Harvest

- University of Colorado(1994+)
  http://harvest.cs.colorado.edu/
- Major System Components
  - Gatherer : locate + process information(webspider+)
    also Essence subsystem(summarizer)
  - Broker : Information server(interface to gathered data)
  - Indexing/Search subsystems : specialized search interface to broker
  - Object cache : supports rapid retrieval of frequently used object
  - Replicator : supports transparent mirror sites

Distributed database technology { Object cache, Replicator

Harvest Motivation

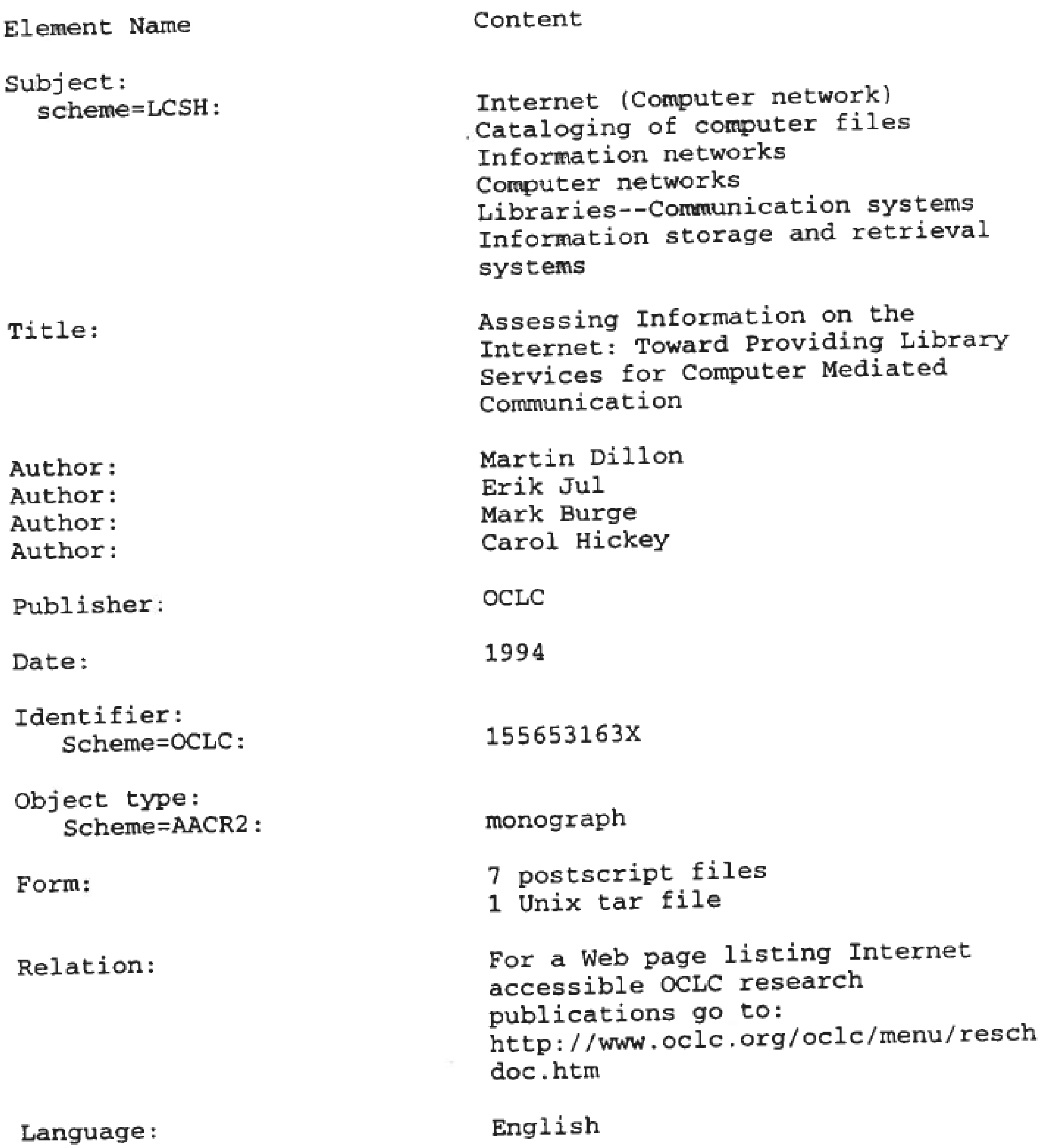
- Current web indexing systems exhibit much
  - Duplication of effort
  - Inadequate granularity and quality of indexing and meta-data storage/access
  - Inadequate cooperation with content providors

- Potential solution:
  - Semi-centralized information collectors (gatherers) with good cooperation with service providers (may even be run on local service providers' sites)
  - Secondary indexing programs, filters and agents, using broker interface to the Gatherer content sources

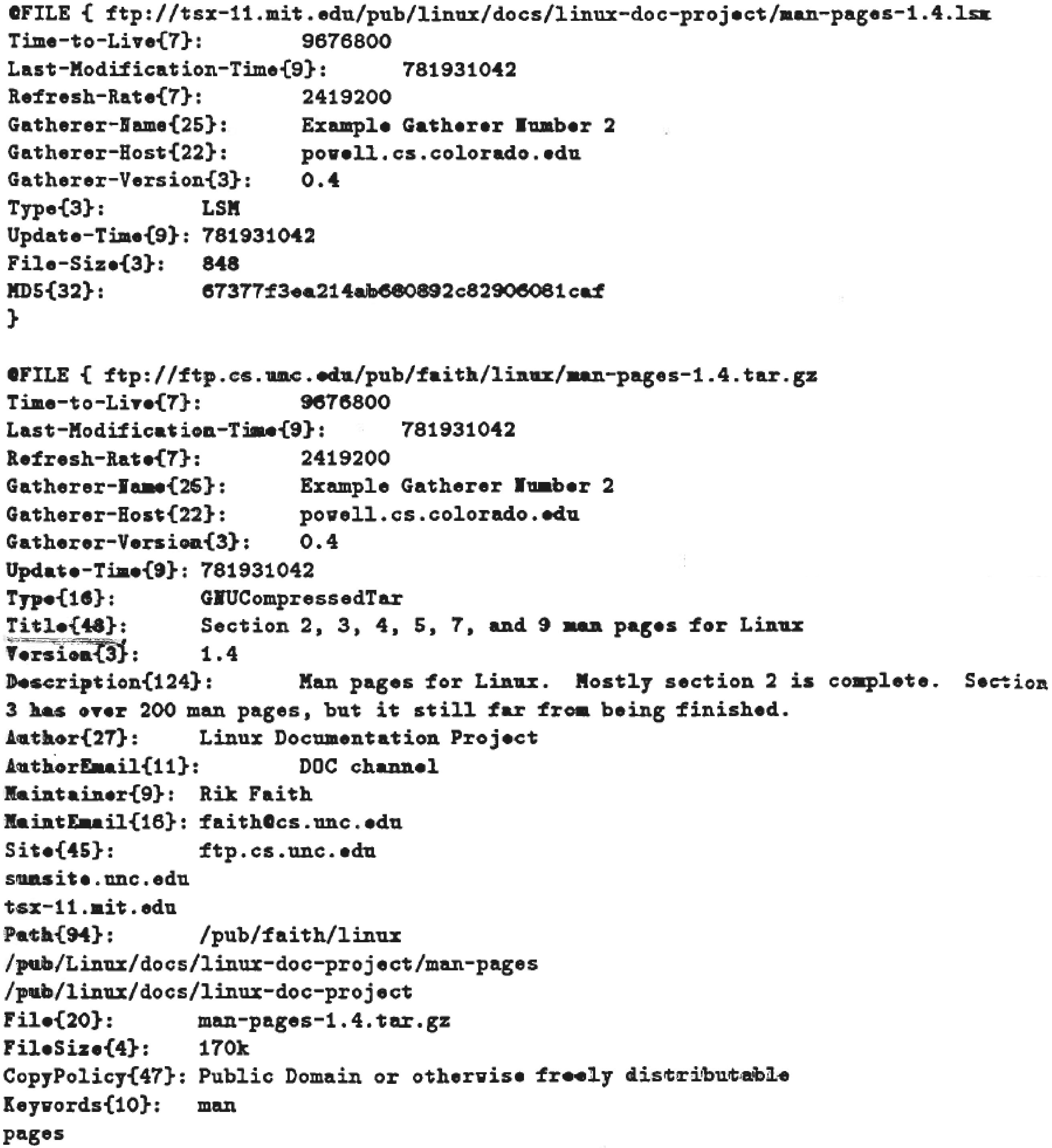
Gather

- Catalogs/finds web pages
- Summarize/extract info that search agents need
  - Essentially extracts full SOIF++ header for all documents for downstream indexing use
- Including:
  - TypeID (e.g. postscript, man page, perl library comments, tar of jpgs)
  - Title
  - Keywords
  - Charset/LanguageID (not trivial)
  - Author
  - Subject/Topic (topic classificiation)
  - References/Links
- Currently based on relatively simple pattern extraction, but there are ambitious goals for diverse media types
- ESSENCE summarizer

