E-Commerce: Searching in Document Collections

Claudio Silvestri

Searching

- A fundamental function of e-commerce web sites is:
  - SEARCH !!!!!!!
- Other example of searching ?
  - Web Search Engines

What's the WEB

- Hyper-textual documents:
  - Text
  - Links
- Web:
  - Billions of documents
  - Written by millions of users (expert and not)
  - Distributed over thousands interconnected machines
- But also:
  - Images, videos, and many other …

Some historical notes

- **MEMEX**, 1945 [Vannevar Bush]
  - "MEMory EXtension"
  - The goal is to generate hyperlinks between documents in order to help the user during the browsing
  - "Photoelectrical-mechanical storage and computing device" able to store a large amount of information, to edit documents, and to create so called "trails", i.e. a linear sequence of links connecting frame of microfilms
  - Bush wanted to mimic the way our minds handles information

  - **It was never built**

Hypertext

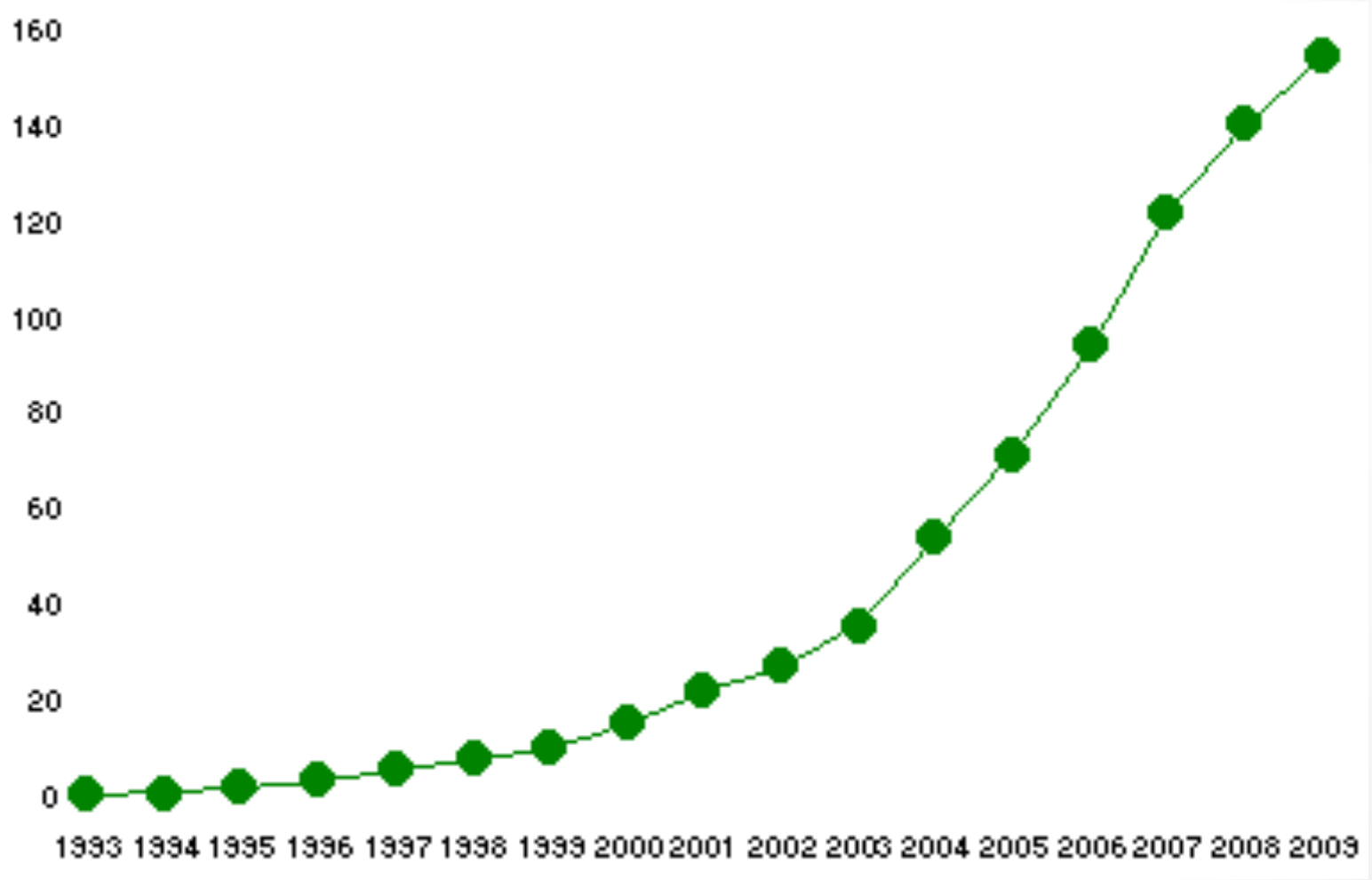
- Hypertext, coined by Ted Nelson in 1965:
  - Hypertext refers to non-sequential reading/writing, a text where the user as multiple choice points, deciding his own reading path

- He founded the **Xanadù** project:
  - Support non sequential reading/writing
  - The universal database of documents (docuverse) allows (bidirectional) links between any two given words or sentences of two different documents
  - But also copyrights and other stuff
- His opinion about Web
  - “Today's popular software simulates paper. The World Wide Web (another imitation of paper) trivializes our original hypertext model with one-way ever-breaking links and no management of version or contents”

The World Wide Web

- Tim Berners-Lee, CERN 1989
  - To allow sharing of information between labs in different countries
  - Born in the community of High Energy physics
  - Today it is the most advanced information system
- What does W3 define?
  - Uniform Resource Identifier (URI)
  - Hypertext Transfer Protocol (HTTP)
  - Hypertext Markup Language (HTML)
- The first Web page:
  - http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html

