Web Dynamics

Web Archiving

Dr. Marc Spaniol

Saarbrücken, May 27, 2010

Agenda

- Introduction
  - Indexing vs. archiving
  - Temporal coherence of Web archives
- Aspects of Web archiving
  - Selection
  - Capturing
    - Conceptual approaches
    - Coherence aware archiving
    - Quantifying (in-)coherence
  - Archiving
  - Hosting
- Summary
- References

Indexing vs. Archiving

- Indexing
  - Completeness
  - Access to content
  - Scalability (speed)
  - Efficiency
  - Freshness

$\Rightarrow$ “Taking a Photo”

- Archiving
  - Completeness
  - Access to content
  - Scalability (coverage)
  - Authenticity
  - Coherence
  - Durability

$\Rightarrow$ “Shooting a Movie”

The Challenge of Web Archiving

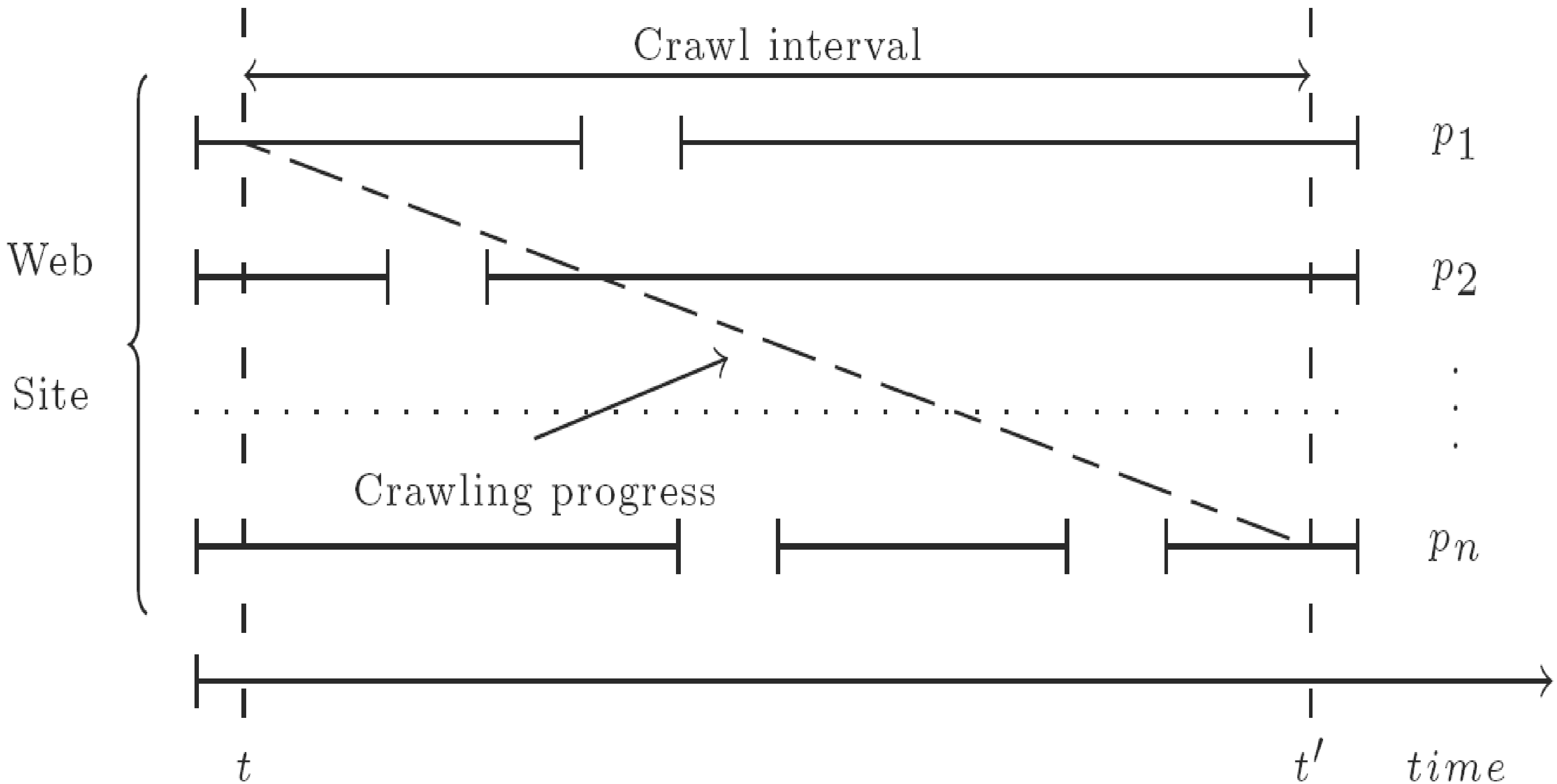
- World Wide Web
  - A disorganized free-for-all
  - Very little metadata
  - Unpredictable additions, deletions, modifications
  - No (coordinated) preservation strategy

- HTTP cannot ask for only new or modified contents
  - *Timestamps* have limited benefit
  - No list of pages that have been deleted, changed, and added
  - *Each* content must be requested, one at a time, *by name*

- There is no “SELECT *” in HTTP
  - Crawlers can only GET one resource at a time, by name
  - HTTP cannot give a crawler a list of all URLs for the site

⇒ Undiscovered or hidden resources will not be captured or refreshed
⇒ “Strategy” required

Temporal Coherence of Web Archives



The Challenge of Archive Coherence

- Crawler operations
  - Visit (pages)
    - Extract (links from pages)
    - Compare (versions of pages)
  - Follow (links)
- Website operations
  - Modifications “inside” pages
    - Content (text)
    - Structure (links)
  - Modifications “inside” site
    - Page creation
    - Page deletion

Taking place in parallel

⇓

Potentially incoherent

Potential Pitfalls in Web Archiving

- Crawling takes a long (!) time
  - Politeness
  - Multiple seeds per crawl
  - Spam
- Crawlers aren't "really" smart
  - Highly volatile against dynamics in CMS
  - Easy to be trapped, if not exactly configured
  - Doesn't recognize patterns of "identical" contents
  - ⇒ Pre-analysis of site(s) needed
- Some examples of crawler behavior
  - Enjoy link generation from JavaScript, PHP, etc.
  - Tend to go for shopping
  - Like time travelling in calendars

