Web-Technologies

- Chapters
  - Client-Side Programming
  - Server-Side Programming
  - Web-Content-Management
  - Web-Services
  - Apache Webserver
  - Robots, Spiders and Search engines
    - Robots and Spiders
    - Search engines in general
    - Google

Web Services 1

- What means “Web Service” ?
- Architecture
- Examples
- Current and future use

Web Services 2

- What means „Web Service“?

  A Web service corresponds to the XML based representation of an application or a software component. Web services are describing; Interface and their meta data are separated. Consumer and service provider of a Web service communicate by means of XML based messages over defined interfaces. Details of the implementation of the Web service remain hidden.

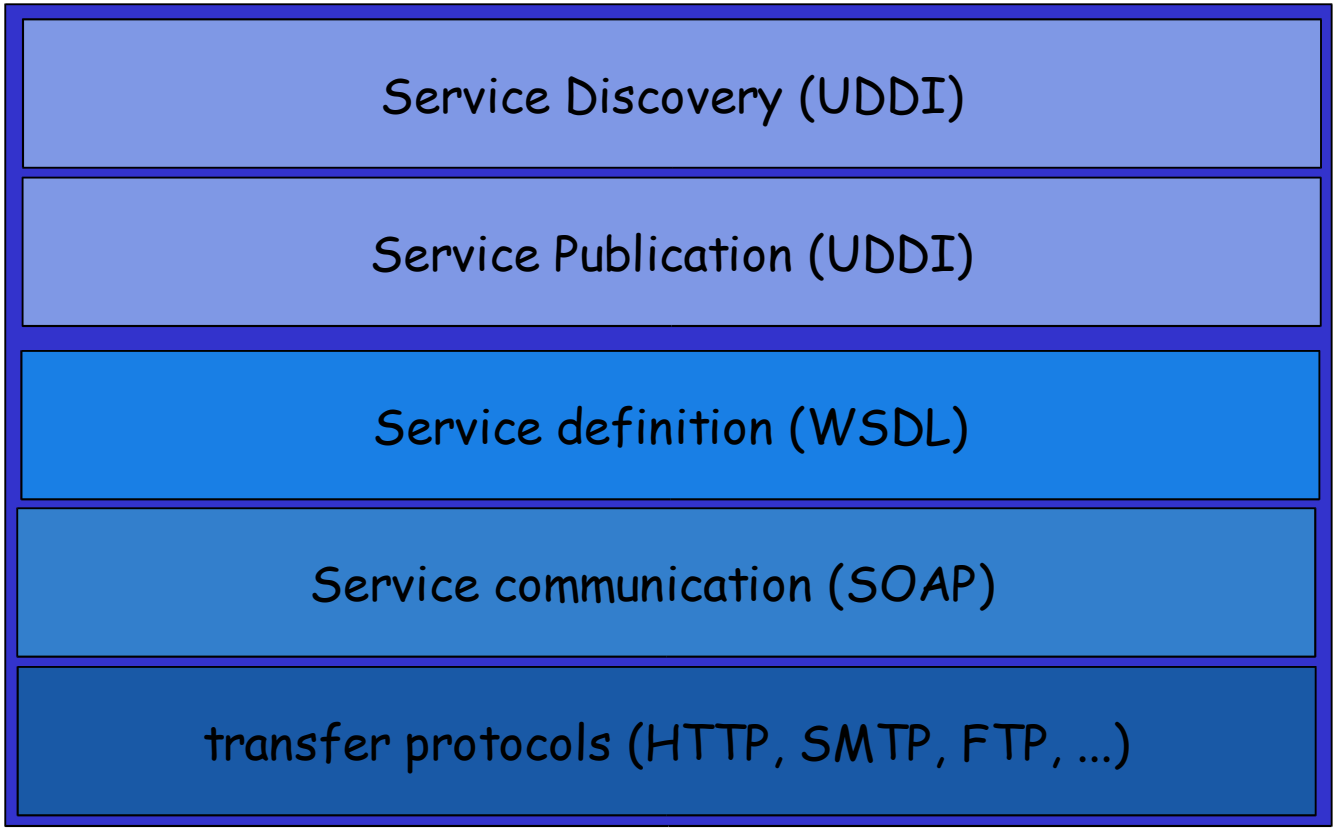
  Notice: An equally used definition of „Web Services“ is still in discussion, also if there is a common understanding about it within the W3C.

Web Services 3

- Architecture
  - Global (and local) directories are used to publish sites with Web services: *Universal Description, Discovery and Integration (UDDI).*
    Like „yellow pages“ here all web services are registered with data about their use, owners and interfaces.
  - The *Web Service Description Language (WSDL)* defines commands, attributes and type-definitions for a given web service.
  - With the *Simple Object Access Protocol (SOAP)* an application and a web service will communicate, using the definitions as given by the WDSL.
  - It‘s not defined which transfer protocol (like HTTP) is beeing used with SOAP.