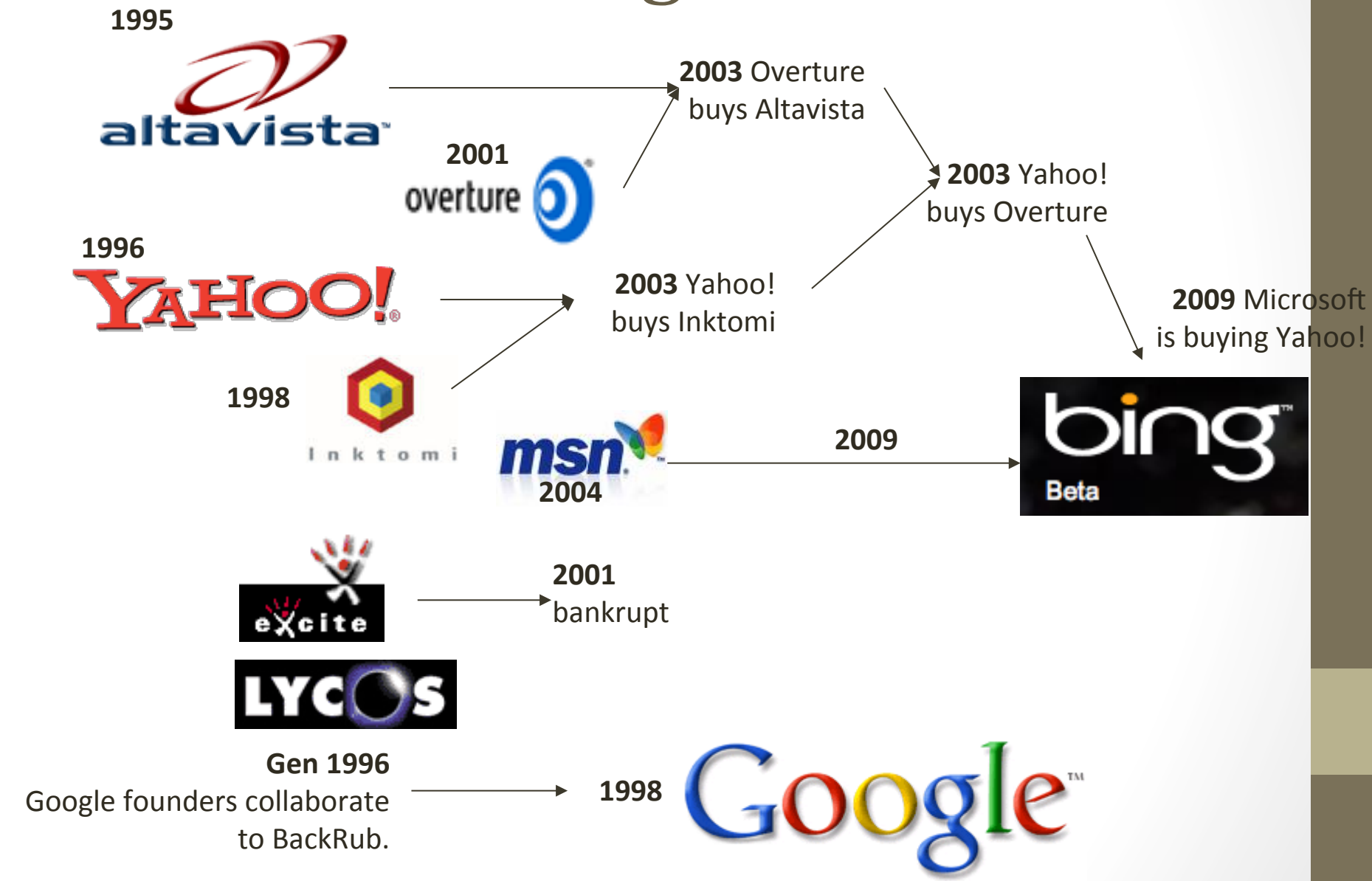
Number of Internet Hosts in Europe



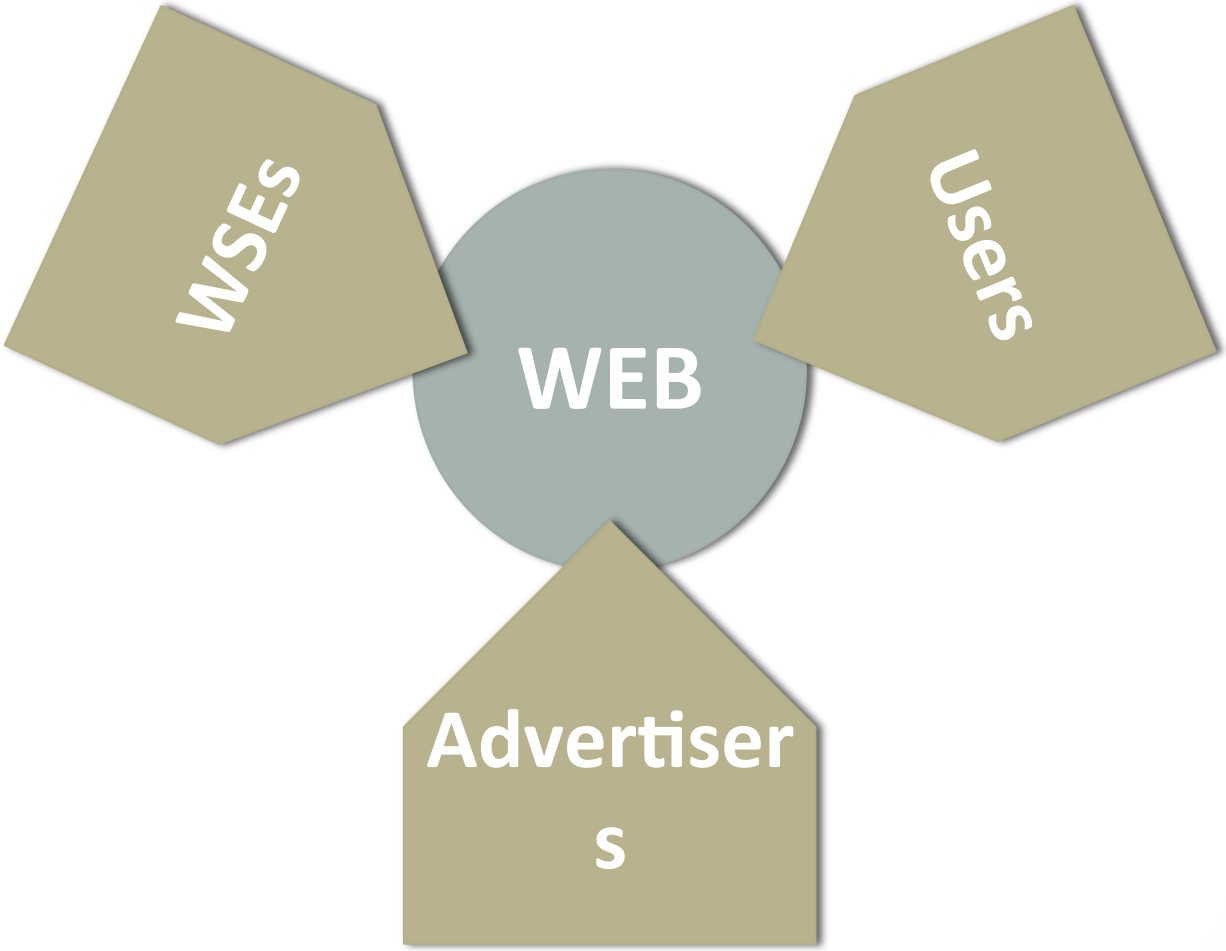
After a little while

- Overwhelming amount of information:
  - Billions of pages
  - Millions of users
  - Never-ending information flow
- And …
  - Duplicates / mirrors / redundancy / spam
  - Authoritativeness ??
- How to find the information you need ??

