\right)}$

Principles to achieve Scalability ("RPC")

- identify and address bottlenecks
- partitioning
  - split systems into parts that can operate independently to a large extent
- replication
  - provide several copies of components
    - that are kept consistent eventually
    - that can be used in case of failure of copies
- locality (caching)
  - maintain a copy of information that is nearer, cheaper/faster to access than the original

Principles to achieve Scalability (2)

- specialize functionality/interfaces
- right level of consistency
  - caches, replicates, ... need not always be fully consistent
- lazy information dissemination
- balance load

Some Challenges

- balance load
  - keep load under reasonable threshold
    - at each component
    - in the communication subsystems
  - load balancing can be static or dynamic. Will study a detailed example for dynamic load balancing later(MosiX).
- minimize the delay induced by “RPC”
- prepare for change
- information dissemination
  - choose right degree of consistency

Case study: DNS

- some numbers of growth...

Roles: Names, Identifiers, Addresses

- names
  - symbolic
  - have a meaning for people

- identifiers
  - identifies a component (uniquely)
  - are used by programs

- addresses
  - locates a component
  - can change

Name resolution

- name resolution:
  - map symbolic names to objects
  - better: to a set of attributes such as:
    identifiers, addresses, other names, security properties

- Principle interface:
  - Register (Name, attributes, ...)
  - Lookup (Name) -> attributes

Related

- compilers
  - statically map names to addresses
- dynamic libraries
  - dynamically remap addresses
- port mapper
  - map service to port

Name resolution is a form of dynamic mapping of pathnames to attributes.

Observation

Many services, tools, ... provide their own name resolution

- file systems (UNIX: path names to I-Nodes)
- login
- RPC (remote procedure call) systems (portmapper)

Purpose of Directory Services

- integration of name services
- generic name service
- world-wide use of names

Today mostly used:

- email/web
- computer attributes (IP addresses)
- people attributes (certificates, …)
- ...

A Bit of History

- UUCP/MMDF (cum grano salis):
  - ira!gmdzi!oldenburg!heinrich!user (path to destination)
  - user@ira!heinrich%gmdzi
    (mixing identifiers and path information)
- ARPA-Net:
  - a single file: hosts.txt
  - maintained at Network Information Center of SRI (Stanford)
  - accessed via ftp
  - TCP/IP in BSD Unix => chaos name collisions, consistency, load
- DNS: Paul Mockapetries (84) …