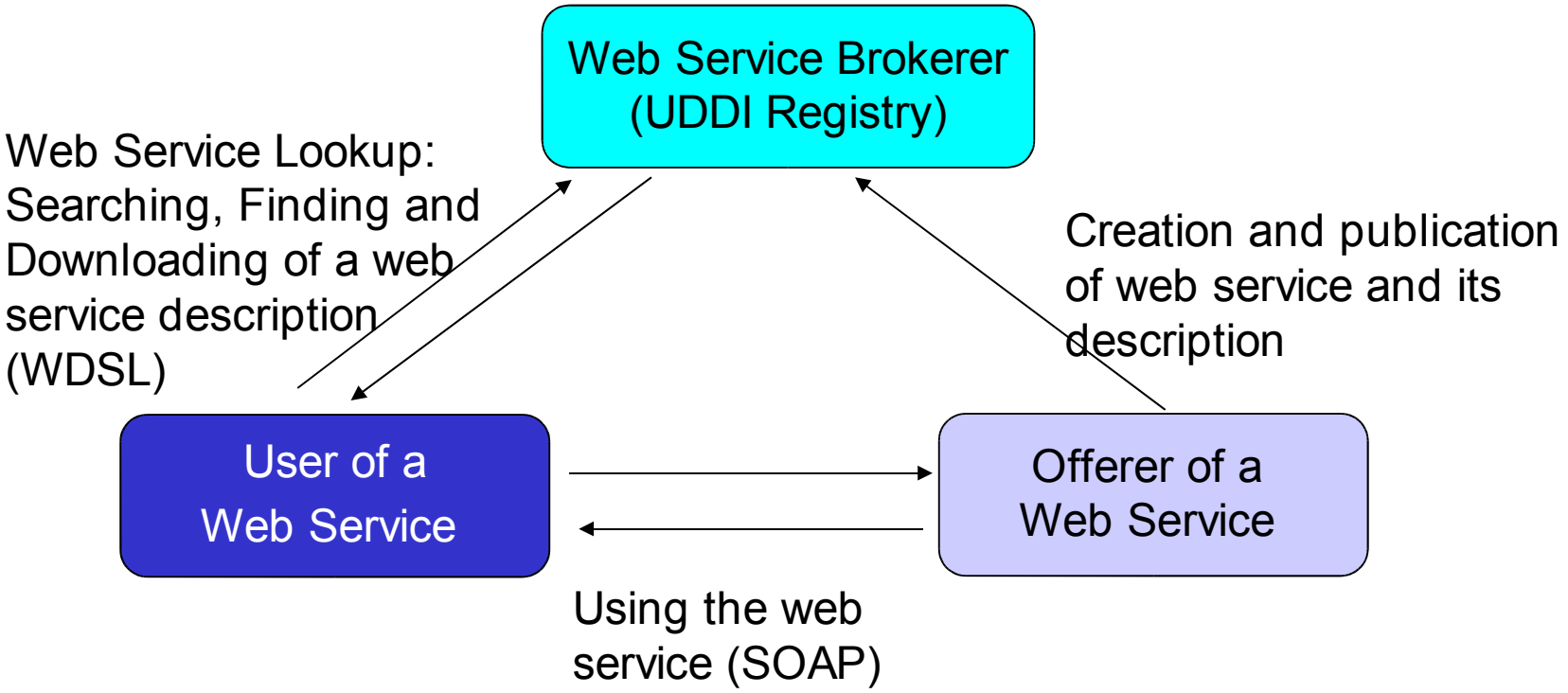
Web Services 4

- Architecture



Web Services 5

- Architecture
  - Use of a web service



Web Services 6

- Examples (Communication using SOAP)
  - SOAP request:

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap; charset=utf-8

<?xml version="1.0"?>
<soap:Envelopexmlns:soap=http://www.w3.org/2001/12/soap-envelope
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock" />
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Web Services 7

- Examples (Communication using SOAP)
  - SOAP response:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/soap; charset=utf-8
Date: Sat, 12 May 2002 08:09:04 GMT

<?xml version="1.0"?>
<soap:Envelopexmlns:soap=http://www.w3.org/2001/12/soap-envelope
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   <soap:Body xmlns:m="http://www.stock.org/stock" />
      <m:GetStockPriceResponse>
         <m:Price>34.5</m:Price>
      </m:GetStockPriceResponse>
   </soap:Body>
</soap:Envelope>
```

Web Services 8

- Current and future use
  - Currently big commercials like Microsoft, SUN, IBM and others are trying to establish “web services” as a new innovational service.
  - Extensions useable for SOAP and WDSL in e-Business: *Electronic Business Extensible Markup Language (ebXML)*
  - There are still open questions within the definition of the architecture. Mostly cause of bad communication between members of W3C and OASIS (*Organization for the Advancement of Structured Information Standards*) and cause of marketing aspects
  - Several commercials and non-profit organisations are currently working on applications using “web services”. Example: Google’s Web Service Interface
  - Future of “web services” ? Still unknown; “Application Service Providing“ was a hype too...

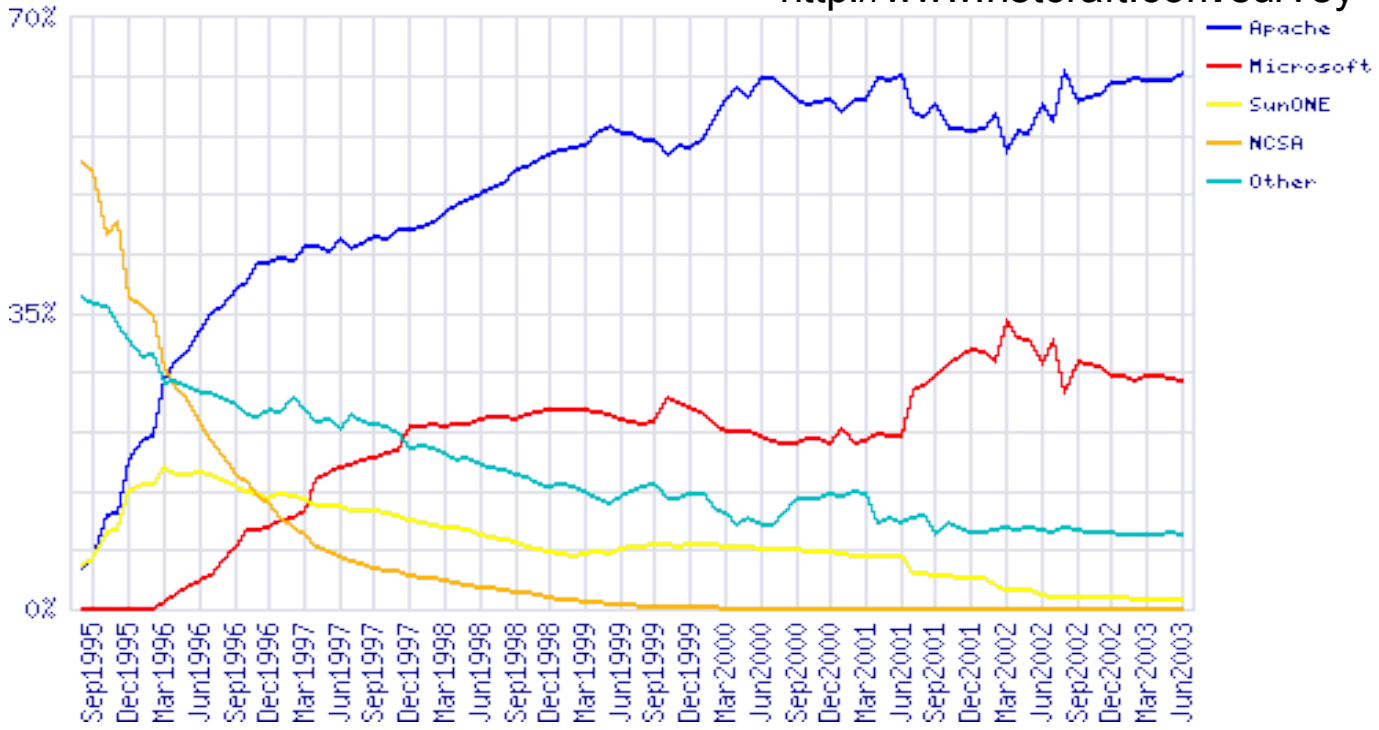
Apache 1

- Apache („a patchy server“)
  - Free HTTP server, supports HTTP/1.1 (RFC2616)
  - Available on nearly all OS (but not Mac)
  - Built upon NCSA httpd (V1.3) since 1994. First release of Apache: April 1995, V 0.6.2 as beta
  - First public version in December 1, 1995
  - Developer-Team consists of volunteers – open source project
  - Today the #1 webserver on the internet
  - Current version (Jun 2003): 1.3.27 as final and 2.0.46 as current release
  - http://www.apache.org

Excurse Apache 2

- Apache (cont.)
  - Currently used by appr. 63% of all servers in use. (MS-IIS: 27%)
  - About 41 Million sites tested

http://www.netcraft.com/survey



Excurse Apache 3

- Principle:
  - After start Apache will listen to requests on port 80 (or any other defined port)
  - Configuration is stored within a textfile „httpd.conf“, which is read by the httpd-process
  - On a request it will fork itself;
  - The child-process will answer the request, close the connection and then die
    - Before sending an answer, the process will parse the requesting URL and look it up for errors.
    - If the request aims a special filetype (like a server-parsed SSI-document), needed moduls are dynamically loaded or called

Excurse Apache 4

- Sample configuration file (extract)