Web Search Engines War



WSEs / Users / Advertisers

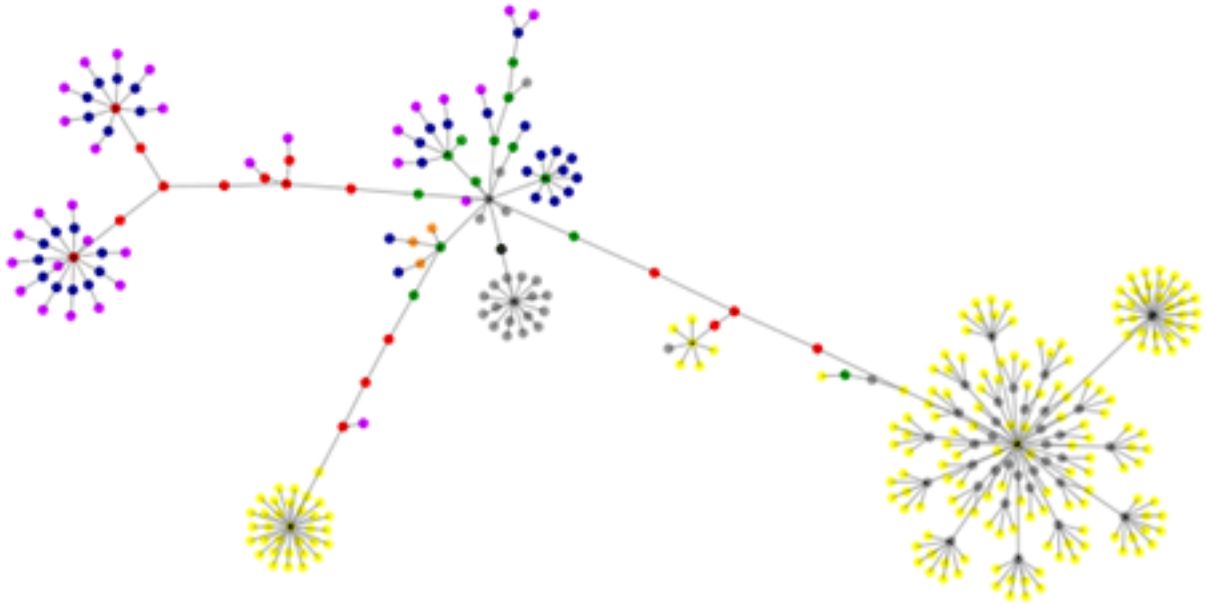


The big players

Core Search Entity	Share of Searches (%)	
	Sep-09	**Oct-09**
Google Sites	64.9%	65.4%
Yahoo! Sites	18.8%	18.0%
Microsoft Sites	9.4%	9.9%
Ask Network	3.9%	3.9%
AOL LLC Network	3.0%	2.9%

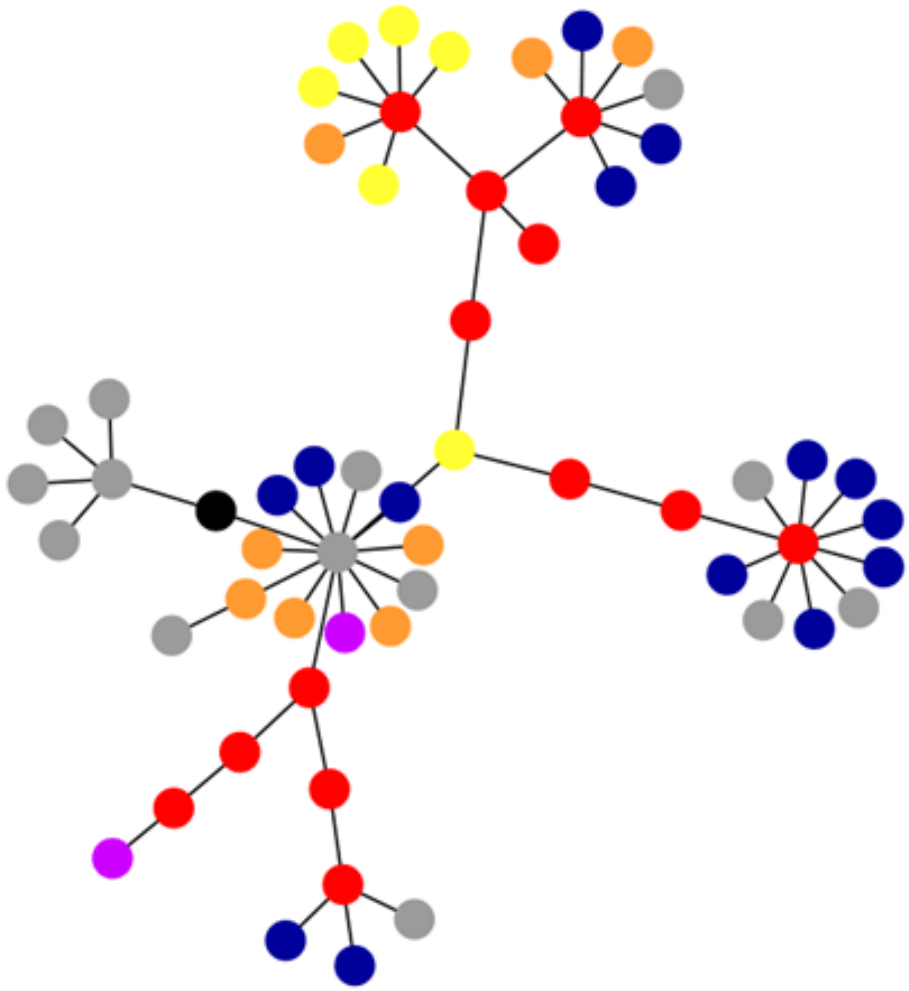
Pieces of the Web

- http://www.aharef.info/2006/05/websites_as_graphs.htm
- http://www.aharef.info/