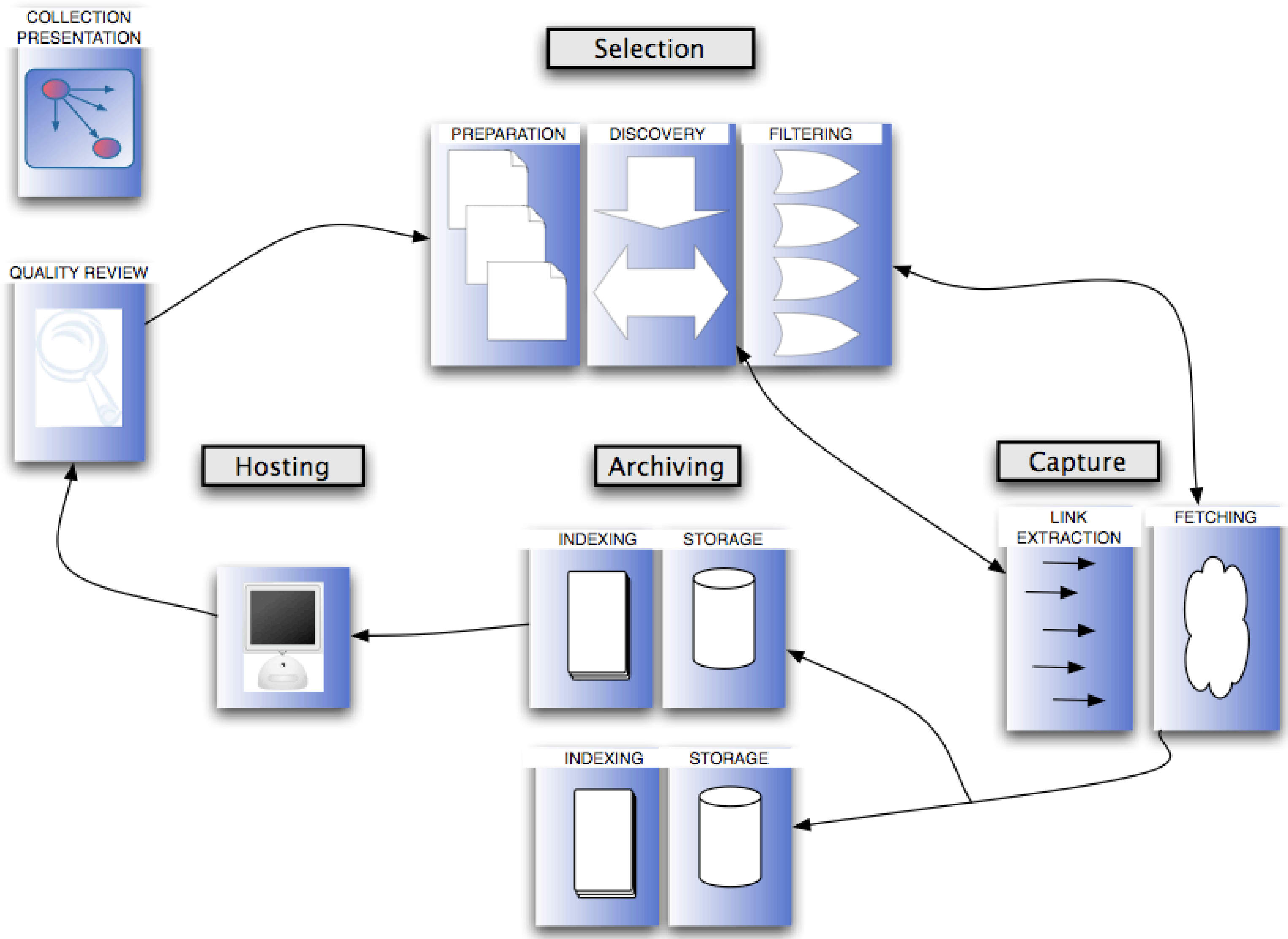
⇒ Crawling is simply "unpredictable"
⇒ Crawlers need "constant" monitoring

⇒ Archive in Danger!