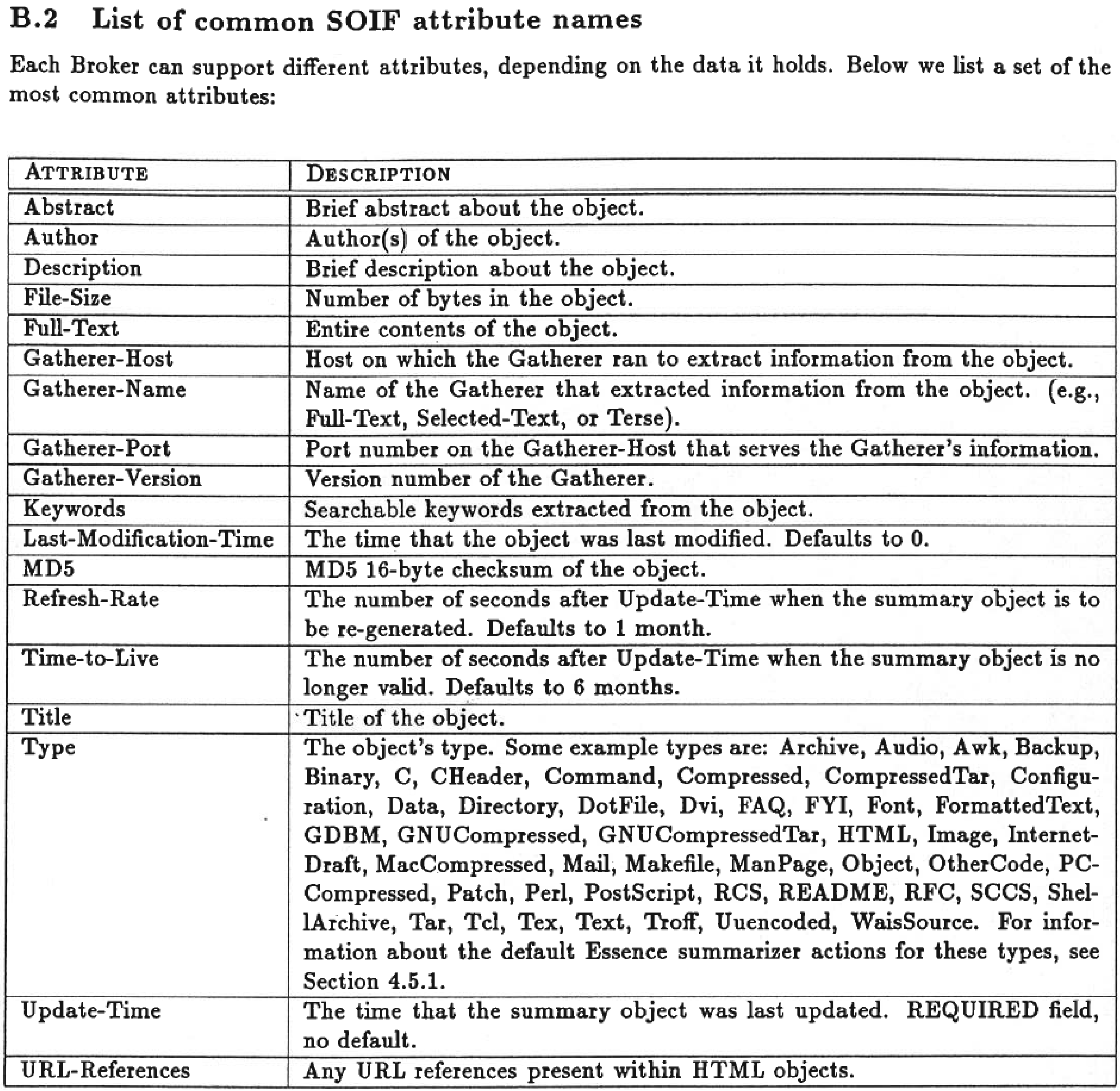
Example 1: Dublin Core record for an electronic version of an OCLC Report



Element Name	Content
Subject: scheme=LCSH:	Internet (Computer network) Cataloging of computer files Information networks Computer networks Libraries--Communication systems Information storage and retrieval systems
Title:	Assessing Information on the Internet: Toward Providing Library Services for Computer Mediated Communication
Author:	Martin Dillon
Author:	Erik Jul
Author:	Mark Burge
Author:	Carol Hickey
Publisher:	OCLC
Date:	1994
Identifier: Scheme=OCLC:	155653163X
Object type: Scheme=AACR2:	monograph
Form:	7 postscript files 1 Unix tar file
Relation:	For a Web page listing Internet accessible OCLC research publications go to: http://www.oclc.org/oclc/menu/resch doc.htm
Language:	English