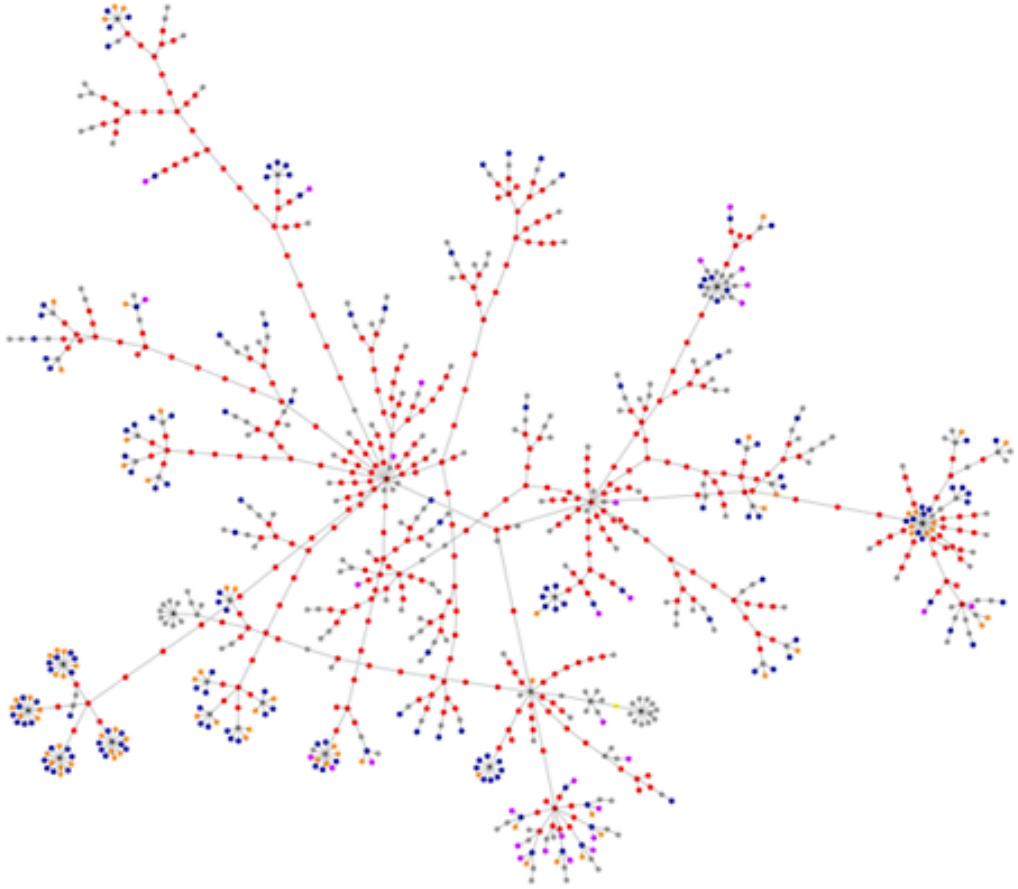
APPLE.COM

Pieces of the Web



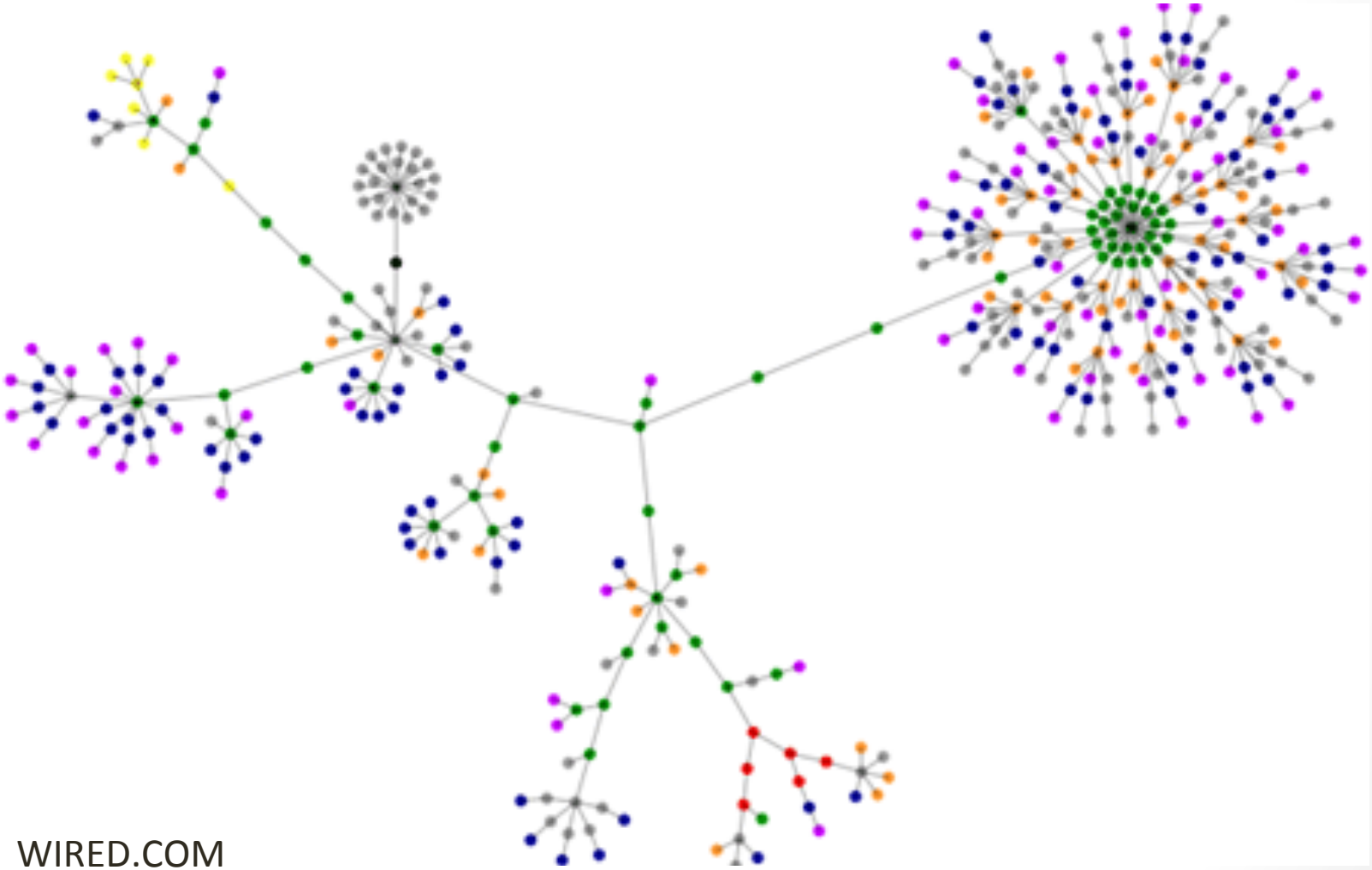
GOOGLE.COM

Pieces of the Web



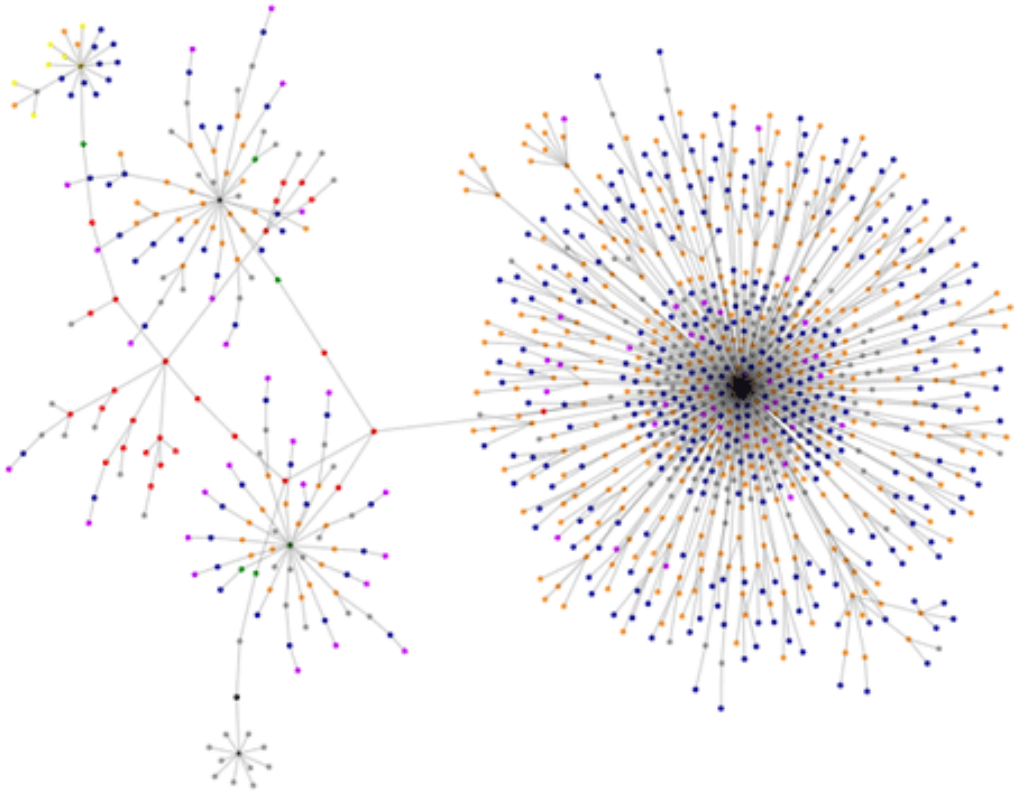
YAHOO.COM

Pieces of the Web



WIRED.COM

Pieces of the Web



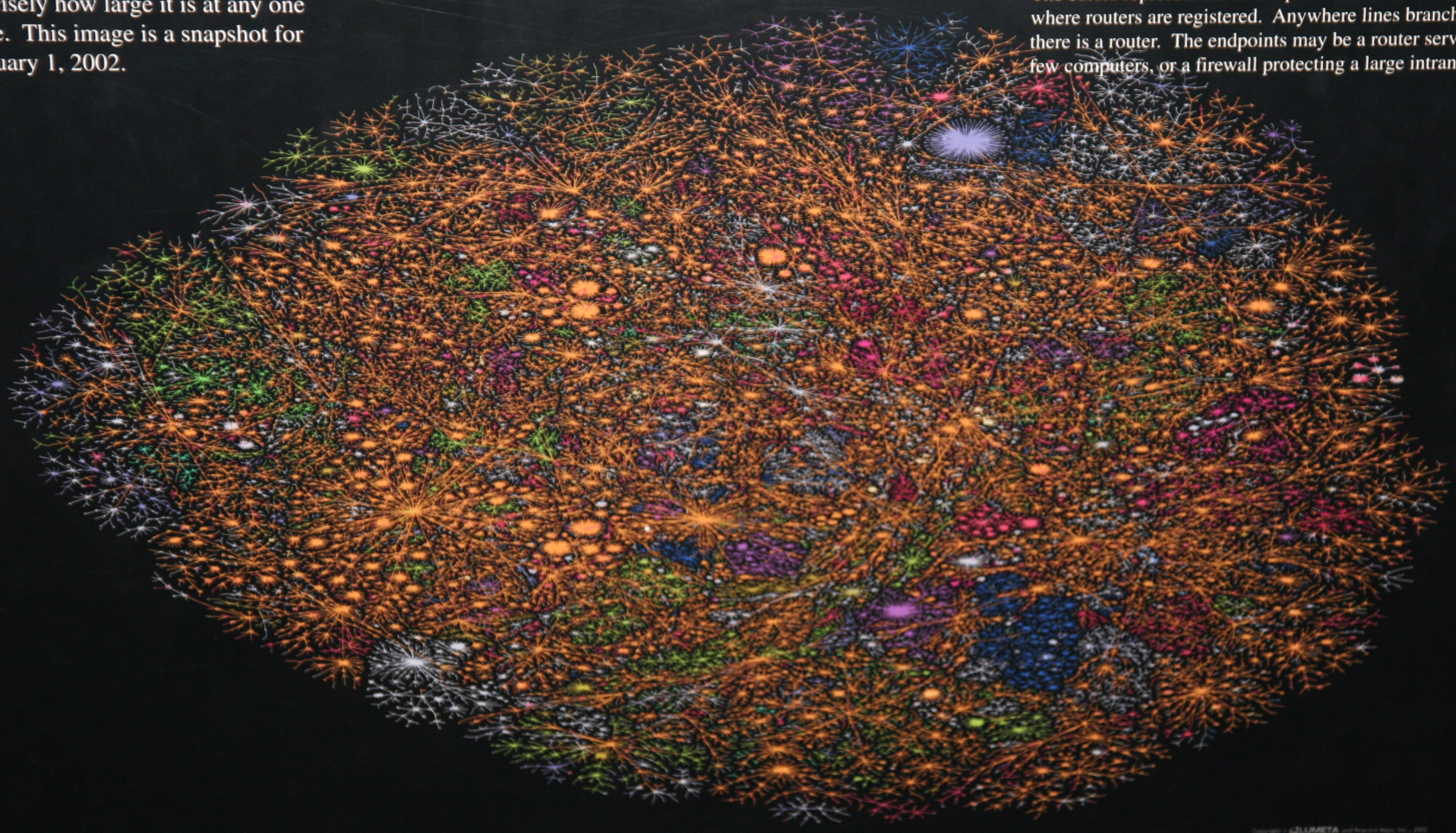
BOINGBOING.NET

The Internet in 2002

approx. 500 million hosts

The Internet has become so big and so fluid, that it is almost impossible to say precisely how large