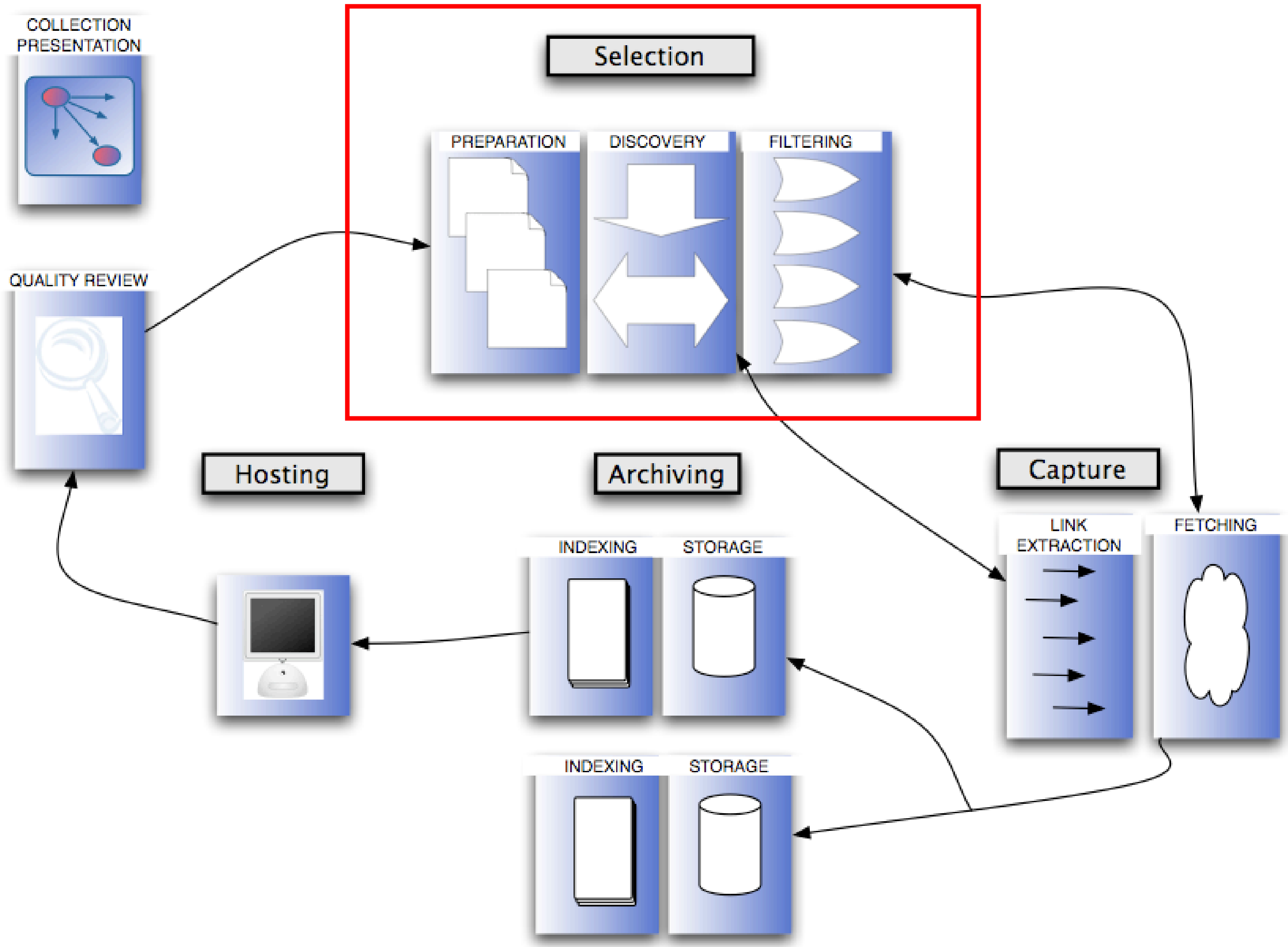
Smart(er) Crawling Strategies

Evaluation of Crawl Coherence

Aspects of Web Archiving



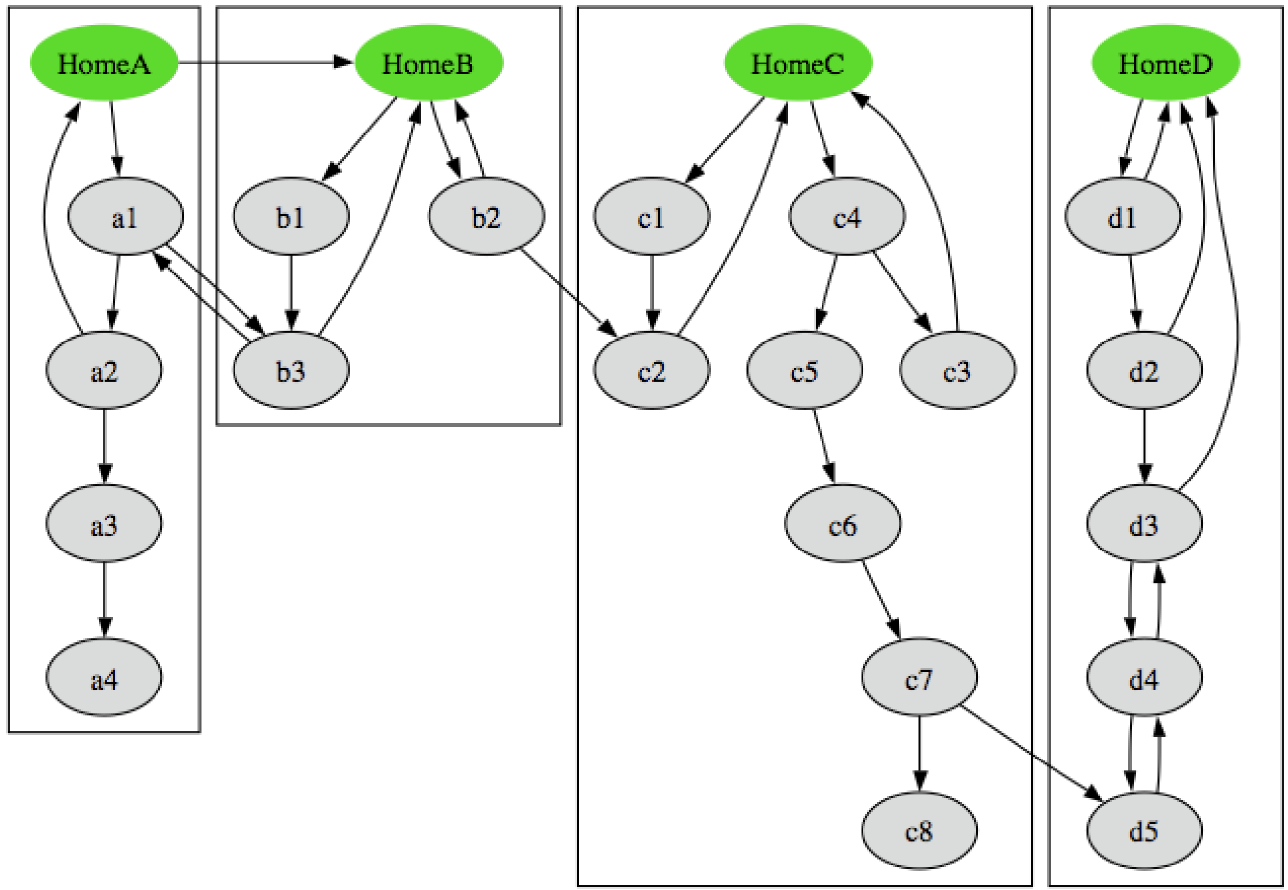
Selection


COLLECTION PRESENTATION
Selection
PREPARATION
DISCOVERY
FILTERING
QUALITY REVIEW
Hosting
Archiving
Capture
LINK EXTRACTION
FETCHING
INDEXING
STORAGE
INDEXING
STORAGE

Selection of Seed(s) and Scope

- Entry point / seed:
  Where the capturing process (crawl) starts. Top of the hypertext path that will be followed.

- Scope:
  The extent of the area that will be included in the gathering, as defined by criteria applicable to each node.