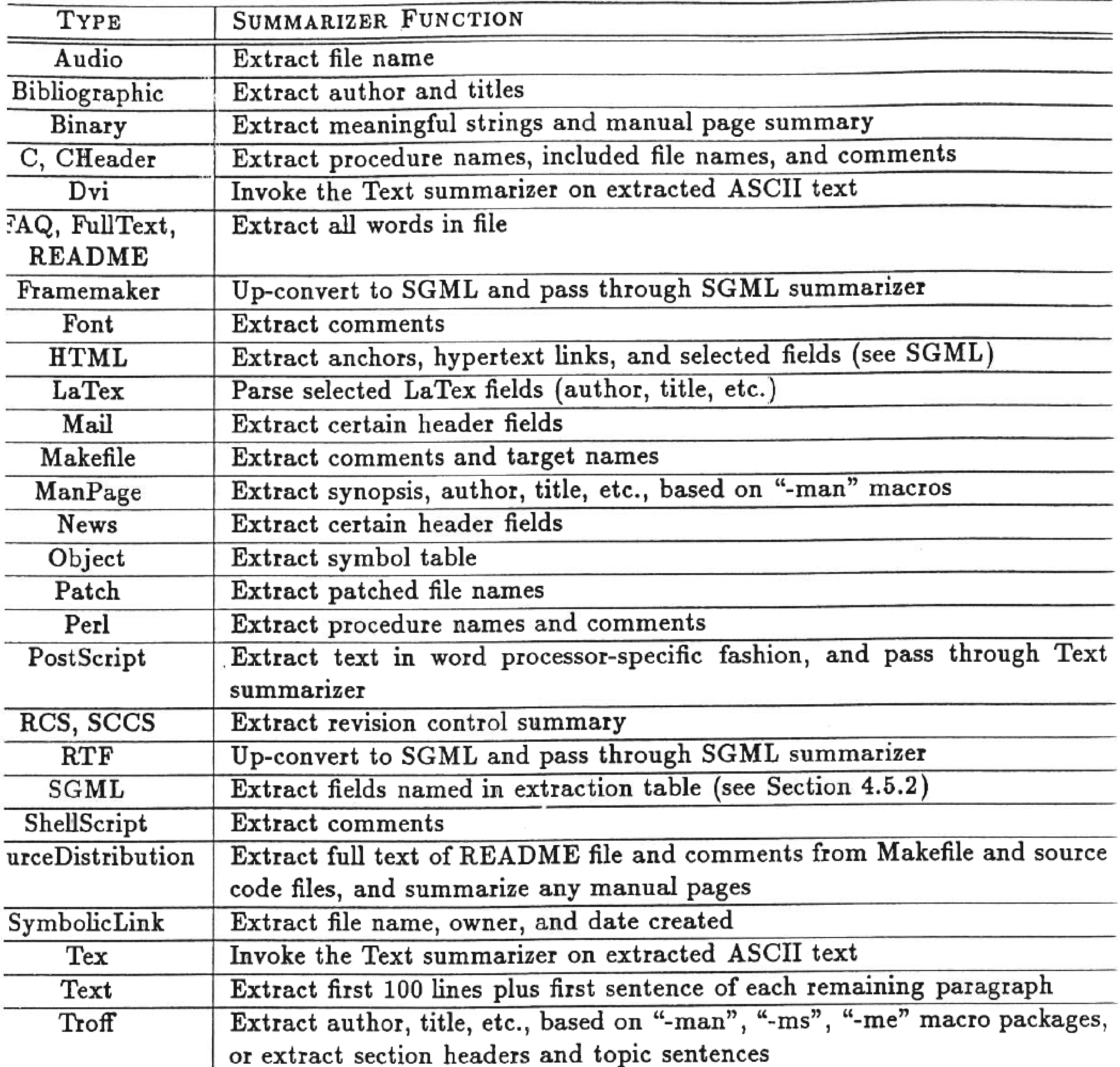
```
@FILE { ftp://tsx-11.mit.edu/pub/linux/docs/linux-doc-project/man-pages-1.4.lsm
Time-to-Live{7}:        9676800
Last-Modification-Time{9}:      781931042
Refresh-Rate{7}:        2419200
Gatherer-Name{25}:      Example Gatherer Number 2
Gatherer-Host{22}:      powell.cs.colorado.edu
Gatherer-Version{3}:    0.4
Type{3}:        LSM
Update-Time{9}: 781931042
File-Size{3}:   848
MD5{32}:        67377f3ea214ab680892c82906081caf
}

@FILE { ftp://ftp.cs.unc.edu/pub/faith/linux/man-pages-1.4.tar.gz
Time-to-Live{7}:        9676800
Last-Modification-Time{9}:      781931042
Refresh-Rate{7}:        2419200
Gatherer-Name{25}:      Example Gatherer Number 2
Gatherer-Host{22}:      powell.cs.colorado.edu
Gatherer-Version{3}:    0.4
Update-Time{9}: 781931042
Type{16}:       GNUCompressedTar
Title{48}:      Section 2, 3, 4, 5, 7, and 9 man pages for Linux
Version{3}:     1.4
Description{124}:       Man pages for Linux.  Mostly section 2 is complete.  Section
3 has over 200 man pages, but it still far from being finished.
Author{27}:     Linux Documentation Project
AuthorEmail{11}:        DOC channel
Maintainer{9}:  Rik Faith
MaintEmail{16}: faith@cs.unc.edu
Site{45}:       ftp.cs.unc.edu
sunsite.unc.edu
tsx-11.mit.edu
Path{94}:       /pub/faith/linux
/pub/Linux/docs/linux-doc-project/man-pages
/pub/linux/docs/linux-doc-project
File{20}:       man-pages-1.4.tar.gz
FileSize{4}:    170k
CopyPolicy{47}: Public Domain or otherwise freely distributable
Keywords{10}:   man
pages
```



B.2 List of common SOIF attribute names

Each Broker can support different attributes, depending on the data it holds. Below we list a set of the most common attributes:

Attribute	Description
Abstract	Brief abstract about the object.
Author	Author(s) of the object.
Description	Brief description about the object.
File-Size	Number of bytes in the object.
Full-Text	Entire contents of the object.
Gatherer-Host	Host on which the Gatherer ran to extract information from the object.
Gatherer-Name	Name of the Gatherer that extracted information from the object. (e.g., Full-Text, Selected-Text, or Terse).
Gatherer-Port	Port number on the Gatherer-Host that serves the Gatherer's information.
Gatherer-Version	Version number of the Gatherer.
Keywords	Searchable keywords extracted from the object.
Last-Modification-Time	The time that the object was last modified. Defaults to 0.
MD5	MD5 16-byte checksum of the object.
Refresh-Rate	The number of seconds after Update-Time when the summary object is to be re-generated. Defaults to 1 month.
Time-to-Live	The number of seconds after Update-Time when the summary object is no longer valid. Defaults to 6 months.
Title	Title of the object.
Type	The object's type. Some example types are: Archive, Audio, Awk, Backup, Binary, C, CHeader, Command, Compressed, CompressedTar, Configuration, Data, Directory, DotFile, Dvi, FAQ, FYI, Font, FormattedText, GDBM, GNUCompressed, GNUCompressedTar, HTML, Image, Internet-Draft, MacCompressed, Mail, Makefile, ManPage, Object, OtherCode, PC-Compressed, Patch, Perl, PostScript, RCS, README, RFC, SCCS, ShellArchive, Tar, Tcl, Tex, Text, Troff, Uuencoded, WaisSource. For information about the default Essence summarizer actions for these types, see Section 4.5.1.
Update-Time	The time that the summary object was last updated. REQUIRED field, no default.
URL-References	Any URL references present within HTML objects.



Type	Summarizer Function
Audio	Extract file name
Bibliographic	Extract author and titles
Binary	Extract meaningful strings and manual page summary
C, CHeader	Extract procedure names, included file names, and comments
Dvi	Invoke the Text summarizer on extracted ASCII text
FAQ, FullText, README	Extract all words in file
Framemaker	Up-convert to SGML and pass through SGML summarizer
Font	Extract comments
HTML	Extract anchors, hypertext links, and selected fields (see SGML)
LaTex	Parse selected LaTex fields (author, title, etc.)
Mail	Extract certain header fields
Makefile	Extract comments and target names
ManPage	Extract synopsis, author, title, etc., based on "-man" macros
News	Extract certain header fields
Object	Extract symbol table
Patch	Extract patched file names
Perl	Extract procedure names and comments
PostScript	Extract text in word processor-specific fashion, and pass through Text summarizer
RCS, SCCS	Extract revision control summary
RTF	Up-convert to SGML and pass through SGML summarizer
SGML	Extract fields named in extraction table (see Section 4.5.2)
ShellScript	Extract comments
urceDistribution	Extract full text of README file and comments from Makefile and source code files, and summarize any manual pages
SymbolicLink	Extract file name, owner, and date created
Tex	Invoke the Text summarizer on extracted ASCII text
Text	Extract first 100 lines plus first sentence of each remaining paragraph
Troff	Extract author, title, etc., based on "-man", "-ms", "-me" macro packages, or extract section headers and topic sentences

```
author{15}: Joe T. Slacker
```