it is at any one time. This image is a snapshot for January 1, 2002.

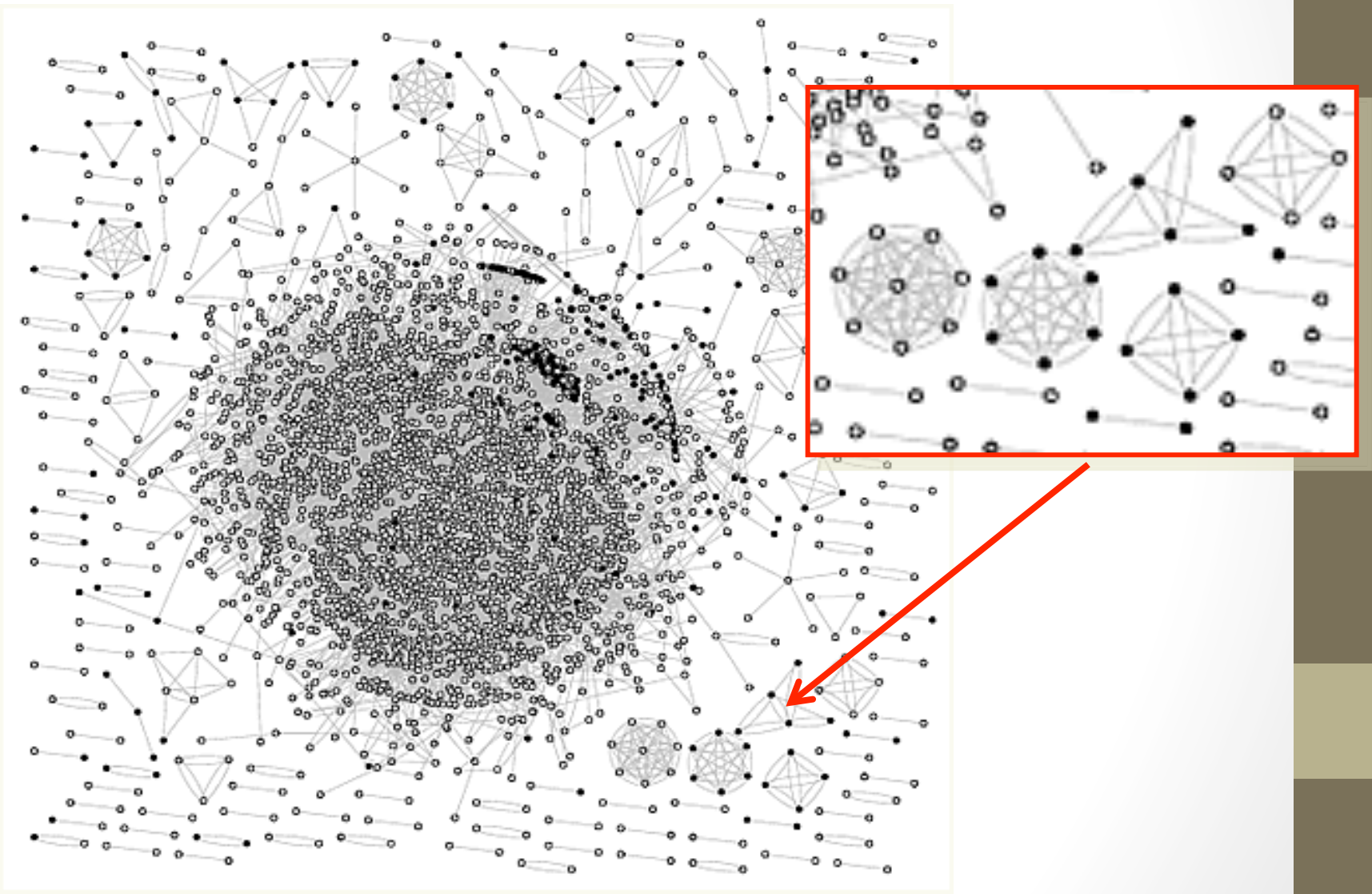
This graph was created by plotting the shortest path between a computer in New Jersey and 177,017 listed networks. The colors represent the 136 top-level Internet domains where routers are registered. Anywhere lines branch, there is a router. The endpoints may be a router serving a few computers, or a firewall protecting a large intranet.