```
Listen 131.188.3.67:80
ServerName www.rrze.uni-erlangen.de
User www
Group www
PidFile logs/httpd.pid
ServerRoot /usr/local/apache
MaxClients 220
...
LoadModule vhost_alias_module libexec/mod_vhost_alias.so
...
AddModule mod_vhost_alias.c
...
```

Excurse Apache 5

- Sample configuration file (cont.)

```
...
NameVirtualHost 131.188.3.67

<Virtualhost 131.188.3.67>
ServerName www.techfak.uni-erlangen.de
User someone
Group somegroup
DocumentRoot /proj/websource/tf/www.techfak.uni-erlangen.de
ScriptAlias /cgi-bin/ /proj/webbin/www.techfak.uni-erlangen.de/
</VirtualHost>
...
```

Excurse Apache 6

- Source-Security (Linux/Unix)
  - DocumentRoot- and CGI-Directory could be protected against other users:
    - Access to /proj/websource allowed only for users which belong to a special user group („webadm“), which is set by „NIS maps“ (net.groupID)
    - Access to DocumentRoot-directory additionally protected with ACLs:

```
> getfacl /proj/websource/tf/www.techfak.uni-
erlangen.de
          # owner: someone
          # group: somegroup
          user::rwx
          user:www:r-x                #effective:r-x
          group::r-x                  #effective:r-x
          mask:rwx
          other:---
```

Robots and Spiders 1

- Robots and Spiders
  - Robots and Spiders are used to search and index webpages, following a set of simple rules:
    1. (Get an URL out of a TODO-List)
    2. Load a document by an URL
    3. Analyse document: Links, Keywords, Content negotiation
    4. Add each link to TODO-List
    5. Save relevant data for the current URL
    6. Repeat from step 1 until end

Robots and Spiders 2

- Simple example in Perl:

```
#!/local/bin/perl

use LWP::UserAgent; use HTML::LinkExtor; use URI::URL;
$ua  = LWP::UserAgent->new;
$p   = HTML::LinkExtor->new(\&callback);
$url      = $ARGV[0];
if (not $url) { die("Usage: $0 [URL]\n"); }
push(@linkliste,"user\t$url");


while(($thiscount < 10) && (@linkliste)) {
    $thiscount++;
    ($origin,$url) = split(/\t/,pop(@linkliste),2);
    @links = ();
    $res = $ua->request(HTTP::Request->new(GET => $url),
                                    sub {$p->parse($_[0])});
    my $base = $res->base;
    @links = map { $_ = url($_, $base)->abs; } @links;
    $title = $res->title;
    for ($i=0; $i<=$#links; $i++) { push(@linkliste,"$title\t$links[$i]");}
}
print join("\n", @linkliste), "\n"; exit;

sub callback { my($tag, %attr) = @_; return if $tag ne 'a'; push(@links, values %attr);
}
```

Robots and Spiders 3

- Robots and Spiders (cont.)
  - Spiders can work parallel (by forking) or serial
  - Spiders never leave their machine: All „crawled“ pages are downloaded; Therefore the spider is also limited by the bandwidth of its machine (See also Lesson „Capacity Planing“)
  - Each entry within the database will time out after a period of time
  - (Friendly) Spider are following a set of rules, the „Robots Exclusion Protocol“, which works through a standardized file „robots.txt“, that should be located on a webserver which‘ pages are being spidered

Robots and Spiders 4

- Robots and Spiders (cont.)
  - Robot-Rules
    - http://www.robotstxt.org/wc/robots.html
    - Example „robots.txt“-file

```
User-agent: *
Disallow: /pictures/
Disallow: /intern/
```

    - Robots META-Tag within a HTML-file