Using the META tags, HTML authors can easily add a list of keywords to thei

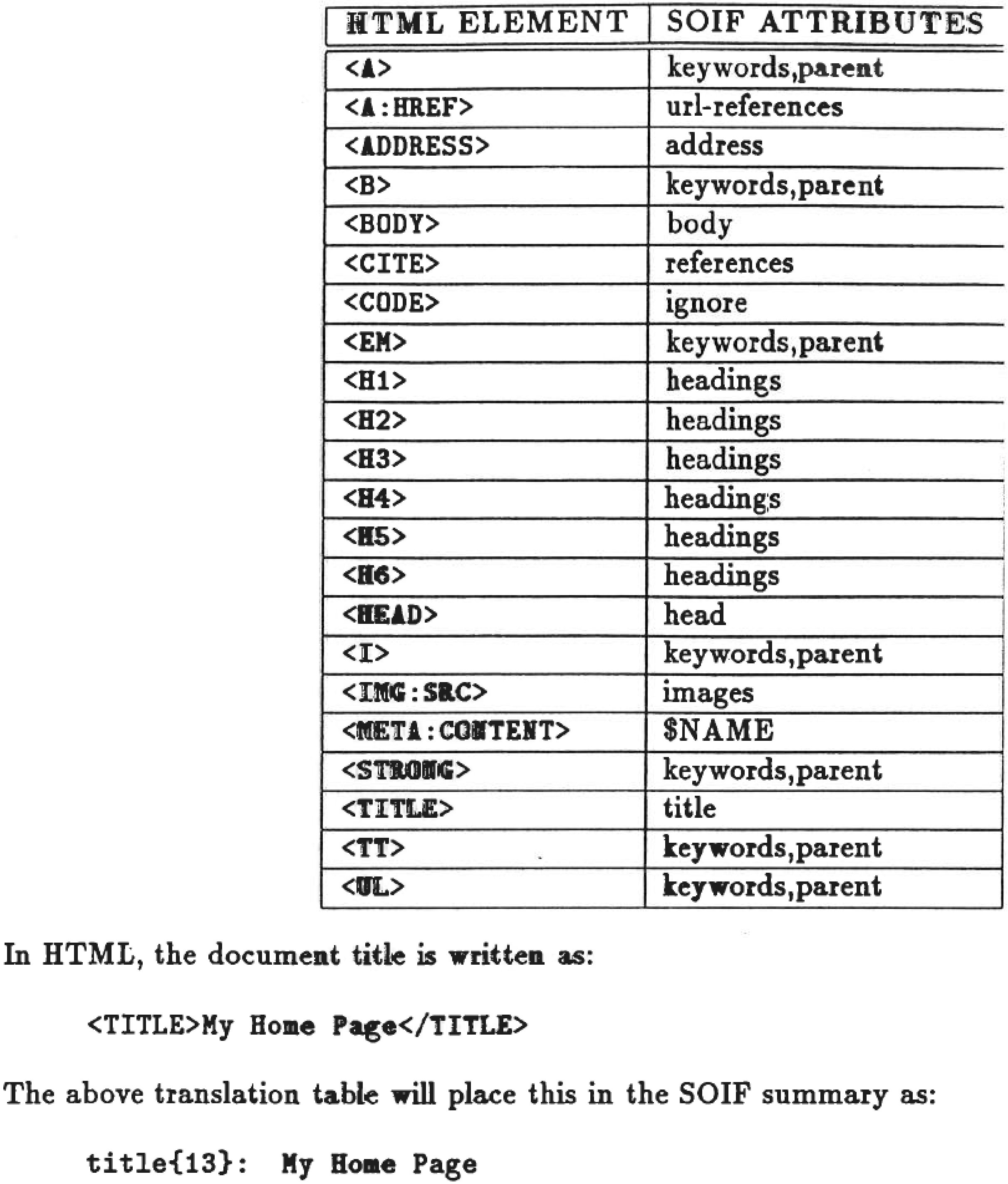
```
<META NAME="keywords"  CONTENT="word1 word2">
<META NAME="keywords"  CONTENT="word3 word4">
```

Other examples A very terse HTML summarizer could be specified with a emphasized words into the keywords attribute:

HTML ELEMENT	SOIF ATTRIBUTES
<A>	keywords
<B>	keywords
<EM>	keywords
<H1>	keywords
<H2>	keywords
<H3>	keywords
<I>	keywords
<META:CONTENT>	$NAME
<STRONG>	keywords
<TITLE>	title,keywords
<TT>	keywords

Conversely, a full-text summarizer can be easily specified with only:

HTML ELEMENT	SOIF ATTRIBUTES
<HTML>	full-text
<TITLE>	title,parent



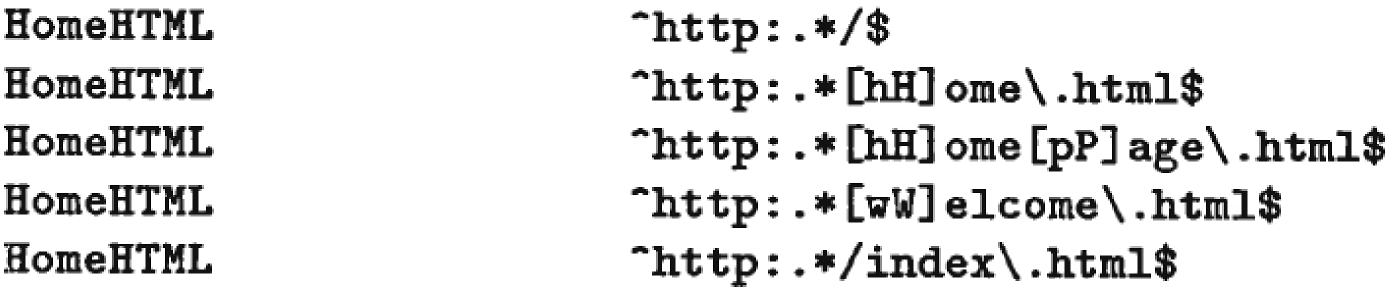
HTML ELEMENT	SOIF ATTRIBUTES
<A>	keywords,parent
<A:HREF>	url-references
<ADDRESS>	address
<B>	keywords,parent
<BODY>	body
<CITE>	references
<CODE>	ignore
<EM>	keywords,parent
<H1>	headings
<H2>	headings
<H3>	headings
<H4>	headings
<H5>	headings
<H6>	headings
<HEAD>	head
<I>	keywords,parent
<IMG:SRC>	images
<META:CONTENT>	$NAME
<STRONG>	keywords,parent
<TITLE>	title
<TT>	keywords,parent
<UL>	keywords,parent

In HTML, the document title is written as:

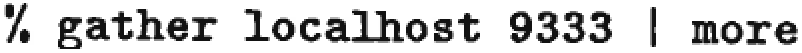
```
<TITLE>My Home Page</TITLE>
```

The above translation table will place this in the SOIF summary as:

```
title{13}:  My Home Page
```



```
HomeHTML                      ^http:.*/$
HomeHTML                      ^http:.*[hH]ome\.html$
HomeHTML                      ^http:.*[hH]ome[pP]age\.html$
HomeHTML                      ^http:.*[wW]elcome\.html$
HomeHTML                      ^http:.*/index\.html$
```



```
% gather localhost 9333 | more
```

Double-Edged Sword

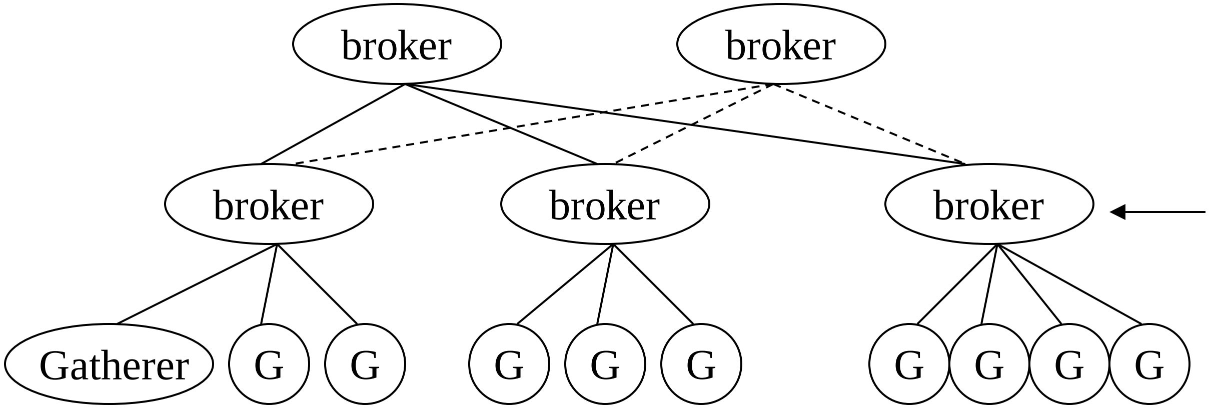
- Pros:
  - Providers have best knowledge of content (much info not even on pages, other ambiguous) – e.g. copyright, author
- Cons:
  - Incentive to lie (to get indexed)
  - Search engines randomly verify if inconsistent/outright lie -> blacklisted
  - Reliable rating scheme/consortium/modification

Broker subsystems

① Database of what resources are where

(Gather finds/extracts, broker serves and maintains)

May be hierarchical
(and definitely distributed)



Responsible for its regions of the web

Broker subsystems

Currently brokers divided primarily by

- document/data type(images, phone #’s)
- content type(e.g. tech reports or news stories)