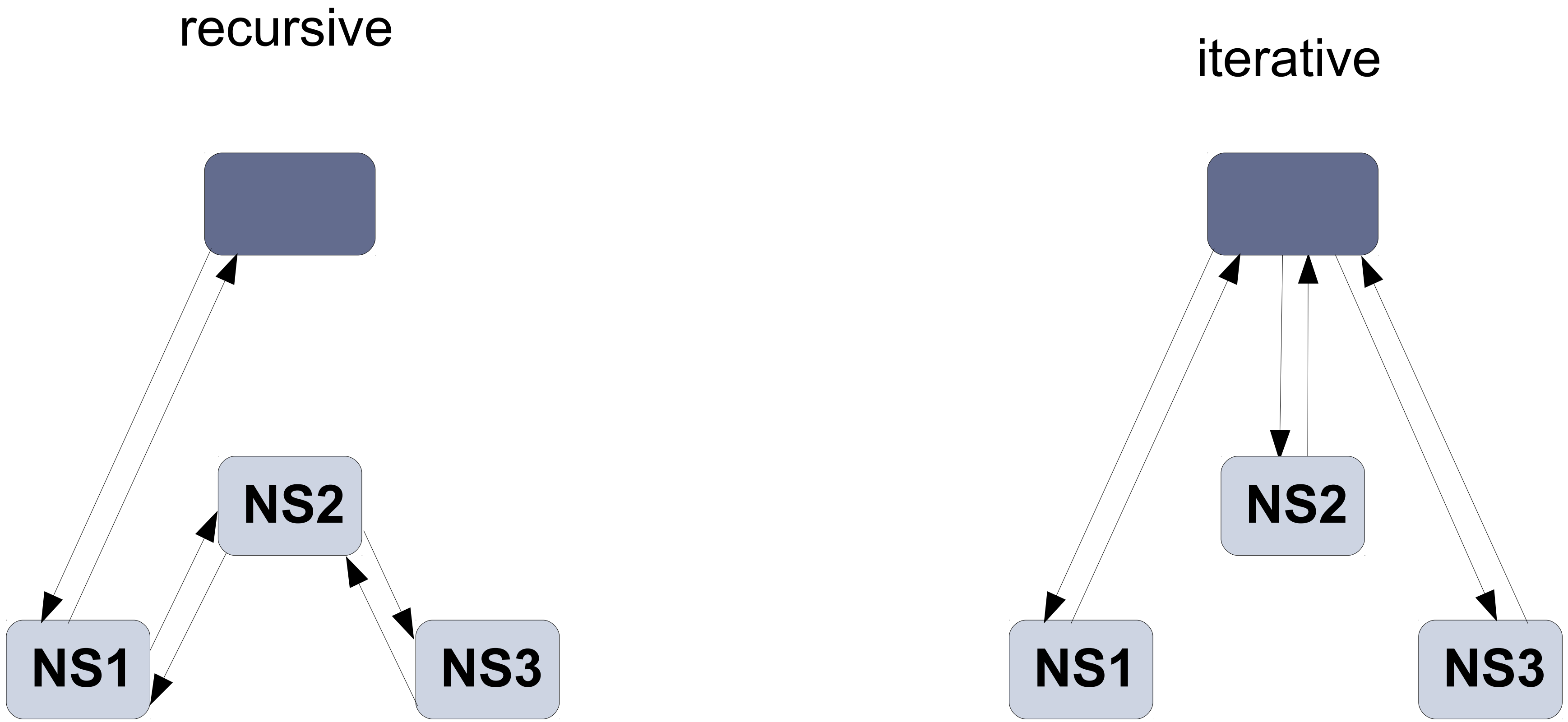
More Terminology

- name space
  - set of names recognized by a name service
- context
  - unit for which a name can be mapped directly
- aliases
  - several names for one object

More Terminology

- naming domain
  - subtree in the hierarchy of DNS contexts
- zone
  - (aka Zone of authority) Subset of a domain over which an authority has complete control. Subzones (starting at apices of a zone) can be delegated to other authorities.
- navigation
  - querying in a set of cooperating name spaces