```
<META NAME=„ROBOTS“ CONTENT=„NOINDEX, NOFOLLOW“>
```

Robots and Spiders 5

- Robots and Spiders (cont.)
  - Problems:
    - Due to limited bandwidth and space, it‘s not possible to index all webpages
    - Spiders cannot parse and index all internet files; They mostly fail at pages generated by client-side plugins (like Flash)
    - Spiders can only follow pages that are referenced! Without manual submit of the URL or use of referer-infos of UserAgents a spider would never visit a page no link is guiding to
    - Typical spiders index only up to 50 pages per domain
    - => Amount of existing internet files is much bigger as a search engine‘s database

Search engines in general 1

- Overview:
  - Local search engines
  - Catalogues
  - Web search engines

Search engines in general 2

- Local search engines
  - Real-Time search engines:
    - CGI-script, which opens a list of files and greps it for the searched word:
    - Filelist contains out of all files of a special type (mostly HTML) in a predefined start-directory
    - Subdirectories of the start-directory may be included optionally
    - Duration of search dependent of amount of webfiles, their size and the programming language;

Search engines in general 3

- Local search engines (cont.)
  - Index search engines
    - Avoids time-consuming real-time search though many files
    - Search only in a prepared index file
    - Index file is generated on regular time intervals
    - Two types of index files:
      - Summarization of all searchable files: Contains as entries the simple addition of all files without any chance and the reference to the orginal file
      - Parsed index file: Contains as entries only special Meta-Tag informations, like title, description and keywords of every file and the reference to the original file
    - Index often as textfile.

Search engines in general 4

- Local search engines (cont.)
  - Client-side index search engine
    - Search engine consists out of a client-side script that contains prepared datafields
    - The script will perform the search within these fields and return prepared result on success
    - Mostly implemented with JavaScript
    - Example datafield within script:

```
Portal|info,eingang,start,main|My Startpage|http://www.somewhere.com
Contact|contact,email,adress, impressum|Contact Page|http://www.....
...
```

Search engines in general 5

- Catalogues
  - As Websites
    - Examples: Yahoo, Web.de, DMOZ Open Directory Project, Portals, ...
    - Entries are made manually or by submit-tools within predefined categories
    - Often entries are checked by humans before their are commited into the index database
    - Indexes without human check get out of control after some time. Entries may get into wrong categories.
    - Management of categories gets complex on big indexes

Search engines in general 6

- Internet search engines
  - Original searchable files are located on other servers.
  - Real-Time search engines
    - Like local search engine, but instead of local file-open, access using HTTP-protocol
    - Very slow
    - Only used for special tasks like website-watchdogs (tools, that inform users about changes on a predefined URL)
  - Index search engines
    - All big comercial search engines: AltaVista, Google, HotBot, ...
    - Index is part of a high scalable database (Altavista: ~500.000.000 entries)

Search engines in general 7

- Internet search engines (cont.)
  - Index search engines
    - Database is filled up by „spiders“
    - If a page contains no link, it will continue at the last unknow link or quit if it was started as parallel process
    - A spider runs over pages until it followed all unknown links (very unlikely!) or it reaches a predefined limit

Search engines in general 9

- Perspective – new concepts:
  - Automatically combinations of Catalogues and Index search engines with help of artifical intelligence
  - Distributed search engines
    - Distributed spiders
    - Distributed index search engines with limited indexes, that point to each other
  - Personalized search engines