Specialists in material of given type

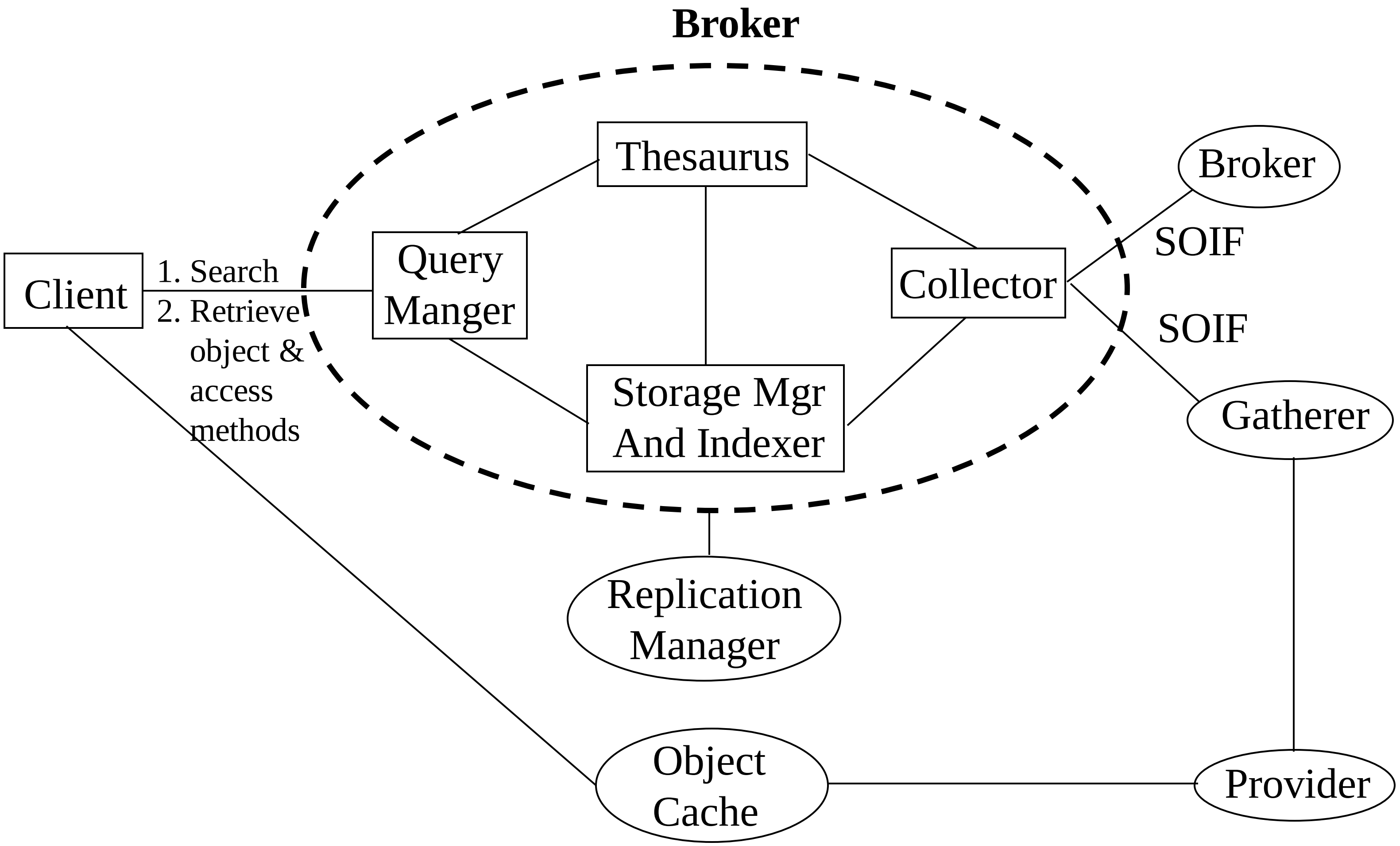
Problem : where do brokers know
where to find data of given type?

May also be partitioned by geographical location

Europe/Asia specialist(country specific trees)

Language specialist(charset expertise)

.EDU/.Com/.AOL specialists



Harvest Software Components

Index/Search subsystems

(built on top of or into brokers)

① GLIMPSE

Traditional word → (Region or document) inverted index

Extension : Variable region size

Comput* → Documents where occur
Graphics → Paragraphs where occur
Hopkins → Paragraphs where occur
Khudanpur →

Supports only as much region detail as necessary
(if same thing only occurs once, why not store exact location?)

Claims : very small index(2 – 4 % of original text size)
but flexible/supportive of collocational query
(use a grep to search regions of potential co-occurrence)

→ Uses Essence objects

Index/Search subsystems

② NEBULA

Supports hierarchical classification schemes(automatic Yahoo!)

And “Views”(precomputed query responses)

basically vector clusters that are returned
As full relevance set

selling point : fast query response
(don’t do individual document tests)
but less flexible

Precompute : compute graphics
commodities trading
venture capital

Caching Subsystem

- Motivation for caching

  Minimize network traffic by reusing frequently requested items

  (e.g. LFU cache replacement strategy)