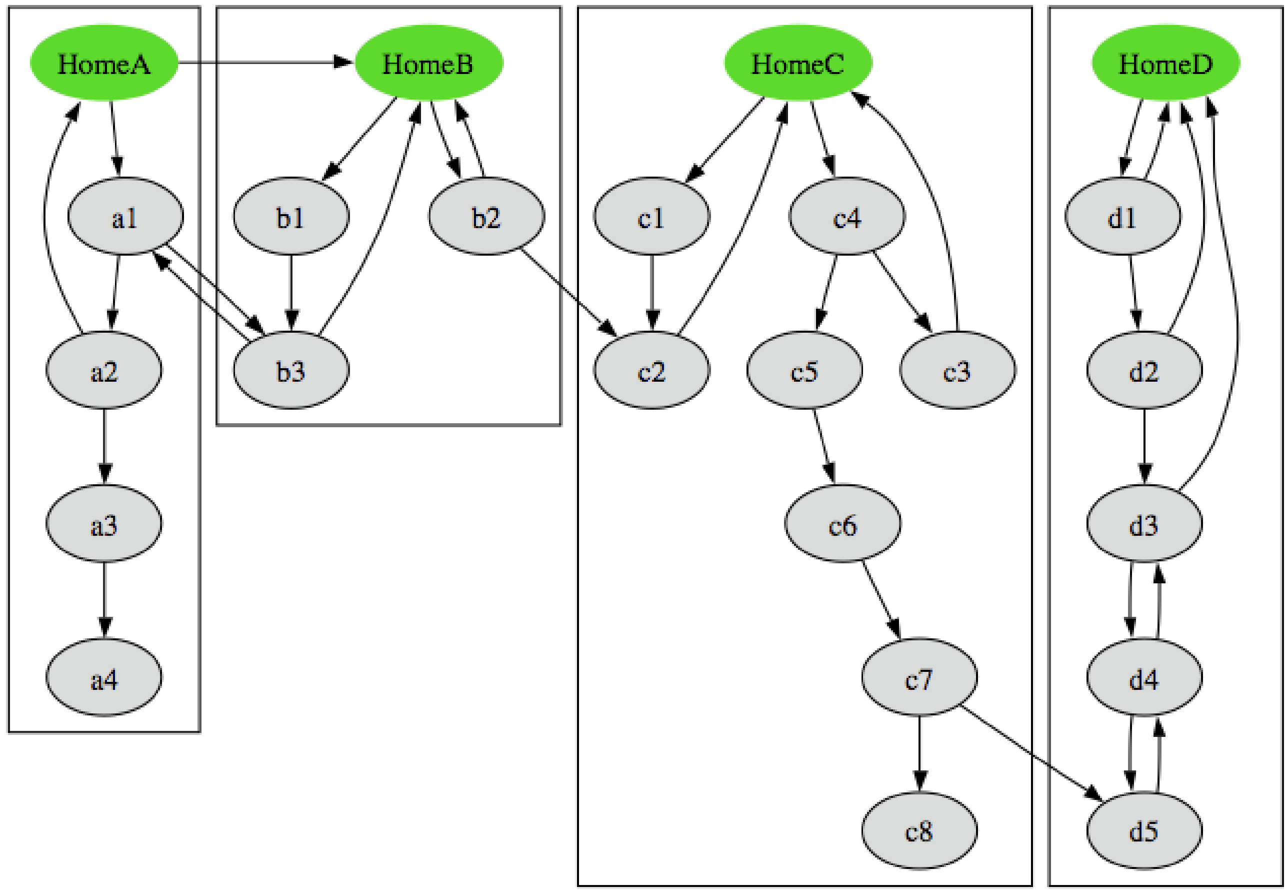


Web Archiving

Dr. Marc Spaniol

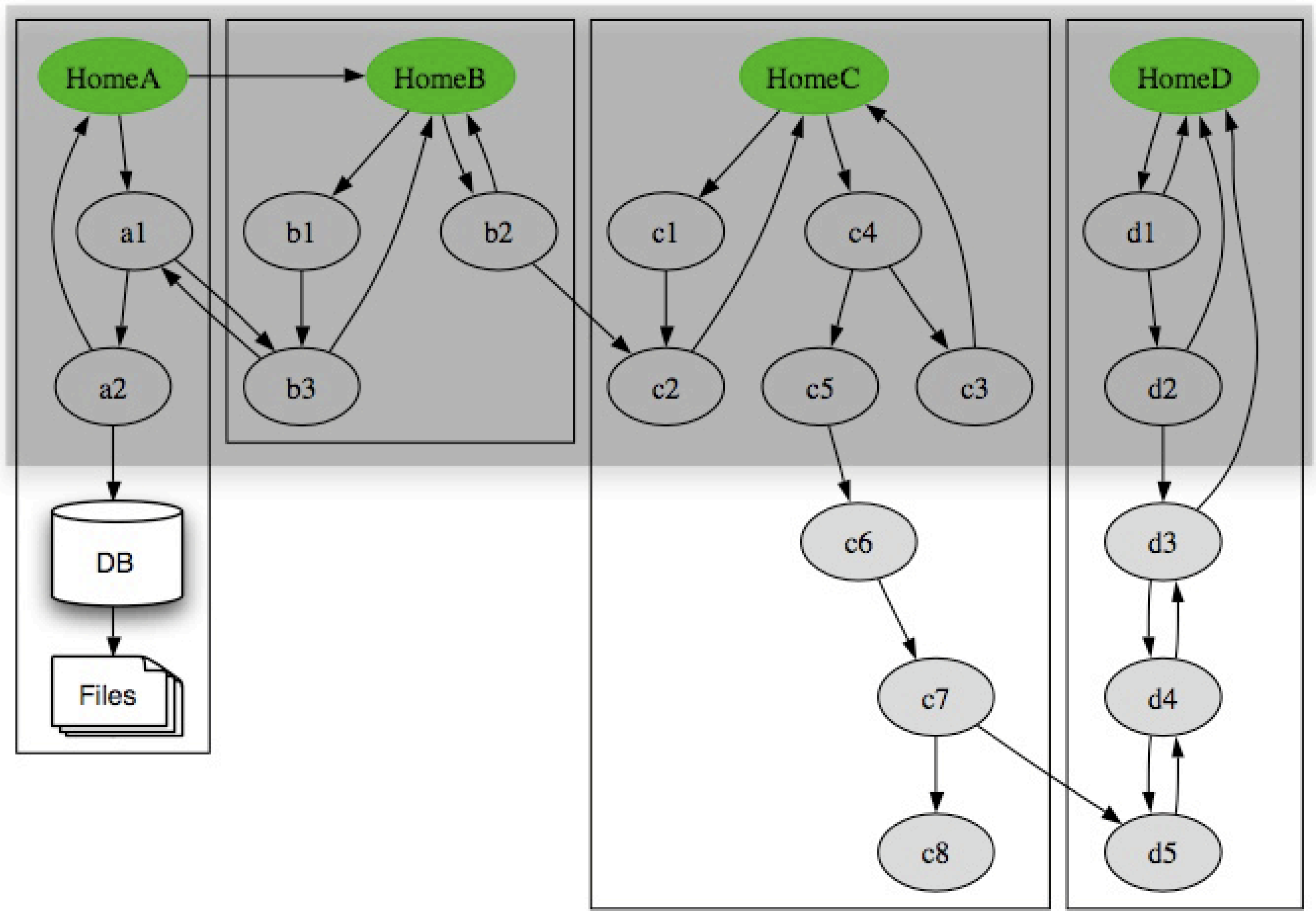
Completeness

- Vertically: Number of relevant nodes found from entry point
- Horizontally: Number of relevant entry points found within the designated perimeter