    - Requests to search engines are enhanced by additional individual informations for the requester to reduce results
    - Results of search engines are parsed through a set of individual rules

Google 1

- Topics:
  - History
  - Presentation of results
    - Link-Popularity
    - PageRank
    - Anchor Text Index
  - Architecture
    - Structure
    - Computing Centers & Google Dance
  - Commercial aspects

Google 2

- History
  - 1997 first publication by Brin and Page
  - Implemented as „Proof of Concept“ in 1998 Stanford University for several methods of sorting and searching results. Esp.:
    - „PageRank“
    - „Anchor Text Indexing”
    - Index plus Cache
    - Concepts for handling big indices
  - Months later Google became popular to the public

Google 3

- History
  - Google‘s enhancement against commercial search engines in 1998:
    - Not based on commercial interests
    - First scientific try to use concepts of Information Retrieval (IR) for hypertexts
    - Scaleable concept
    - No “Ranking for Cash”

Google 4

- Presentation of Results
  - Background
    To get a higher position in results, people tried to use several methods:
    - Misuse of keywords
    - Misuse of <META>-tags
    - Misuse of colors (white colored texts with keywords onto white background)
    - „Doorway-pages“

Google 5

- Presentation of Results
  - Link-Popularity
    - Made as answer against misuse to get a higher position
    - Idea: An URL is more important as another one if more links are pointing to it:
    - Number of links pointing to an URL is used.
    - Pro: Automatic created (commercial) URLs with no connect to the web get a low rank
    - But: Misuse was not stopped: Commercials now automatically created „Satellit websites“ pointing to an URL

Google 6

- PageRank
  („Page“ as reference to its creator L. Page).
  Uses several rules to calculate the rank of an URL:
  - The more links are pointing to an URL, the more important it is