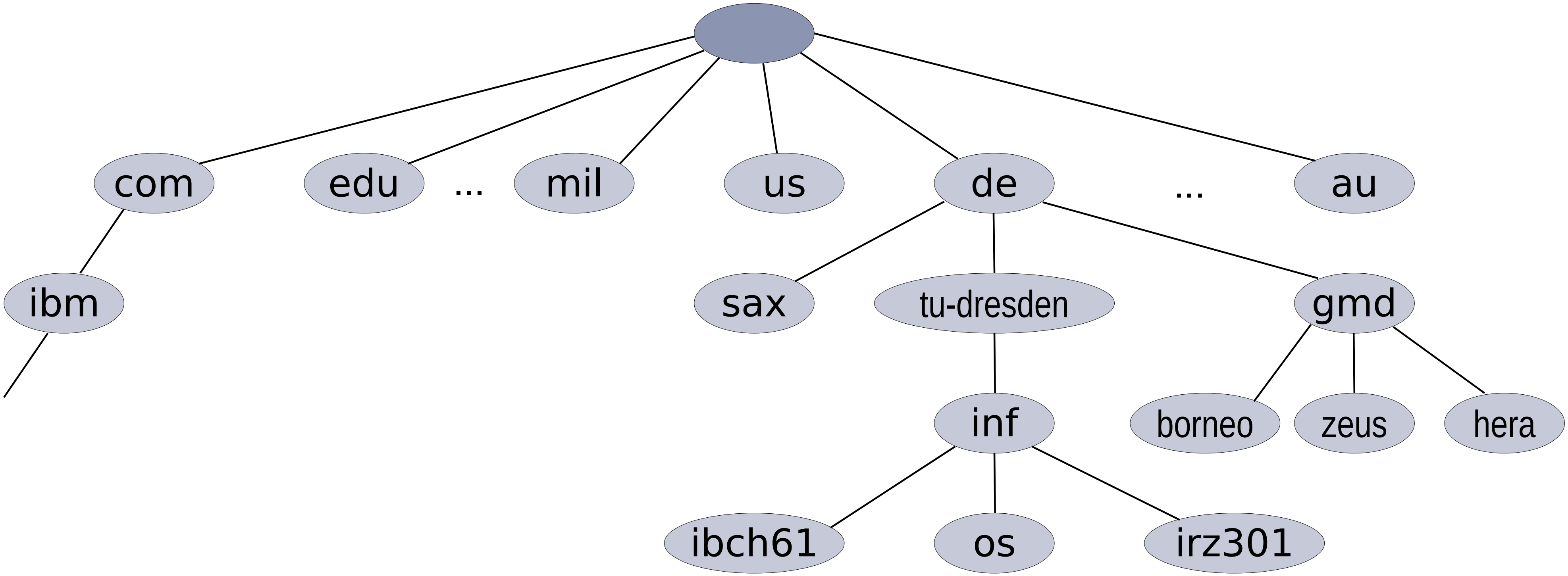
Basic Implementation Variants



Requirements / Properties

- arbitrarily large numbers
- arbitrary units of administration
- long living names, the higher in the hierarchy the longer
- high robustness
- restructuring of name spaces
- consistency
- efficiency

DNS Name Space (original)



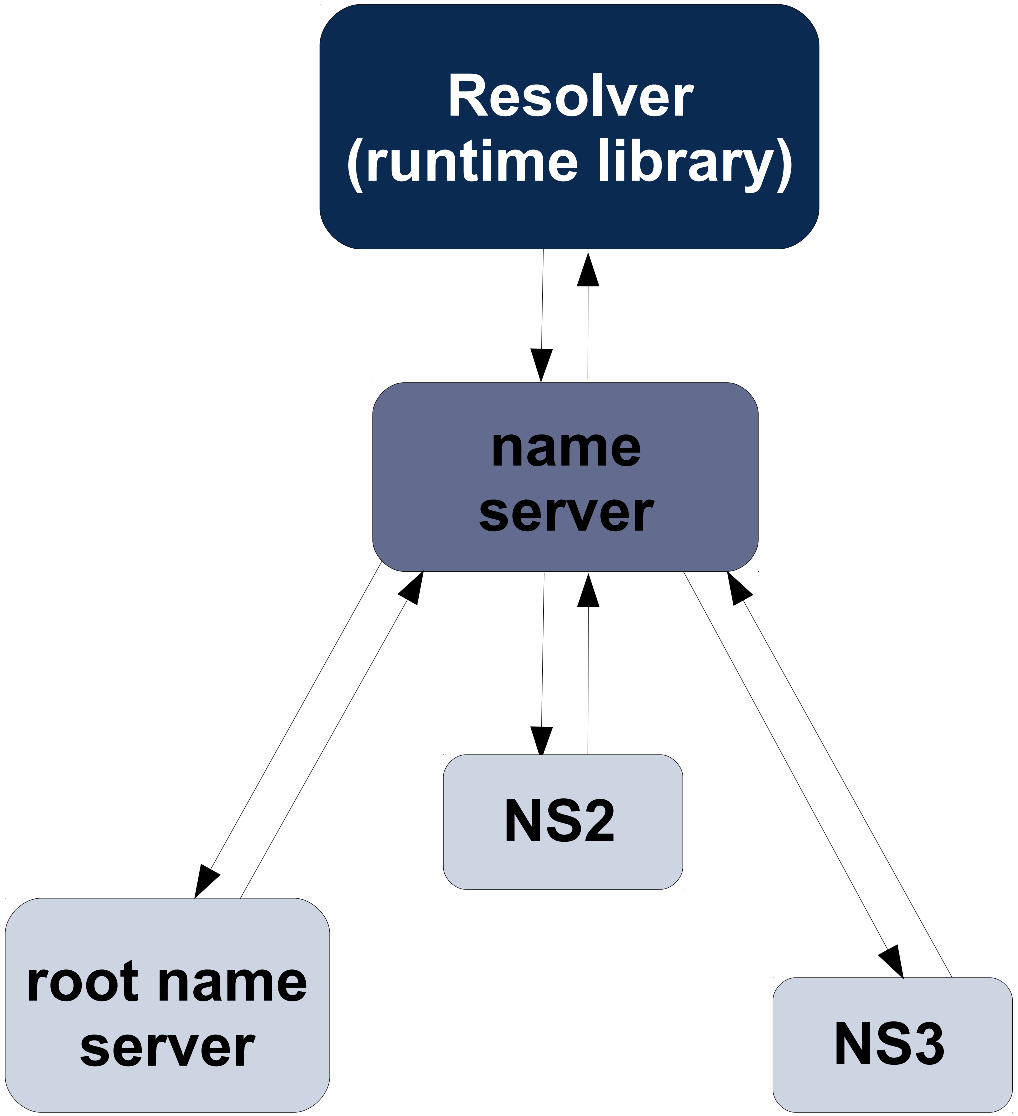
Examples

- inf.tu-dresden.de — domain
- os.inf.tu-dresden.de — computer
- heidelberg.ibm.com — domain