  Least Frequently Used

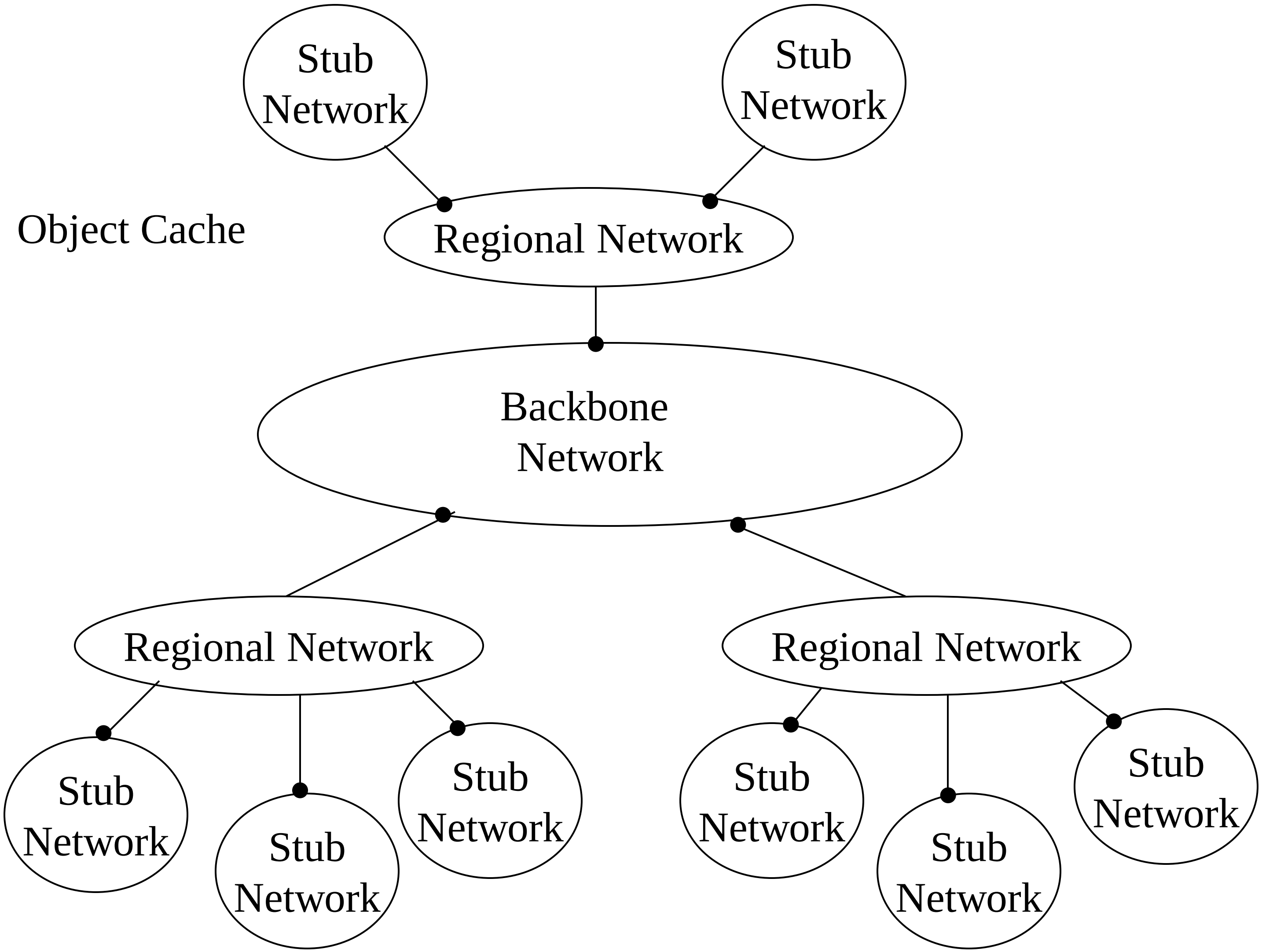
- Hierarchical caching

  Larger caches stored on server shared by many machines

  if not in my local cache, use subnet's cache

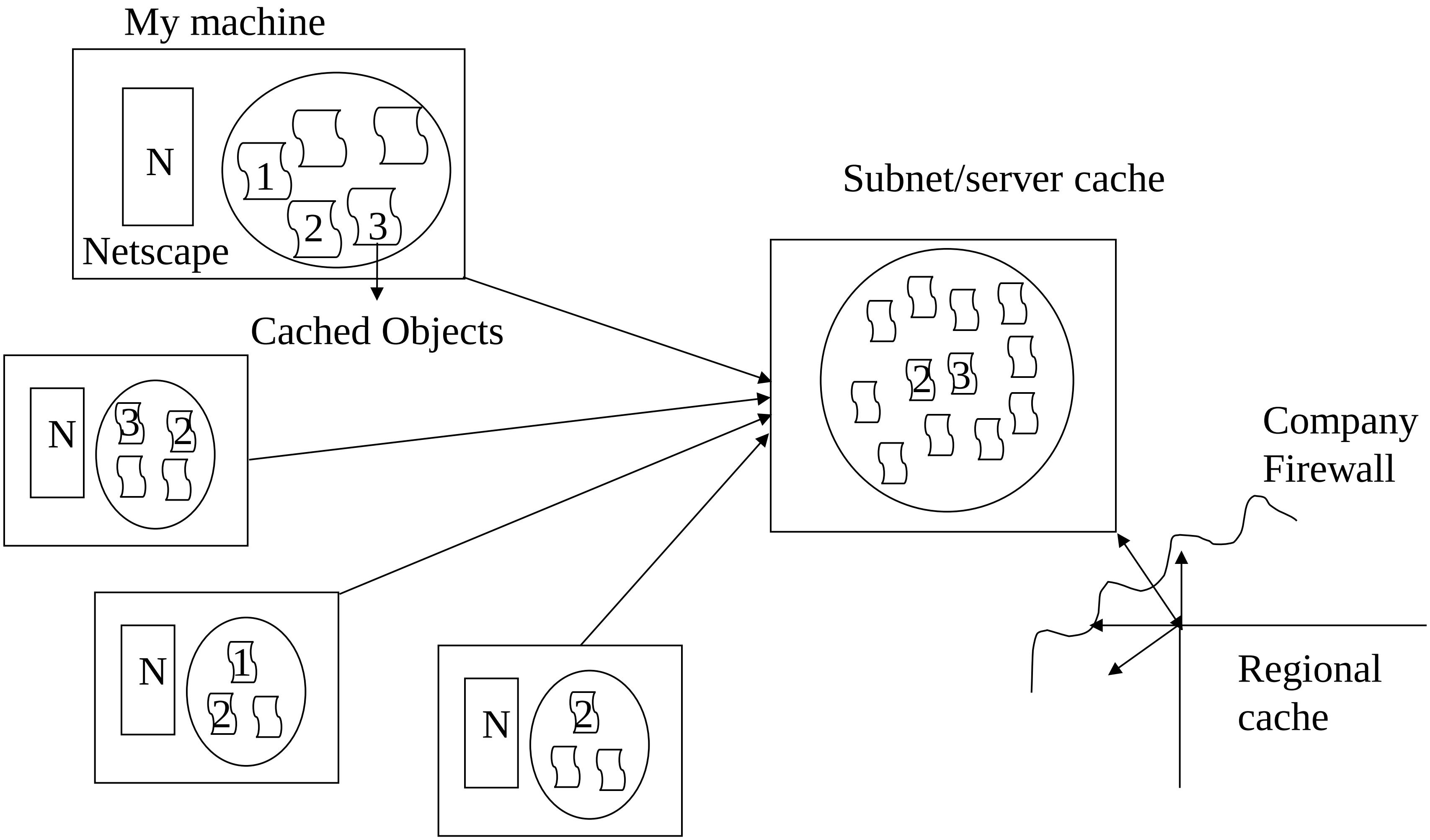
  (often provided by firewall software)

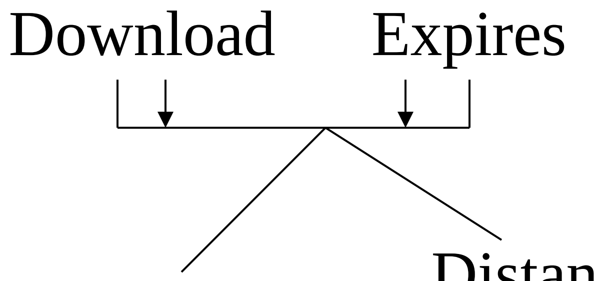
- Object Cache



Hierarchical Cache Arrangement

Caching Subsystem





Download Expires

P(ask for update) = f (Distance to expiration date , Distance from download , Confidence,

Reliability of provider’s estimates , My budget)

Cache Subsystem

- Problem with Caching:
  - I don’t know if a cache object has been updated before its next use without checking(at least HEAD)
  - no mechanism in web for remotely forced cache flush

Expires: 0

Expires : Thu, 16 May 2013 14:40:30 GMT

Only supports predictive expiration
(says in advance how long a copy may be used)
But what if unexpected change before expiration or unchanged
persistence afterwords?

Object Cache/Replicator

Data access efficiency

- Log of use(LRU : Least Recently Used)
- Most popular files distributed access network sites

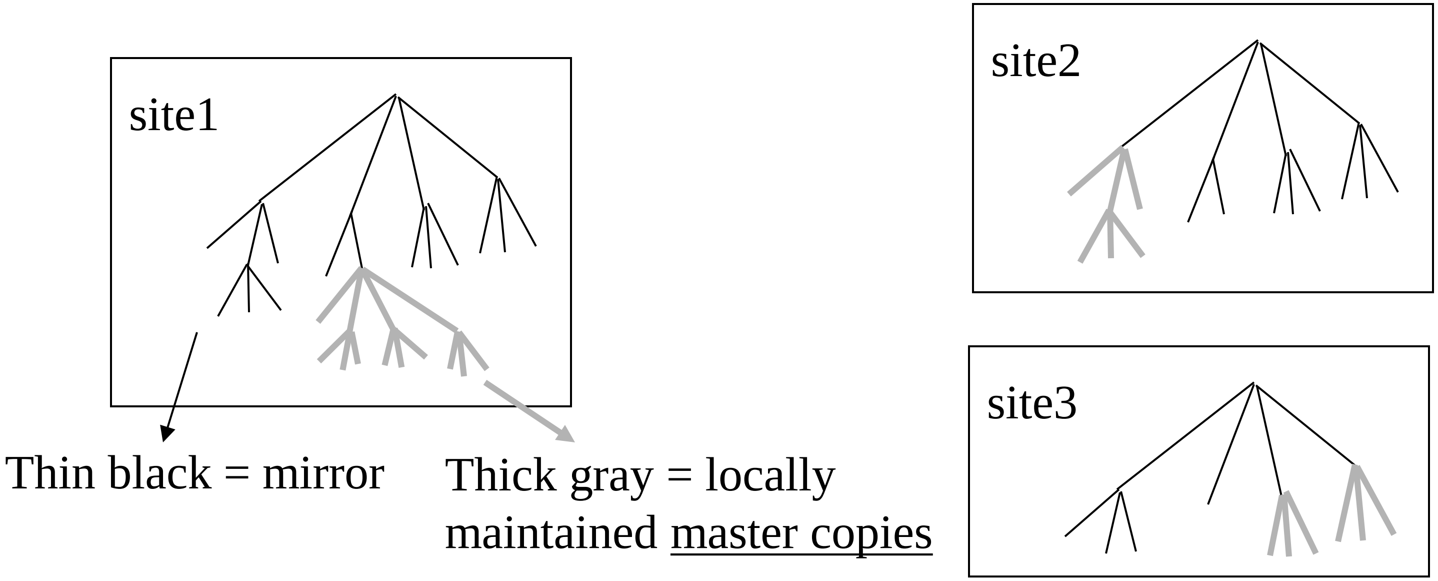
(in local storage)

→ problem of efficient expiration, version control

Harvest Replication Subsystem

Motivation : like to have(complete) regional copies with mechanism to ensure active consistency updates

mirror-d(replication tool for Harvest using ftp mirror)