  - The less links are within a document, the more important is the URL the link is pointing to
  - The more important a document is, the more important are the links
  - The more important a link is, the more important is the URL the link is pointing to

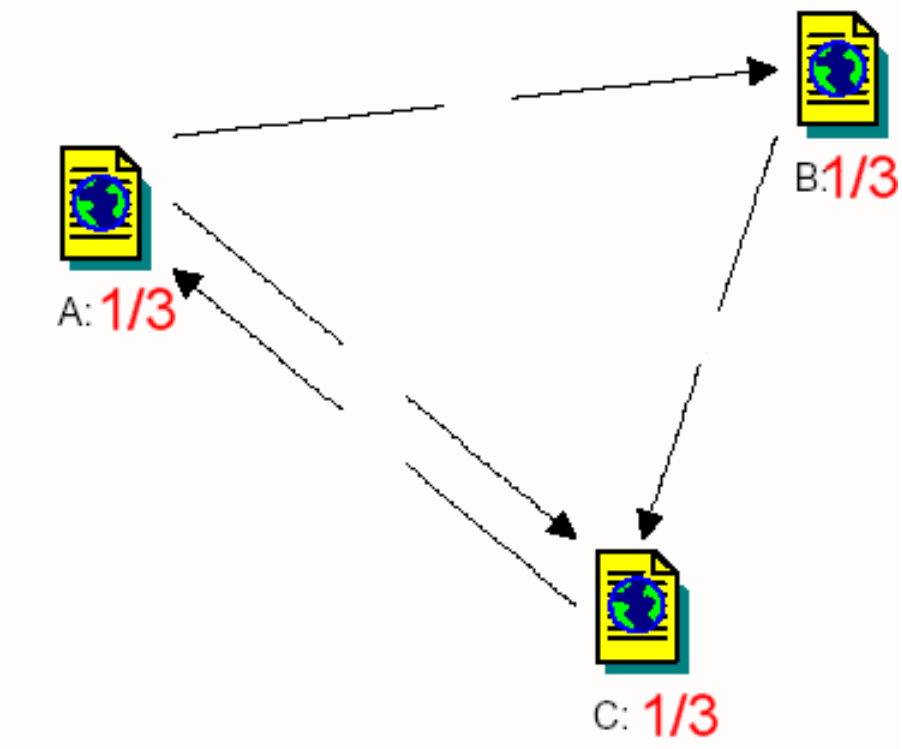
Google 7

- PageRank
  - Iterative algorithmic:

1. Every node (URL) gets initialized with a start value. Mostly used: probability pattern = weight of node = 1 / (Number of all Nodes)
2. Link-weights of nodes are calculated as weight of node / number of links
3. Out of ingoing links the node-weight gets recalculated as $\sum$ link-weights
4. Repeat at point 2.) until node-weights are convers or the proximity fits the limits

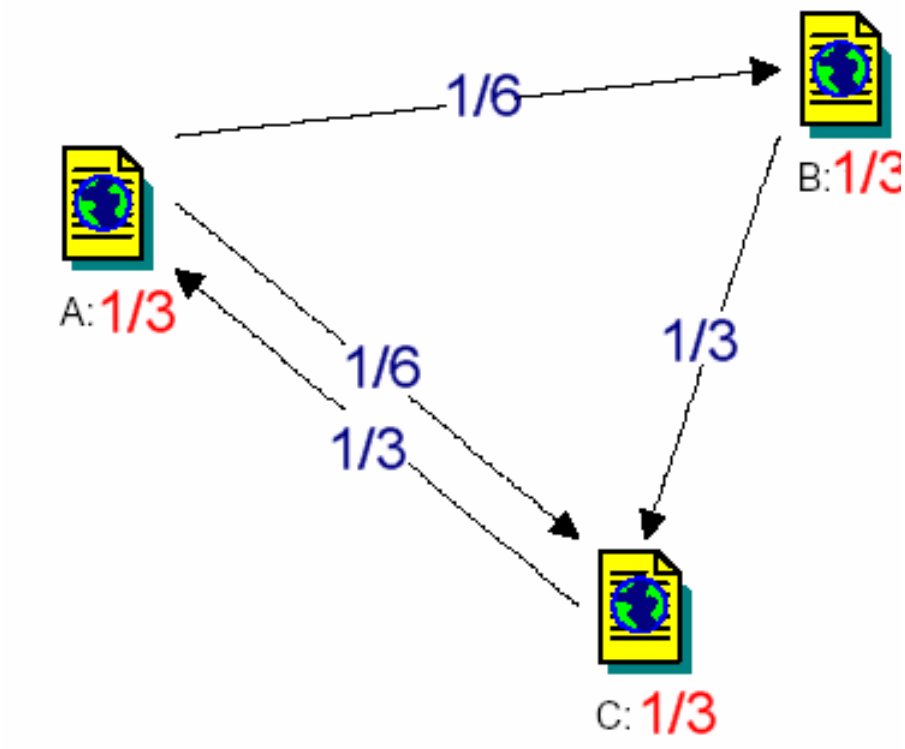
Google 8

- PageRank (step 1)



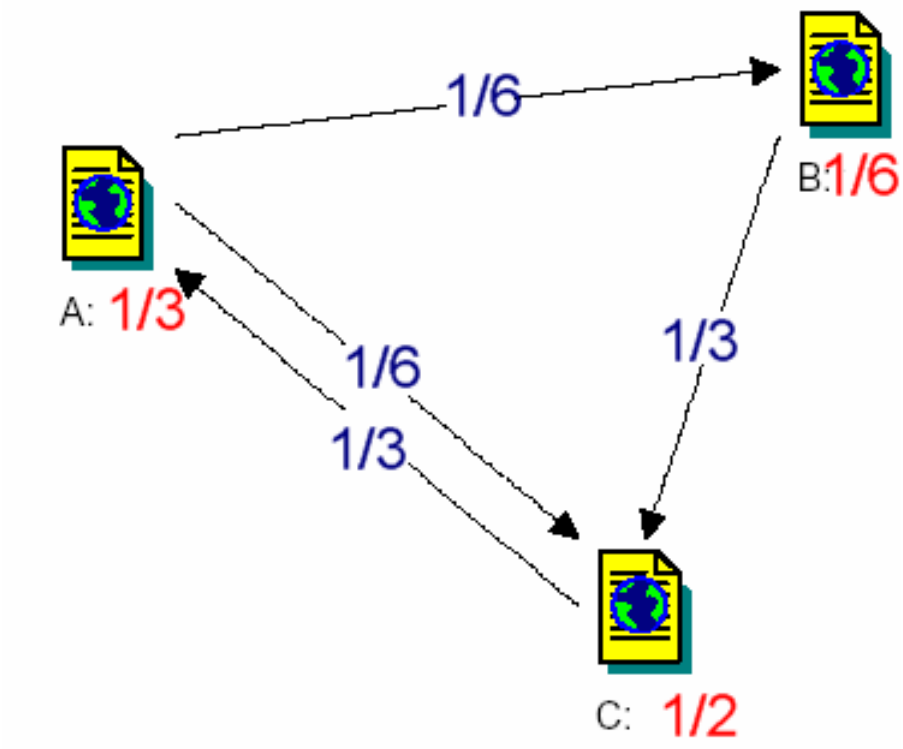
Google 9

- PageRank (step 2)



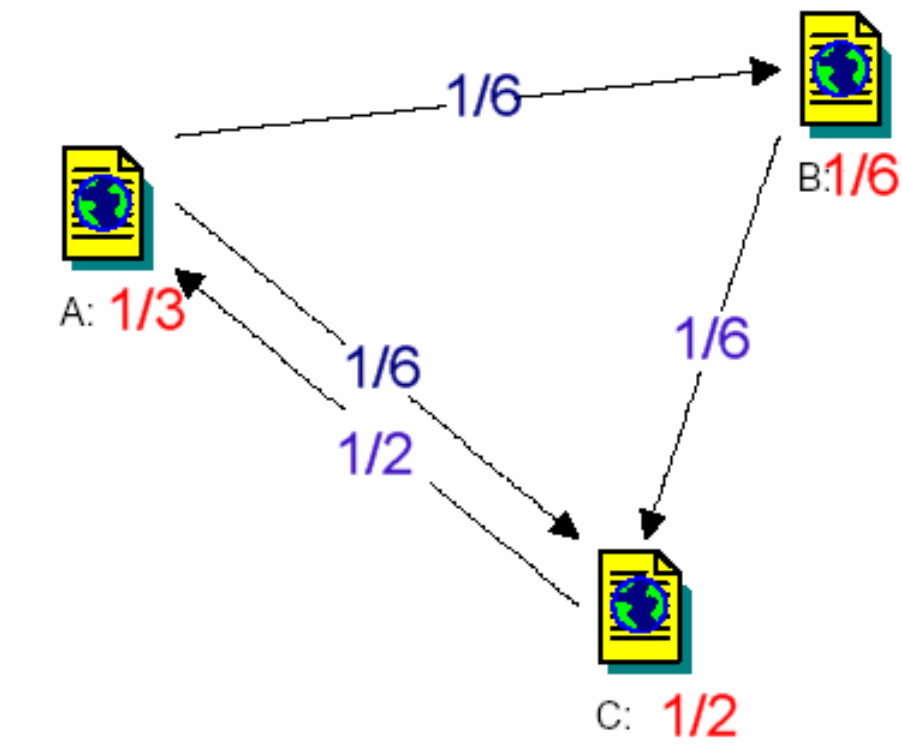
Google 10

- PageRank (step 3)



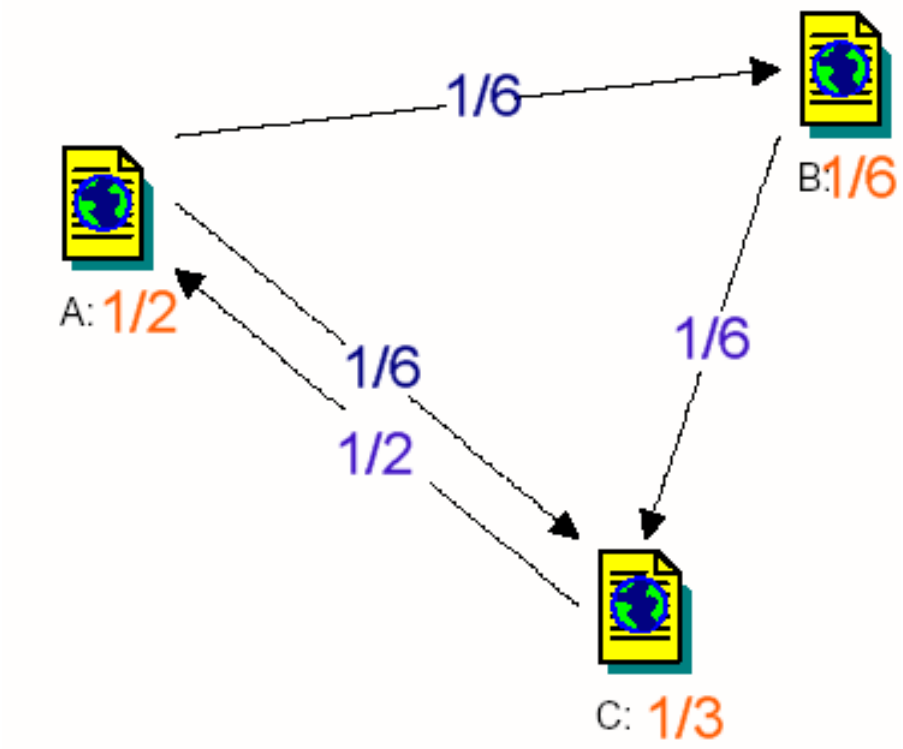
Google 11

- PageRank (repeat step 2)



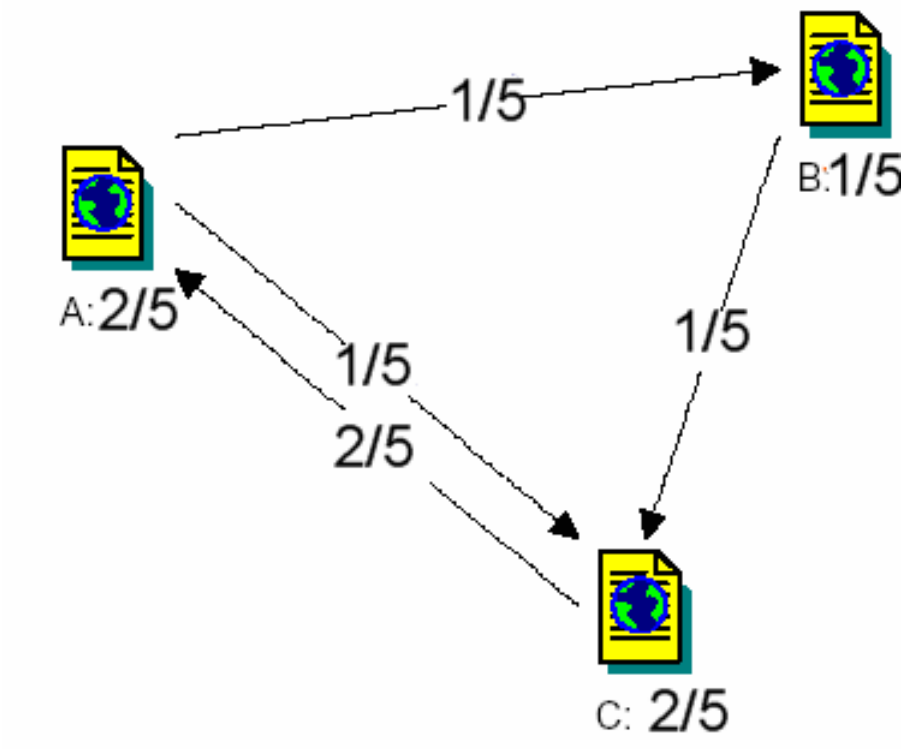
Google 12

- PageRank (repeat step 3)



Google 13

- PageRank (after proximity fits)



Google 14

- PageRank
  - Formular representation for simple form:

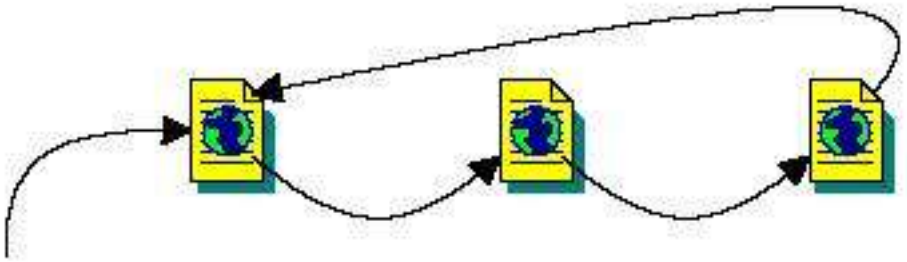
$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v}$$

With: u = URL, F_u = Sum of URLs, which are linked by URL u, B_u = Sum of URLs, which are linking to URL u. $N_u = |F_u|$ sum of all links in u. Factor c < 1 is used for pages without links

  - But: “Rank Sinks” are possible

Google 15

- PageRank
  - „Rank Sinks“ occur on linked pages which are linking in circles (which is common for documentations)



  - Formular is extended with a factor, called „Rank Sources“ E(u):

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u)$$

Google 16

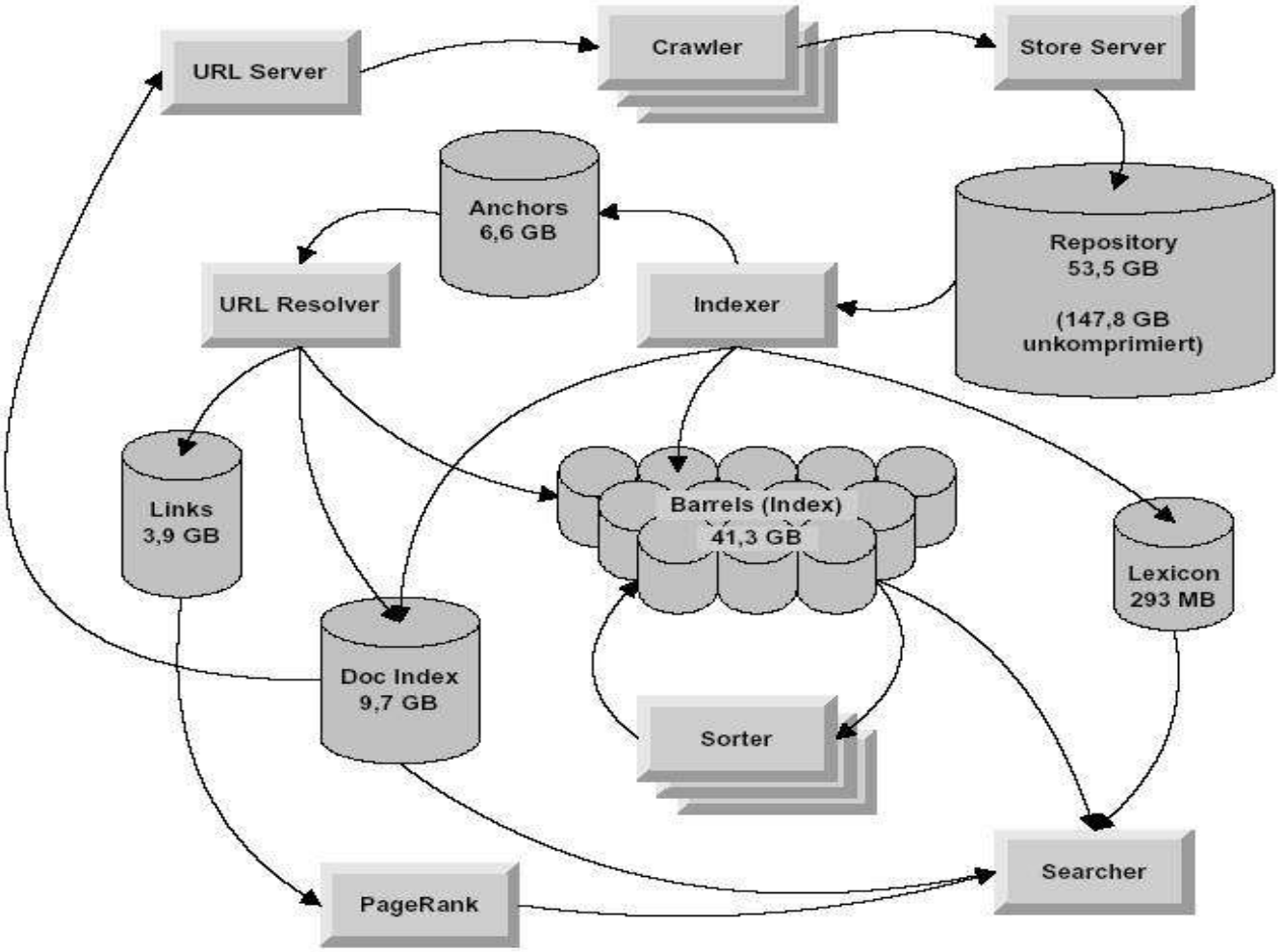
- PageRank