- ftp ftp.inf.tu-dresden.de
  - DNS: → IP address: 141.76.2.3
  - ftp daemon: IP address, port 21

- properties:
  - location independent
  - not very deep

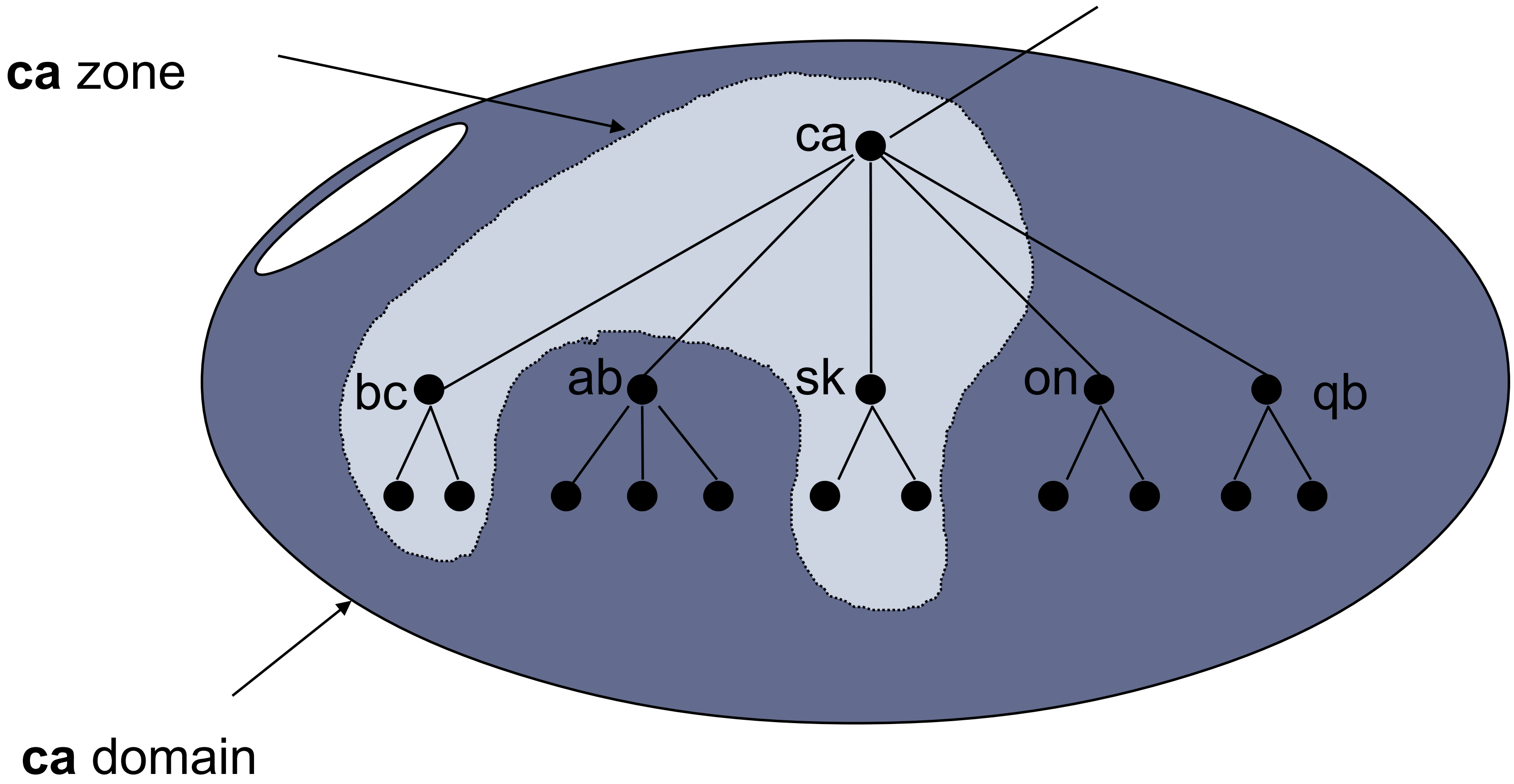
Implementation Structure (BIND)