Thin black = mirror

Thick gray = locally maintained master copies

Harvest Replication Subsystem

“mirror-d” replication tool – weakly consistent replicated tree of files

Motivation : multiple copies for future access

(e.g. Europe, North America) ← replication domain

Problem : maintaining data consistency

(using ftp-mirrors)

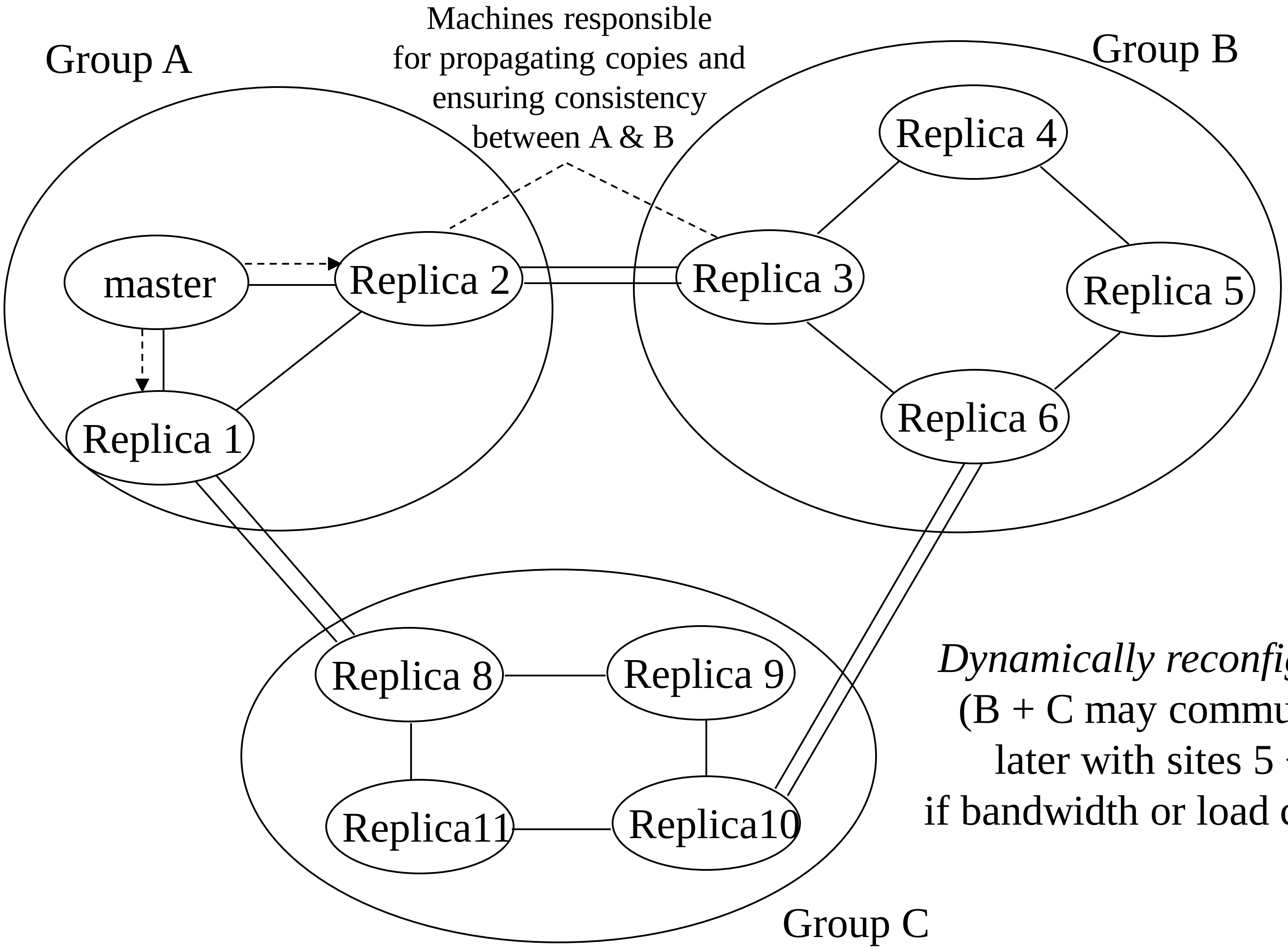
① Logical topology

→ replication subgroups that coordinate consistency internally share updates within subgroup domain/domain.

Physical issues(network bandwidth/usage)

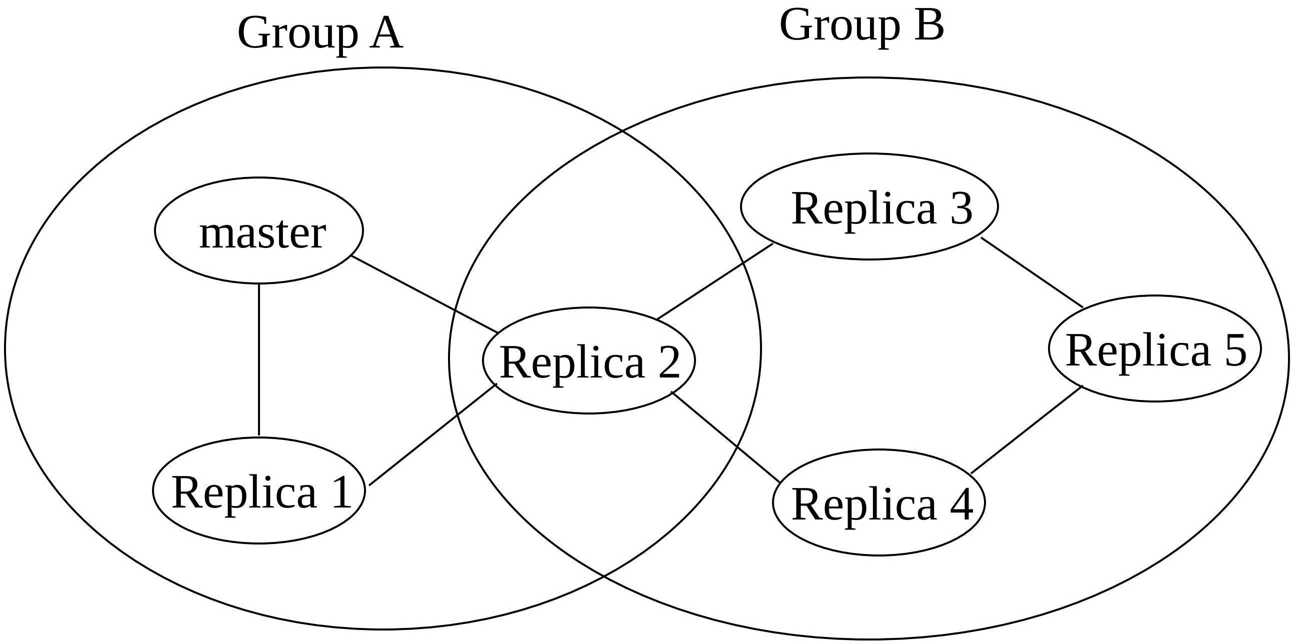
help determine how replication domains propagate(flood) updates among its neighbors