  - Cause not all URLs are part of the index, „Dangling Links“ are defined: „Dangling Links“ are all links in an URL which are pointing to a document outside the index.
  - Dangling Links will be ignored at the calculation of PageRank.
  - Calculation of PageRank is not depending of searching patterns. Therfor, the calculation is made „offline“.
  - Algorithms is fast: 25 Mio documents with 75 Mio links are calculated in one iteration in 6 Minutes, using a standard workstation (2001 !)

Google 17

- Anchor Text Indexing
  - Link in HTML:
    About this link
  - Referencing text is analysed for keywords; this keywords will be added to the URLs keyword index
  - Needed esp. for pages with graphical content and few words.

Google 18

- Architecture (2001)



Google 19

- Computing Centers & Google Dance
  - Google's index is created by about 10000 workstations (Linux) which are divided onto (currently) 8 computer-labs wordwide.
  - Index gets renewed and recalculated one time a month.
  - Due to aspects of data security and time costs, the index of all computer labs are updated after each other: This is called as "Google Dance".
  - In this time, search requests for the same words can answer with different results, cause different computer labs may answer.

Google 20

- Computing Centers & Google Dance
  - The answering computer-lab is chosen by the DNS
  - Different IP-adresses are used for the same servername.
  - TimeToLive (TTL) for „google.com“ is set to 5 minutes only.
    After this time has passed a local name server has to request an update by Google‘s name server. The answer may differ depending on the local name servers location and other aspects.

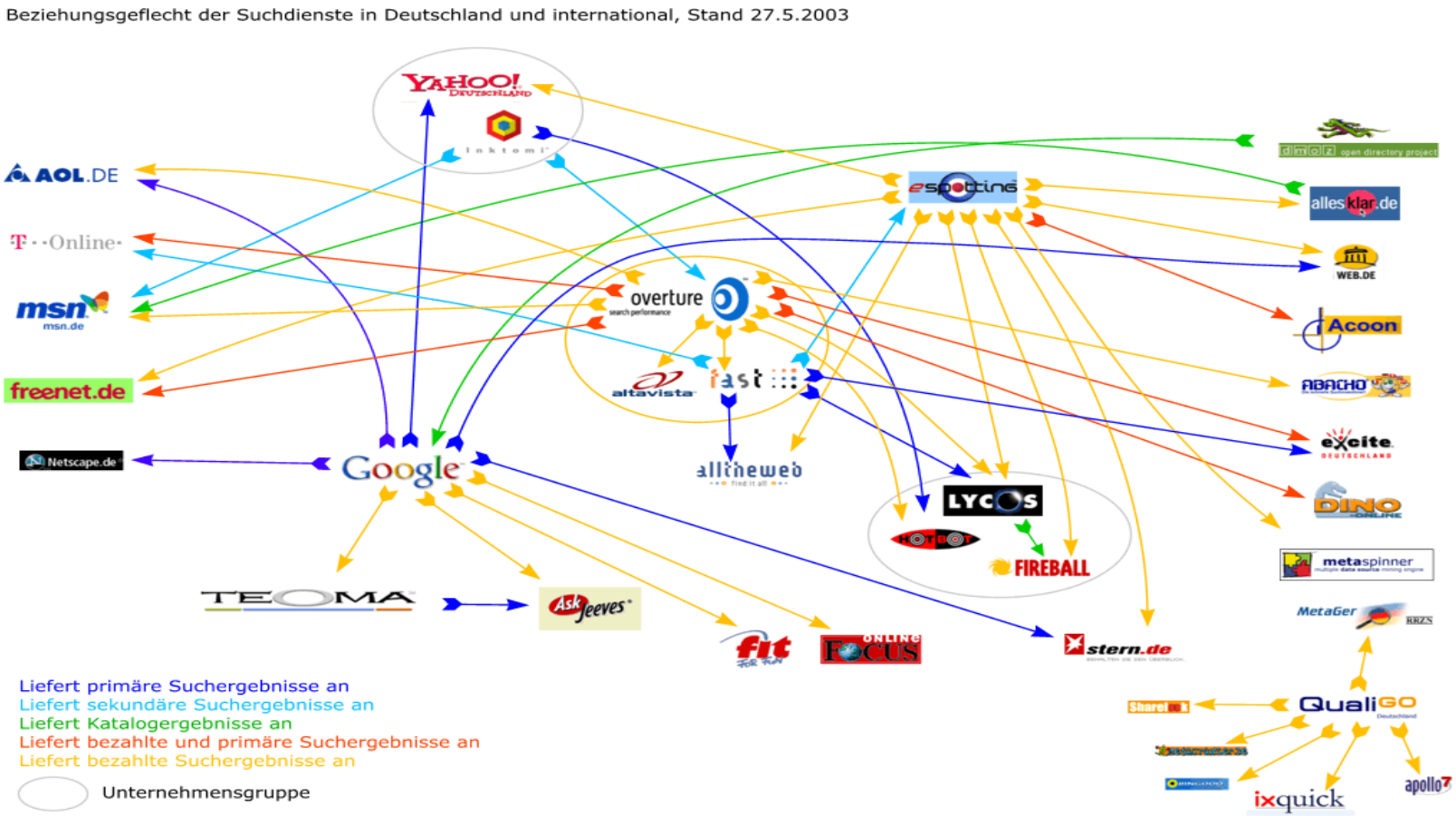
Google 21

- Commercial aspects
  - Google makes cash by
    - Selling licenses for its technique (for use of local search engines)
    - Placing text ads beneath search results
    - Content syndication (!)

Google 22

- Commercial aspects

http://www.suchfibel.de/



Die Suchfibel