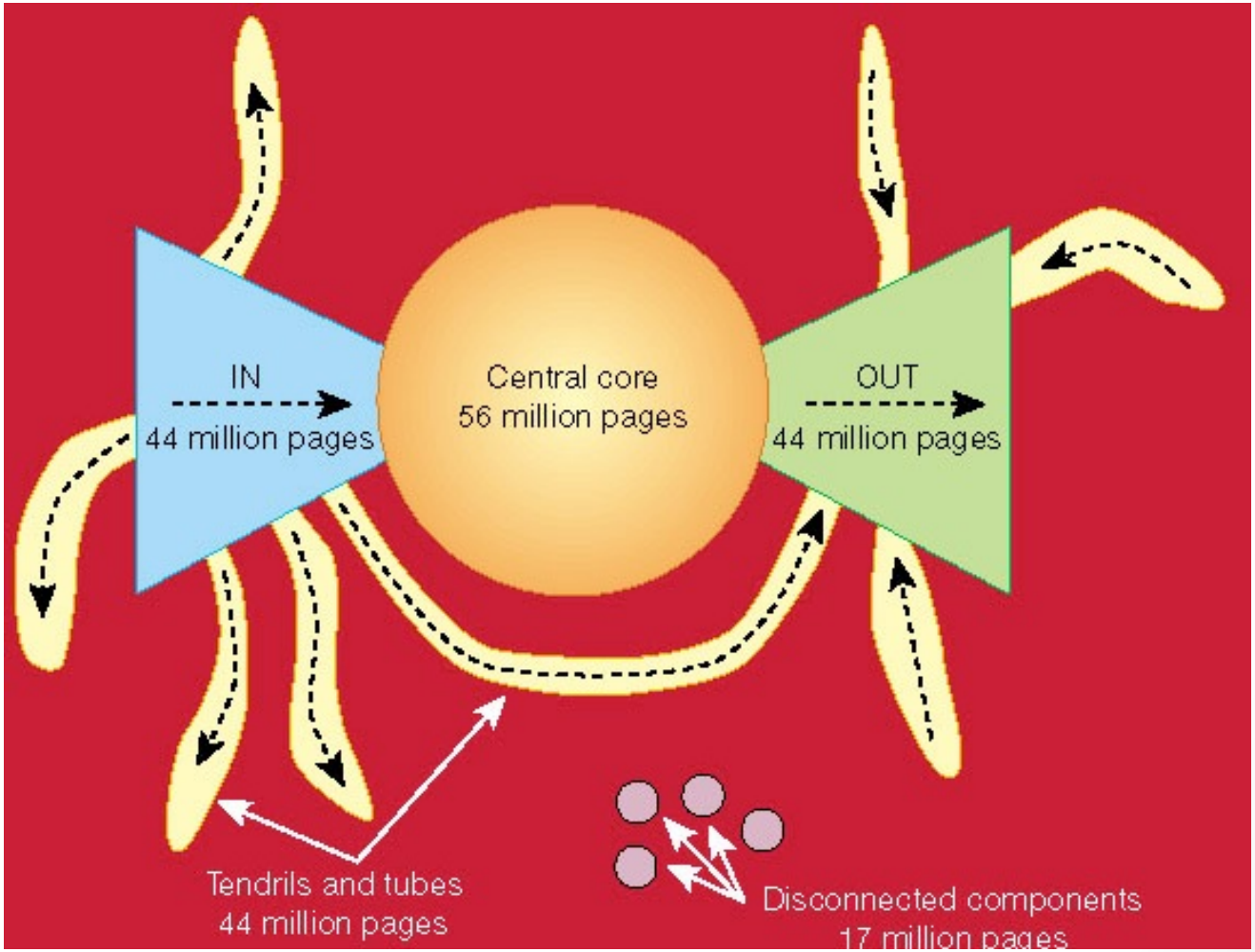
Graph by Hal Burch and Bill Cheswick. Router by Irene Sage and Gregory Shapiro

North America
Central America
South America
Africa
South Africa
Europe
Germany
France
Netherlands
United Kingdom
Asia
Japan
Pacific Islands
Australia
New Zealand
.mil .nato
.gov
.com
.edu
.org
.net
other