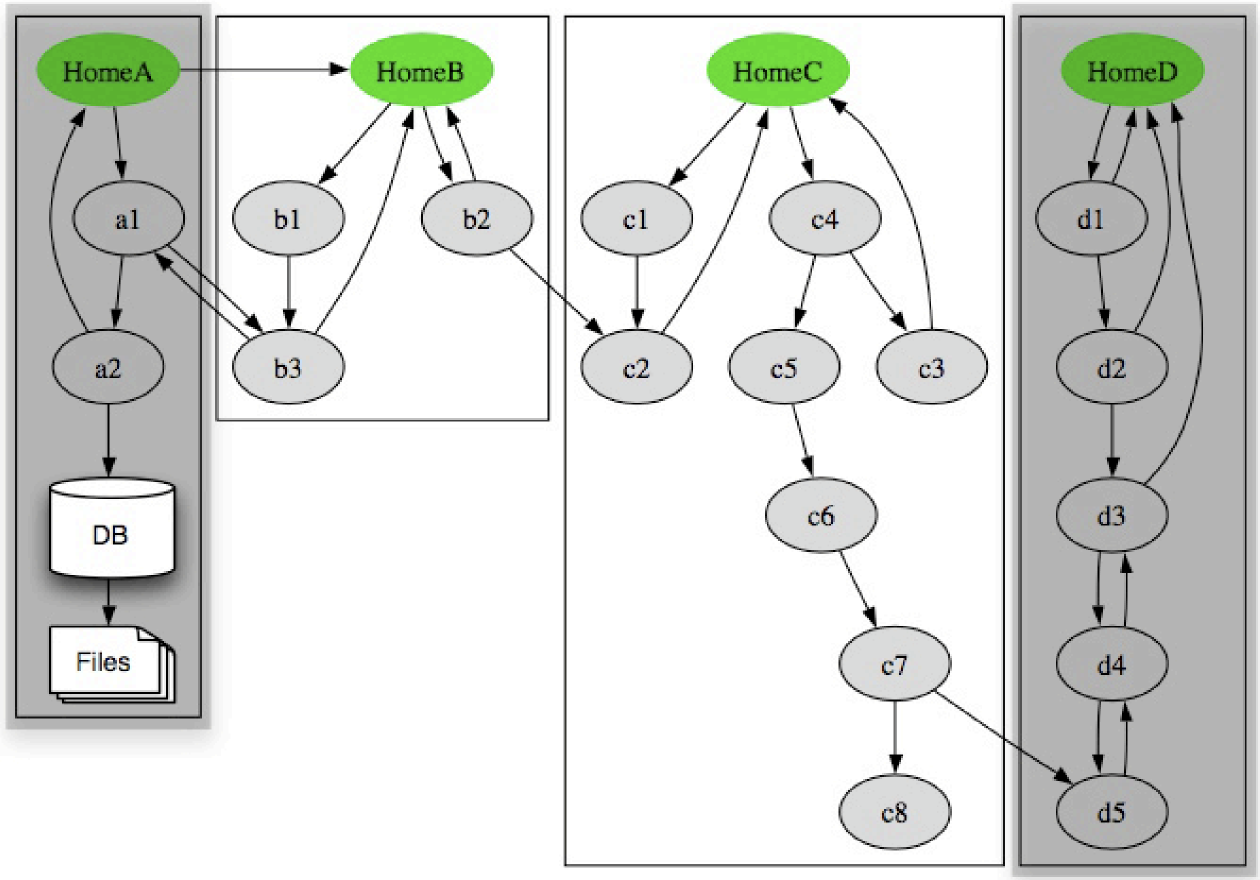
Extensive Collection

- Horizontal completeness is preferred to vertical completeness
- Holistic, domain based, or topic-centric archiving



Intensive Collection

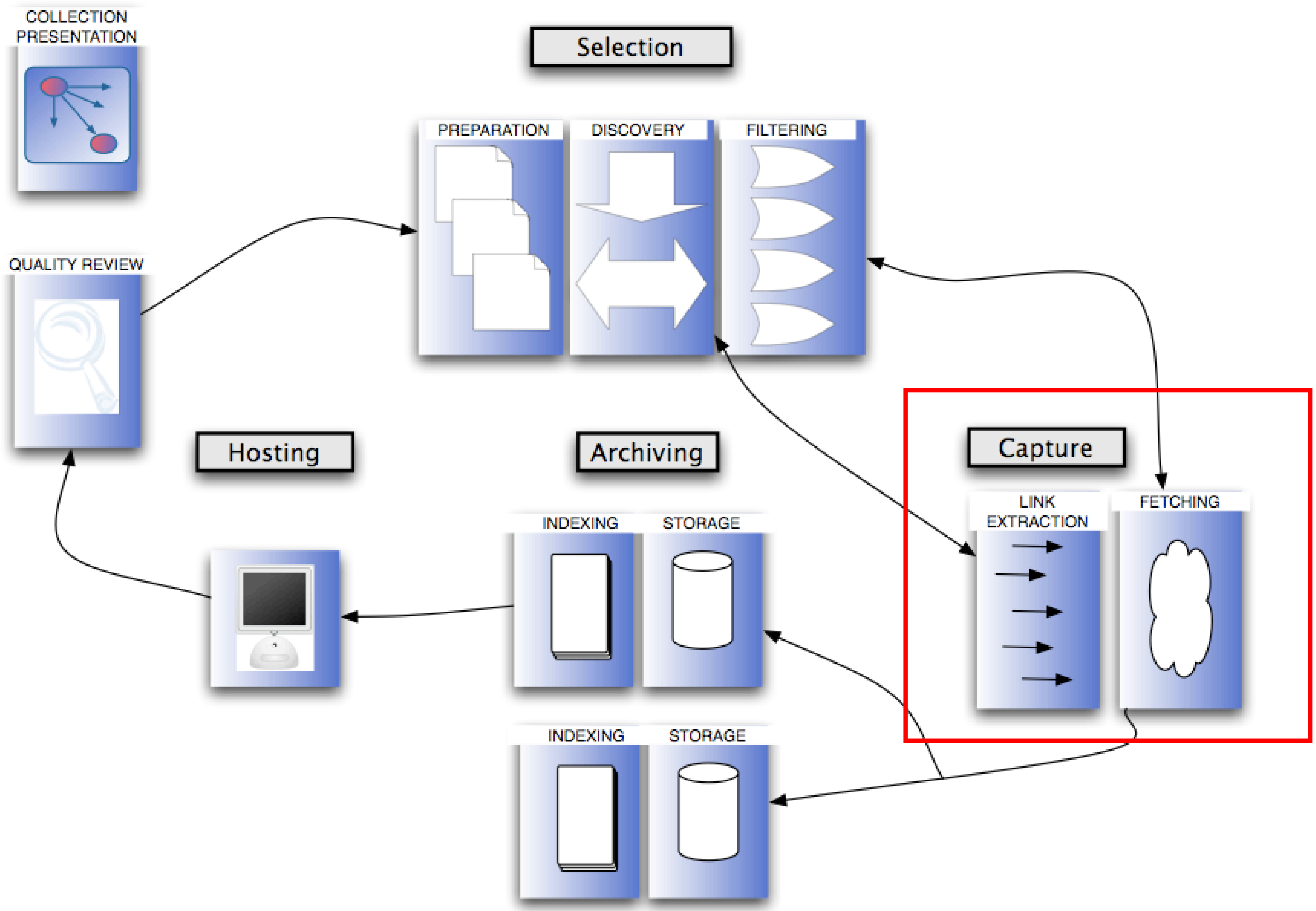
- Vertical completeness is preferred to horizontal completeness
- Site-based archiving
- Defines the high level target of a collection
- Explicit exclusion to avoid duplicate content with other collections