Replication Group



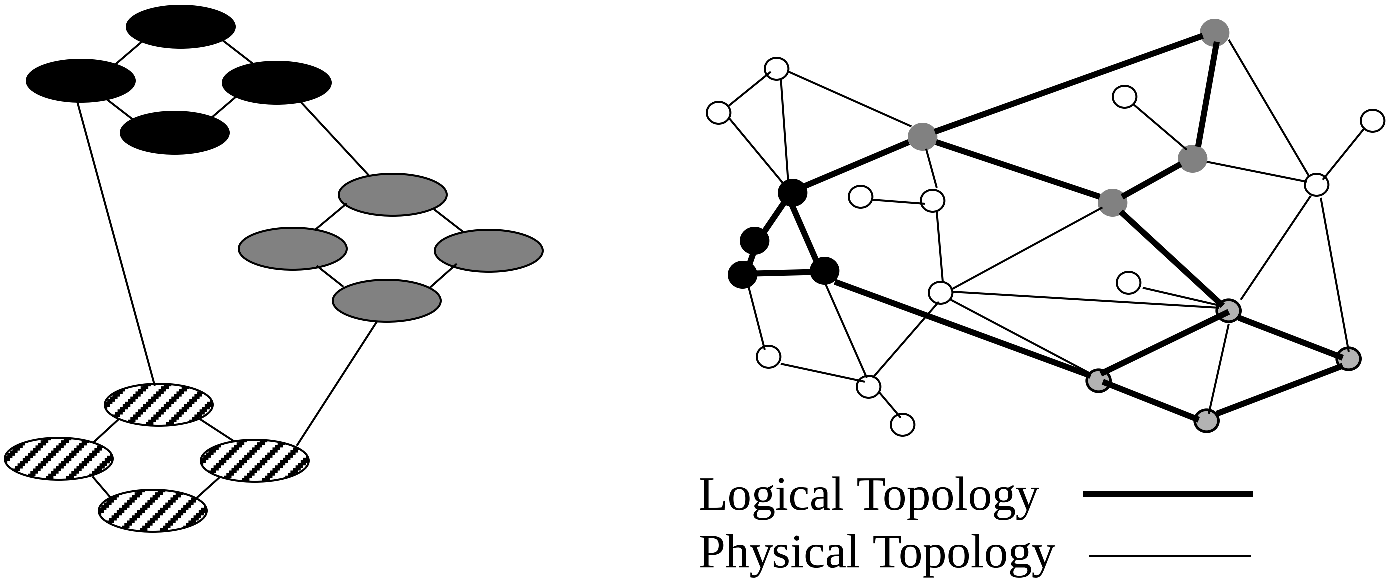
Dynamically reconfigurable
(B + C may communicate
later with sites 5 + 11
if bandwidth or load changes)

Although Replication Domain members are stable,
Pathways for inter-domain communication may change
Based on dynamic properties of load + bandwidth



Replicator System Overview

Logical inter-domain network topology is a subset of the full physical topology
(and is dynamically reconfigurable based on network load and bandwidth)



● Group 1 member ● Group 2 member ● Group 3 member ○ Non-group member

Replication domains and physical versus logical update topology

Replication Subsystem

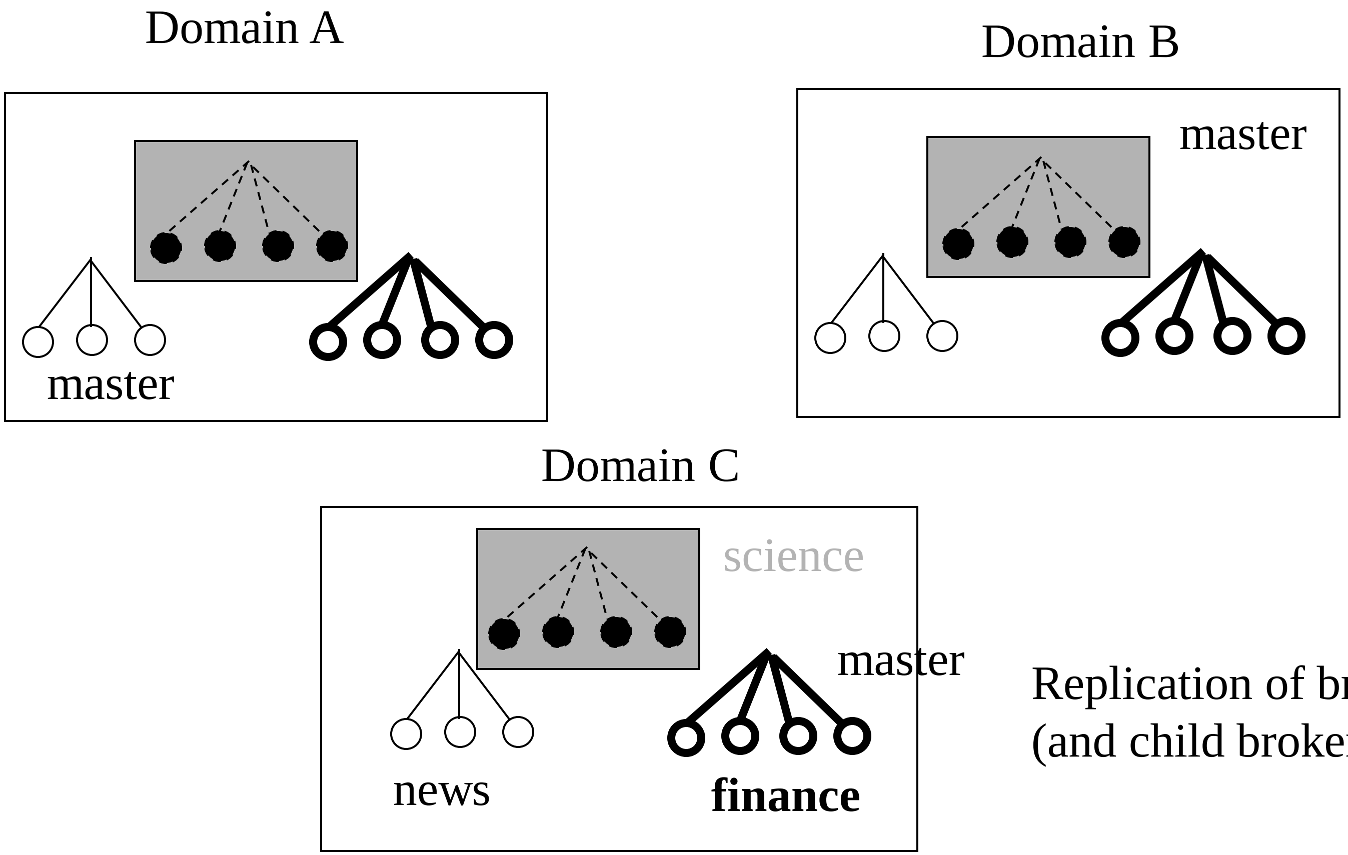
① Active consistent updates

(if a server changes its master copy, it notifies mirror sites)

② Harvest supports replication domains

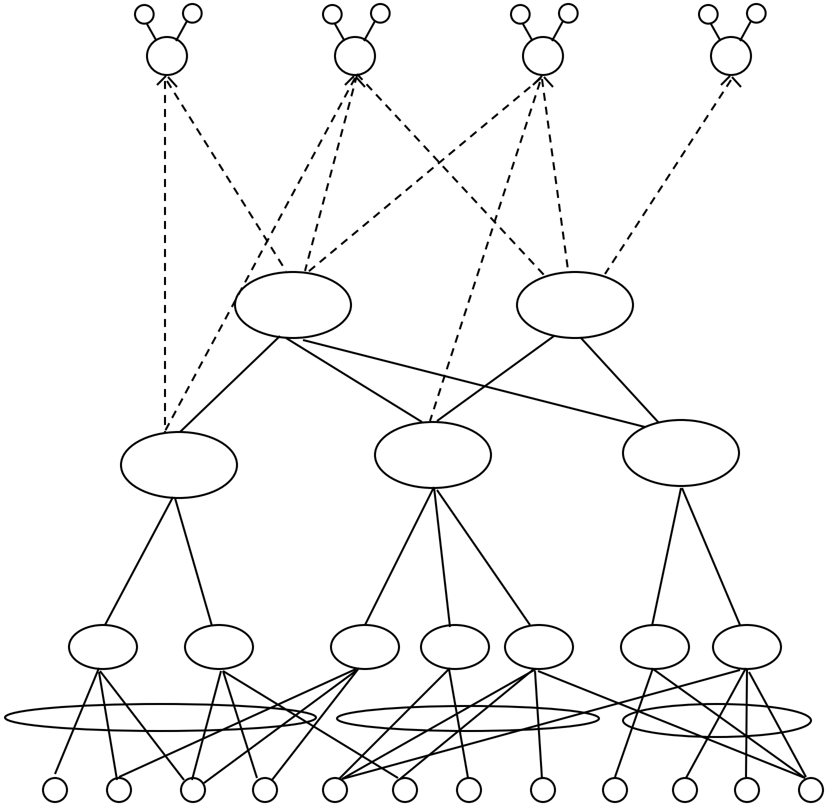
- mirroring within domain carefully coordinated/synchronized
- Mirroring/replication between domains involves gradual propagation of changes(between sites responsible for inter-domain communication)

Replication in Broker world



Replication of brokers
(and child brokers)

The (Future) Organization of the WEB



User agents – goal directed
extraction, analysis,
even dialog

Meta Brokers – meta search
collection/query fusion

Brokers(Index, Search)

Gatherers(Analyze, label) extract "essence"

Finders(Scouts, Spiders) – map + locate page

Content (Web pages + providers)