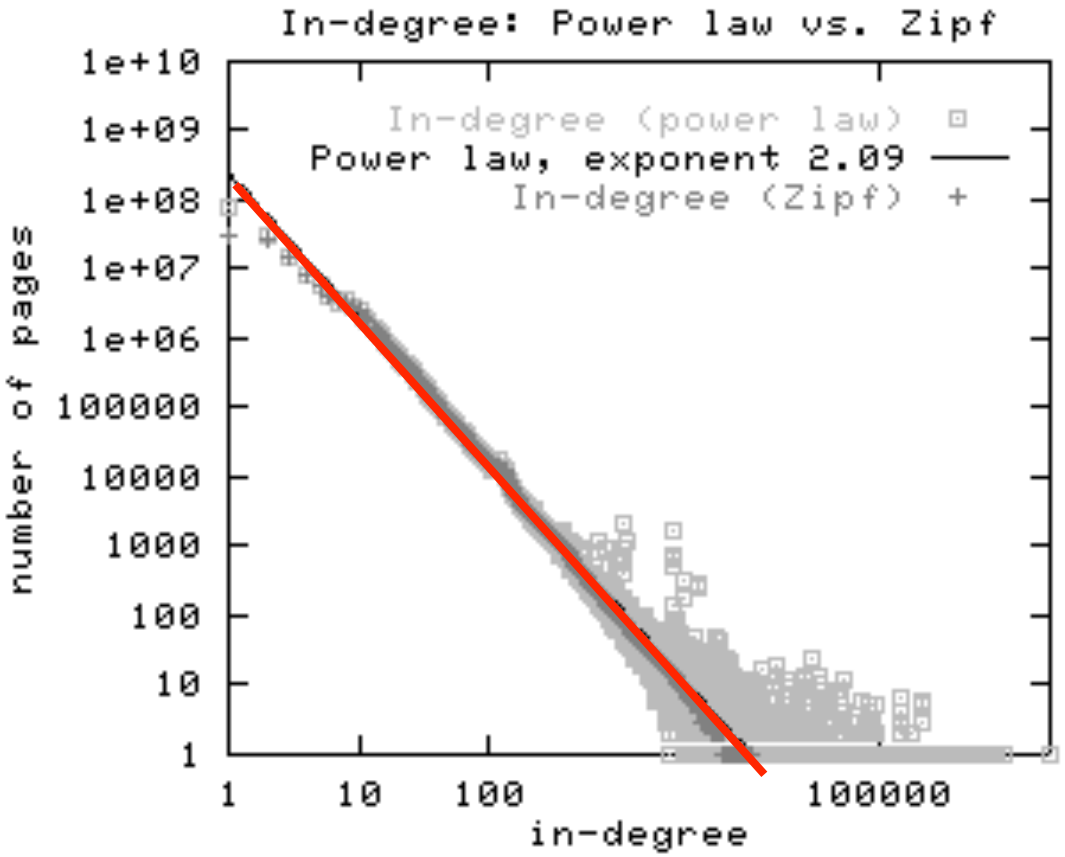
Pieces of the Web Graph



The web is a bow tie.
Nature 405, 113(11 May 2000)