Partitions: Zones

- Zones:
  - administrative unit
- Name Server:
  - maps to names and addresses of name servers responsible for sub zones
  - maintains management data
  - process doing the name resolution for one zone
- Resource records (RR):
  - key interface

Partitions: Zones



Example taken from: Coulouris et al., Distributed Systems

Replication

- currently 13 root name servers

- each zone has at least
  - one primary
  - one secondary

  name server

Caching

- each name server caches resource records
- time to live attribute
- authoritative versus non-authoritative answers

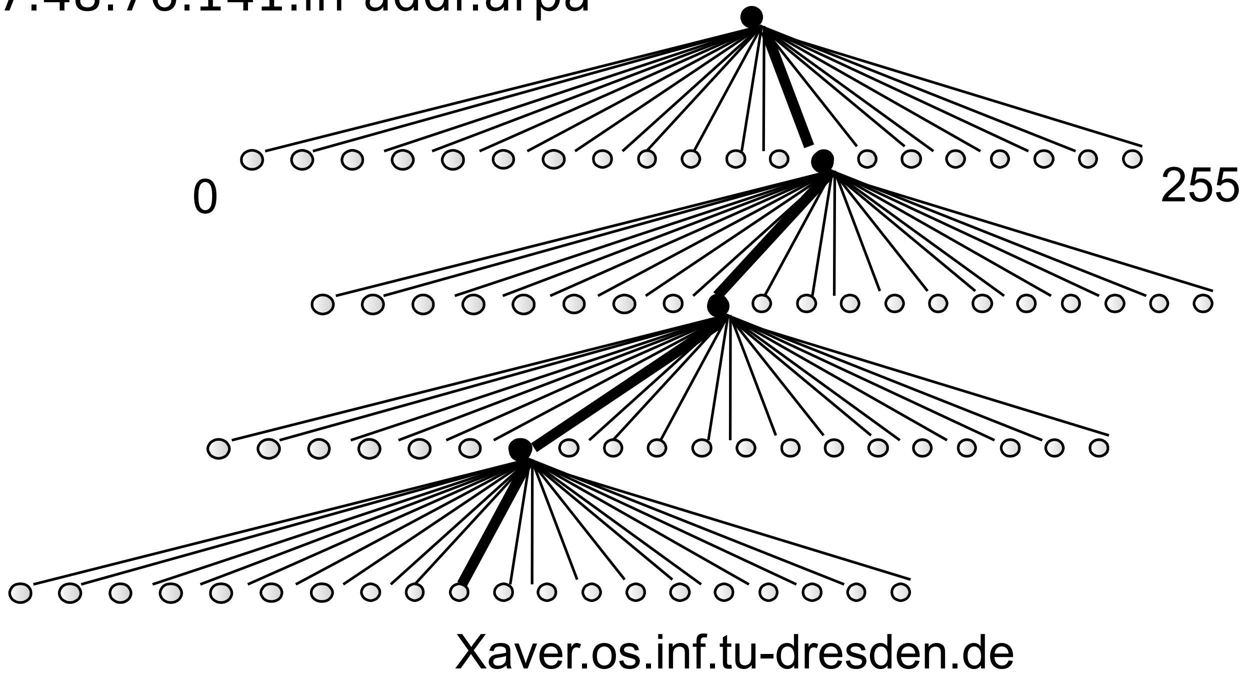
Resource Records

Record type	Interpretation	Content
A	address	IPv4 address
AAAA	address	IPv6 address
NS	Name server	DNS name
CNAME	Symbolic link	DNS name of canonicial name
SOA	Start of authority	Zone-specific properties
PTR	IP reverse pointer	DNS name
HINFO	Host info	Text description of host OS
...	...	...

Reverse Resolution

Example

- IP-Address: 141.76.48.97
  - DNS-Name: 97.48.76.141.in-addr.arpa



Summary: Scalability and DNS

- Good points:
  - replication and caching work well
  - over time, DNS scaled from small numbers to millions

- Bad Points:
  - IP addresses too small
  - no integrated systems security (!!!)

Literature

- **Paul Albitz & Cricket Liu**
  DNS and BIND
  O´Reilly & Associates, Inc.

- **Mark Hill, Michael Marty**
  Amdahl's Law in the Multicore Era
  IEEE