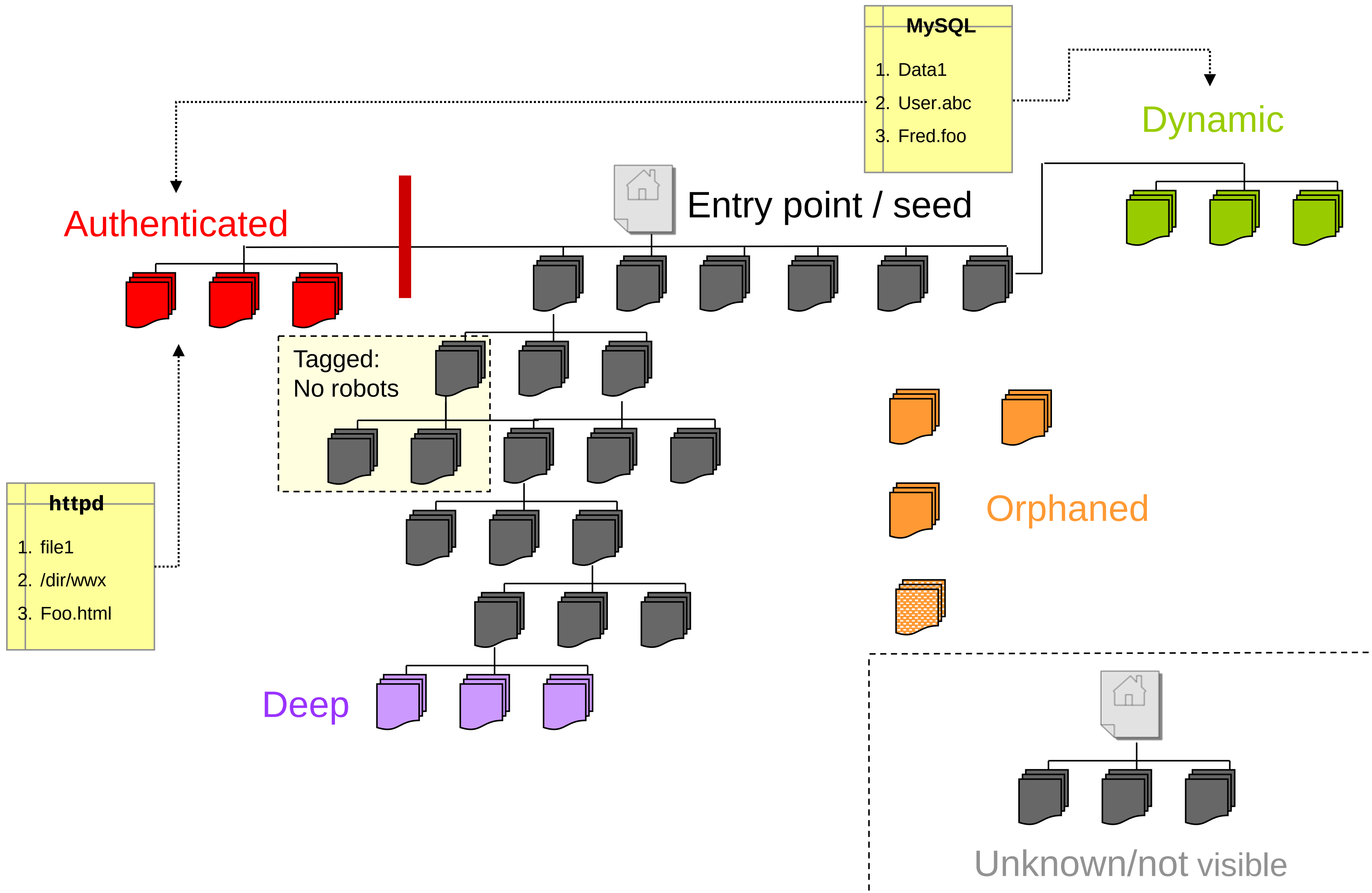
Web Archiving

Dr. Marc Spaniol

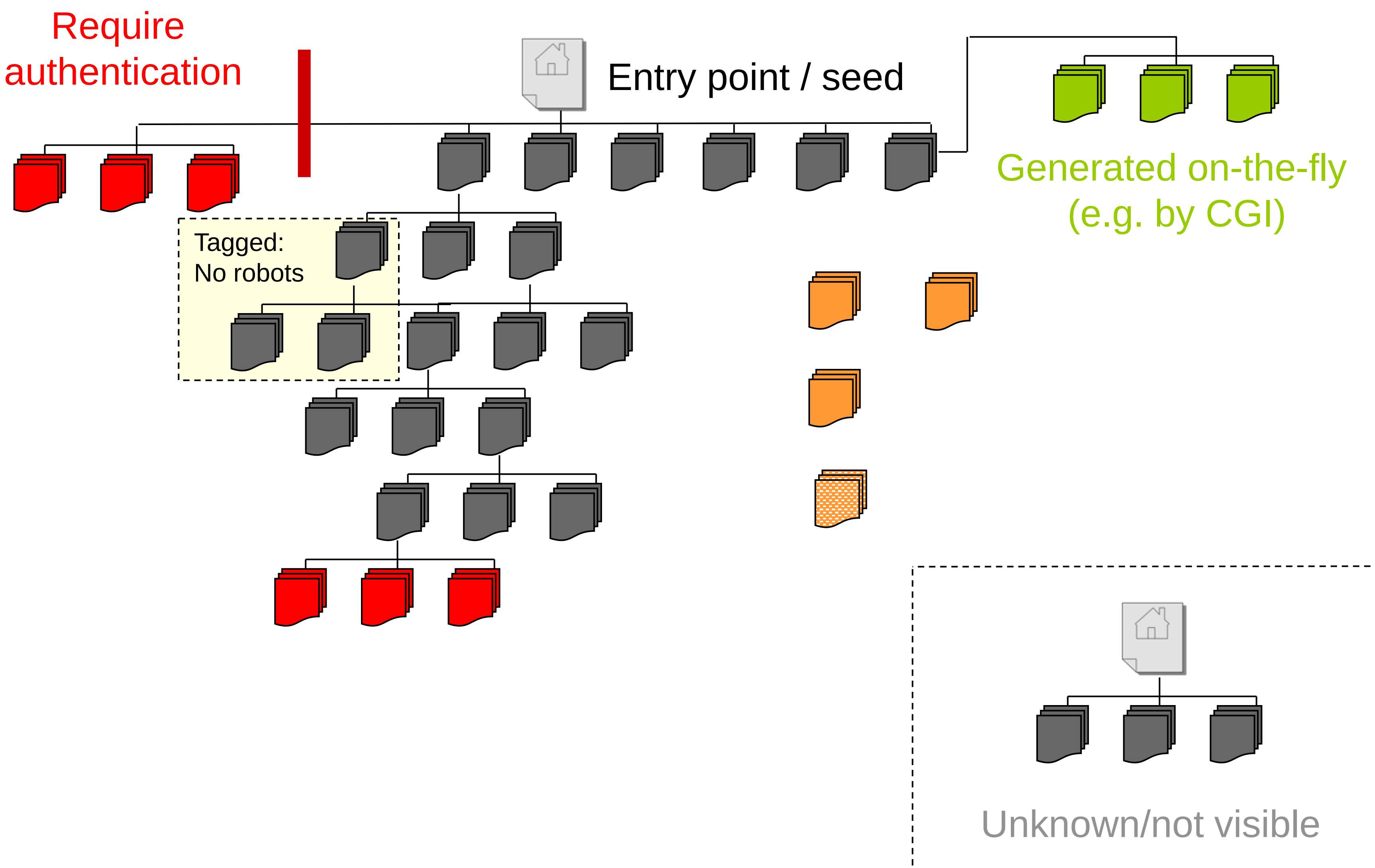
Capturing



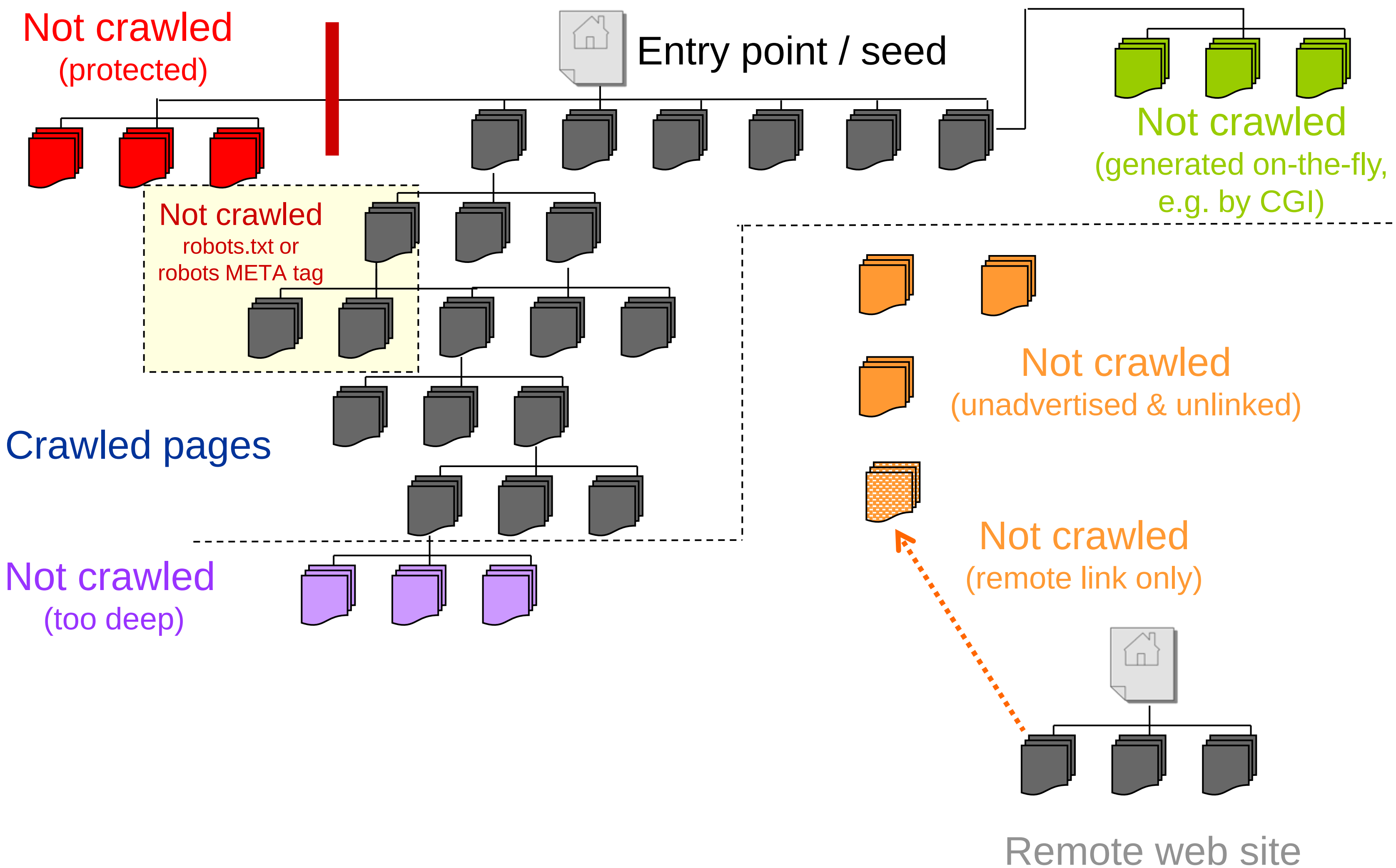
A Webmaster's Omniscient View



Web Server's View of a Web Site

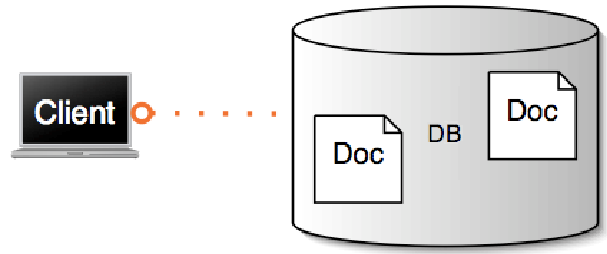


A Crawler's View of a Web Site