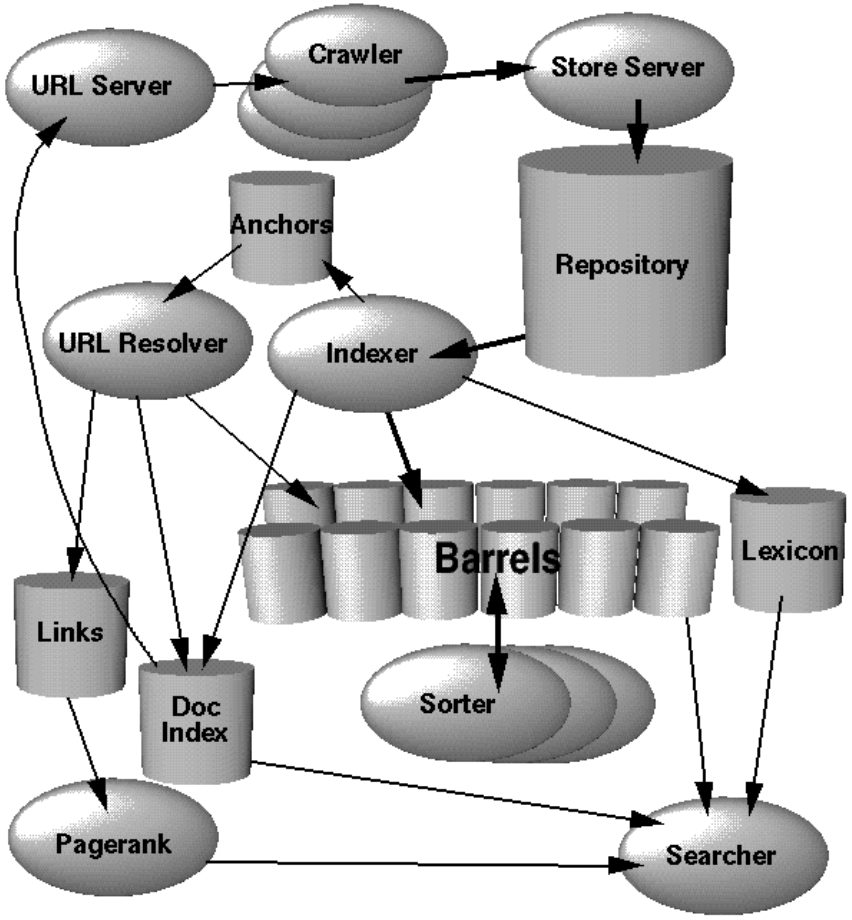
In-Degree: Power Law Distribution



Power Law

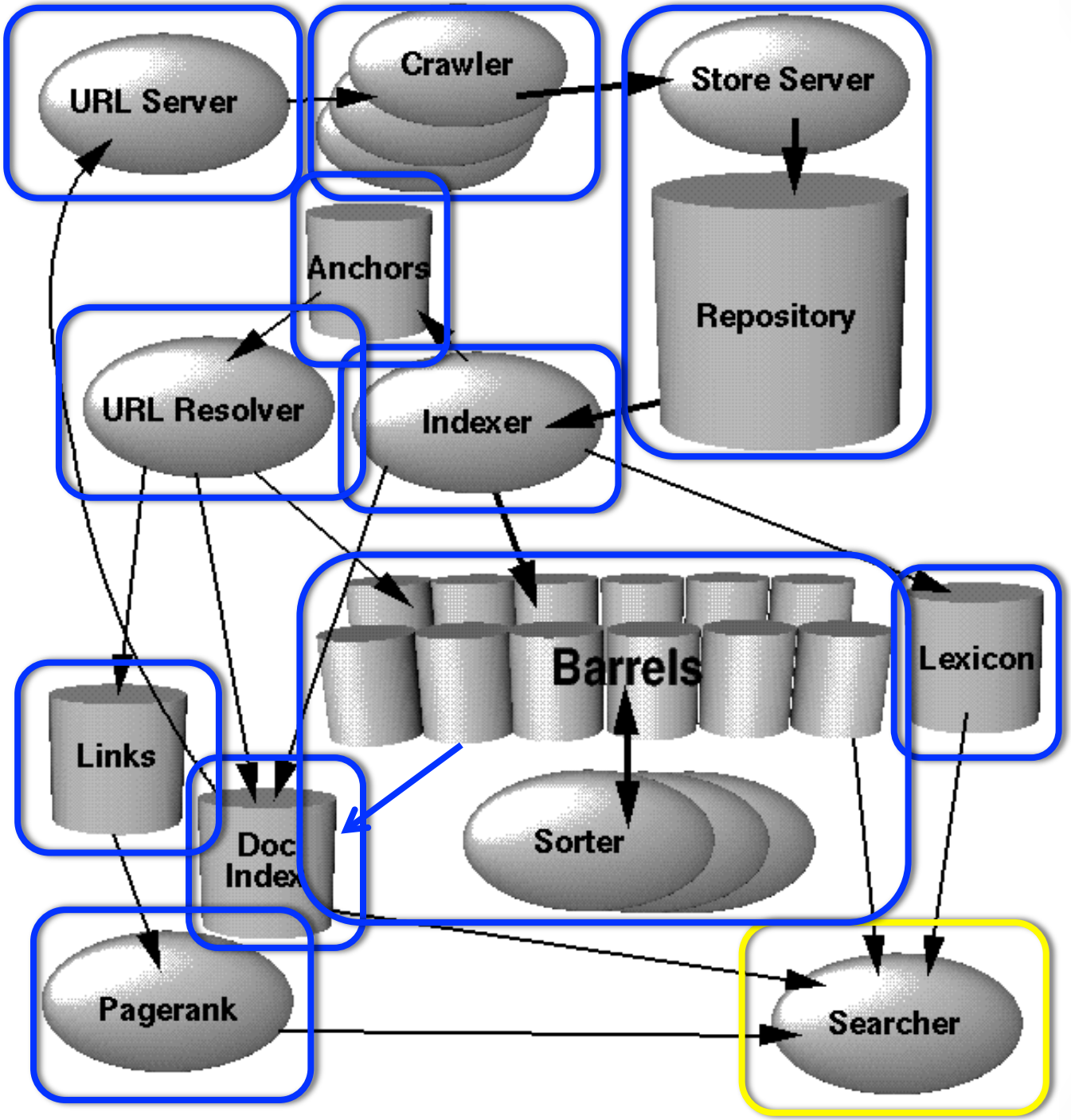
$$Y = 1 / x^{a}$$

(1998) The Anatomy of a Large-Scale Hyper-textual Web Search Engine



http://infolab.stanford.edu/~backrub/google.html

Sergey Brin and Lawrence Page (Stanford)


URL Server
Crawler
Store Server
Repository
Anchors
URL Resolver
Indexer
Barrels
Lexicon
Links
Doc
Index
Sorter
Pagerank
Searcher

Crawling the Web

"Running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. **Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers** and various name servers which are all beyond the control of the system."

Crawling the Web

… Each crawler keeps roughly 300 connections open at once… At peak speeds, the system can crawl over **100 web pages per second using four crawlers. This amounts to roughly 600K per second of data.**

… Each of the hundreds of connections can be in a number of different states: looking up DNS, connecting to host, sending request, and receiving response…

Crawling the Web

"Wow, you looked at a lot of pages from my web site. How did you like it?"

"This page is copyrighted and should not be indexed"...

Crawling the Web

"…our system tried to crawl an online game. This resulted in lots of garbage messages in the middle of their game!"

"It is virtually impossible to test a crawler without running it on large part of the Internet…. Systems which access large parts of the Internet need to be designed to be very robust and carefully tested.

"crawlers will invariably cause problems, there needs to be significant resources devoted to … solving these problems as they come up"

Something about today's crawlers

- Deep Web:
  - Everything beyond a form
  - Everything which is not linked
  - Non textual data
  - Dynamic pages

  - Modern crawler simulate browsers and run web pages !

- Structured information:
  - Train timetables
  - http://www.wolframalpha.com/

Indexing and Query Processing

- How to find all the documents answering the query “Divina Commedia” ??
- Naïve approach (*forward index*):
  - Scan the collection document by document
  - Measure the similarity between the query and the document
  - Maintain the list of the top-10 most similar documents
- Disadvantages:
  - The cost depends on the total number of documents.
    It takes the same time even if there is no document answering the query

Inverted Index

- Lexicon:
  - The set of terms occurring in the collection
- Postings Lists:
  - A list of document IDs containing a given term
- Inverted file:
  - Is an array of postings lists, one for each term in the lexicon

- Divina => [1, 12, 15, 18, 20, 25]
  Commedia => [7,12, 15, 25, 33]
  - Documents [12,15, 25] answer the query “Divina Commedia”

Size

- Forward index:
  - 1 Billion documents
  - Each document is about 4 KB
- Inverted file:
  - Lexicon
    - Heap's Law: $|\text{Lexicon}| \propto \text{sqrt}(|\text{Collection}|)$
  - For each term, we store one list of DocIDs
- The inverted file is going to be smaller (but not small enough)

Creating the Lexicon: Tokenization

- Transform an HTML web page into a set of meaningful ***tokens***, i.e. words of the lexicons
- Ignore punctuation: period, comma, parentheses, etc.
- Everything lowercase
- Remove HTML tags
- Extract additional data:
  - Tokens in URL
  - <meta … content="this page talks about …">
  - <title> this is the title </title>
- Remove Stop-Words:
  - "a", "an", "the", "to", …

Creating the Lexicon: Stemming

- “Computer” and “Computers” are the same word
- Rather than storing a postings list for each word, store a postings list for each **stem**
- Porter Stemmer
  - Iteratively remove prefixes and suffixes without using a reference dictionary
  - computer, computers, computing -> comput
  - organization, organ -> organ
  - police, policy -> polic
  - May not match cylinder with cylindrical

Other tough operations

- Am, have been, being -> BE
- Detect named entities, dates, currencies, etc.
  - Venice, Bank of England, …
  - March 11, 2010
  - Price is **666$**
- Synonyms and other semantic analysis
  - Car, auto, etc.
  - Notebook, Netbook, etc.
  - Venezia (is related to the Veneto region)

Intersection: Term-at-a-time

1. Given the query $q = \{t_0, t_1, \ldots, t_n\}$
2. The the postings list of t_0 and store it in *res*.
3. For i =1 to n

   Intersect the postings list of t_i with *res*,
   and store the result in *res*
4. Sort the documents in *res* (if any)
   according to some similarity measure (e.g. cosine)

Intersection: Document-at-a-time

1. Given the query $q = \{t_0, t_1, \ldots, t_n\}$
2. "Open" each postings list of the terms in the query
3. Scan the postings lists until a DocId is present in every of them

   Add such DocId to the result set *res*
4. Repeat 3 as long as no postings list is empty
5. Sort the documents in *res* (if any)
   according to some similarity measure (e.g. cosine)

Which one is best ?

- Term-at-a-time forces to read “every” Doc-ID in every postings list.
- Document-at-a-time allows to “skip” uninteresting portions of a postings more easily.
- There are external measures of interest that can provide early stopping criteria to the Document-at-a-time strategy.

AND vs. OR

- This was for AND queries,
  how would you perform an OR query ???

Boolean Model vs. Vector Space Model

- Boolean model consists in performing the intersection/union of postings list
  - No scoring is present
  - Nice expressivity for the user
- Vector space model takes the union and then compute the cosine similarity
  - Results are scored
  - The user cannot express OR vs. AND queries
    - Tough there are some hybrid models: Salton et al. introduced the Extended Boolean Model in 1983

Crawling @home

http://www.chilkatsoft.com/refdoc/pythonCkSpiderRef.html

http://www.example-code.com/python/pythonspider.asp

```
import chilkat
spider = chilkat.CkSpider()
spider.Initialize("www.chilkatsoft.com")
spider.AddUnspidered("http://
www.chilkatsoft.com/")
for i in range(0,10):
    success = spider.CrawlNext()
    if (success == True):
        print spider.lastUrl()
```