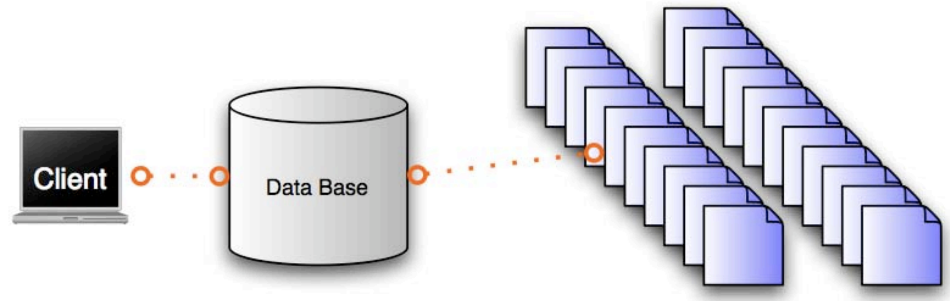


Web Information Systems

Dynamic Web sites



Hidden Web



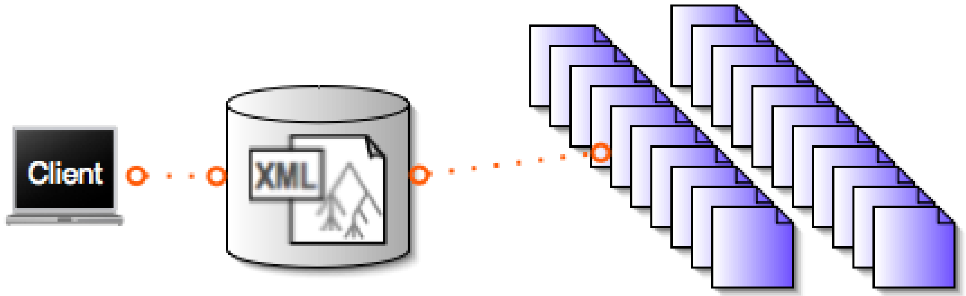
- Each interaction with a Web information system can potentially generate a unique customized response

⇒ Document the context of this interaction, or pseudo-transaction

Crawler-Server Collaboration

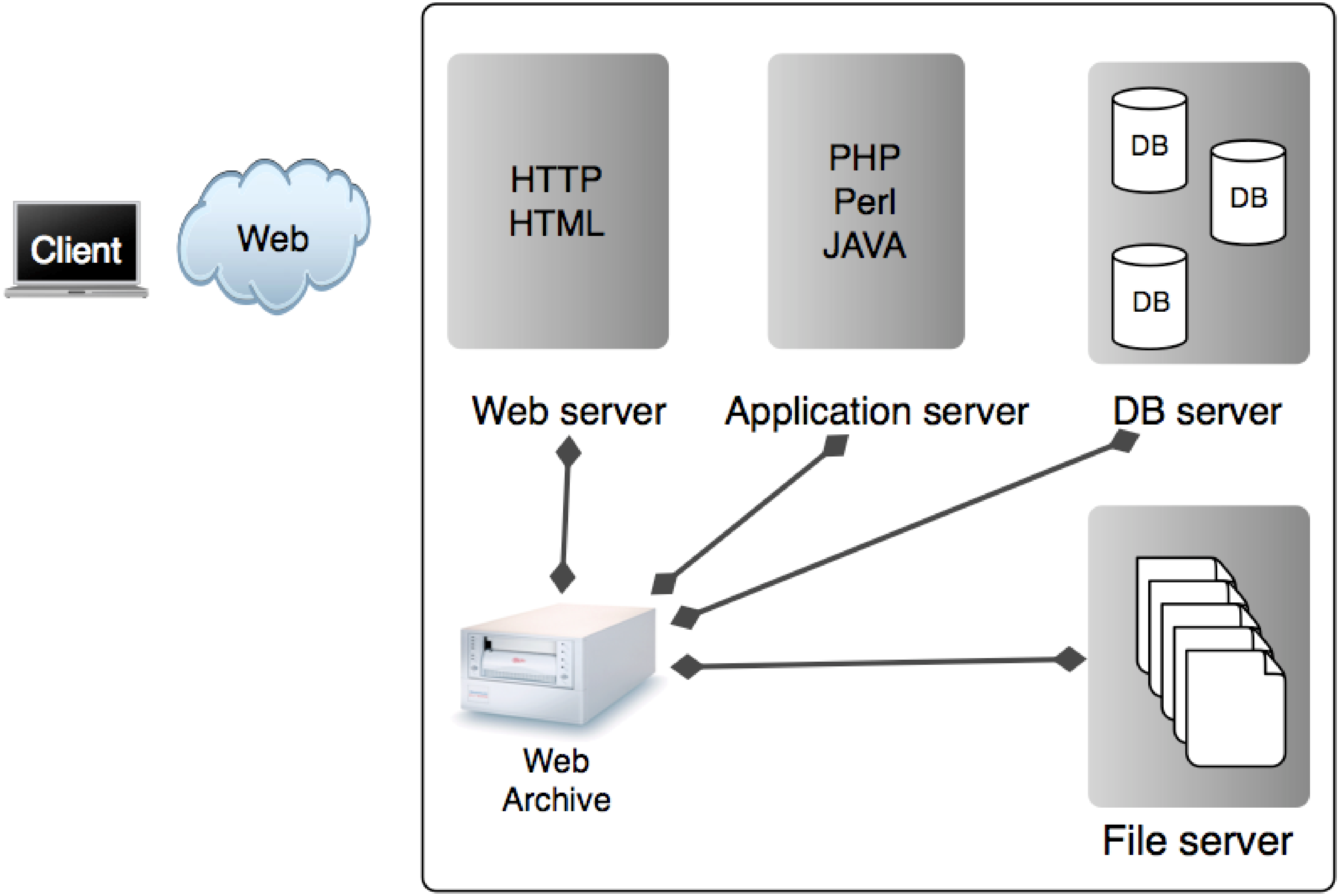
- Open Archives Initiative (OAI) Protocol for Metadata Harvesting
- Provided flat list (maybe hidden for public)
- RSS feeds
- OAI server
  - Pushed by search-engines
  - Yahoo content acquisition program, google

$\Rightarrow$ The *sitemap* standard is intended to list the resources at a site

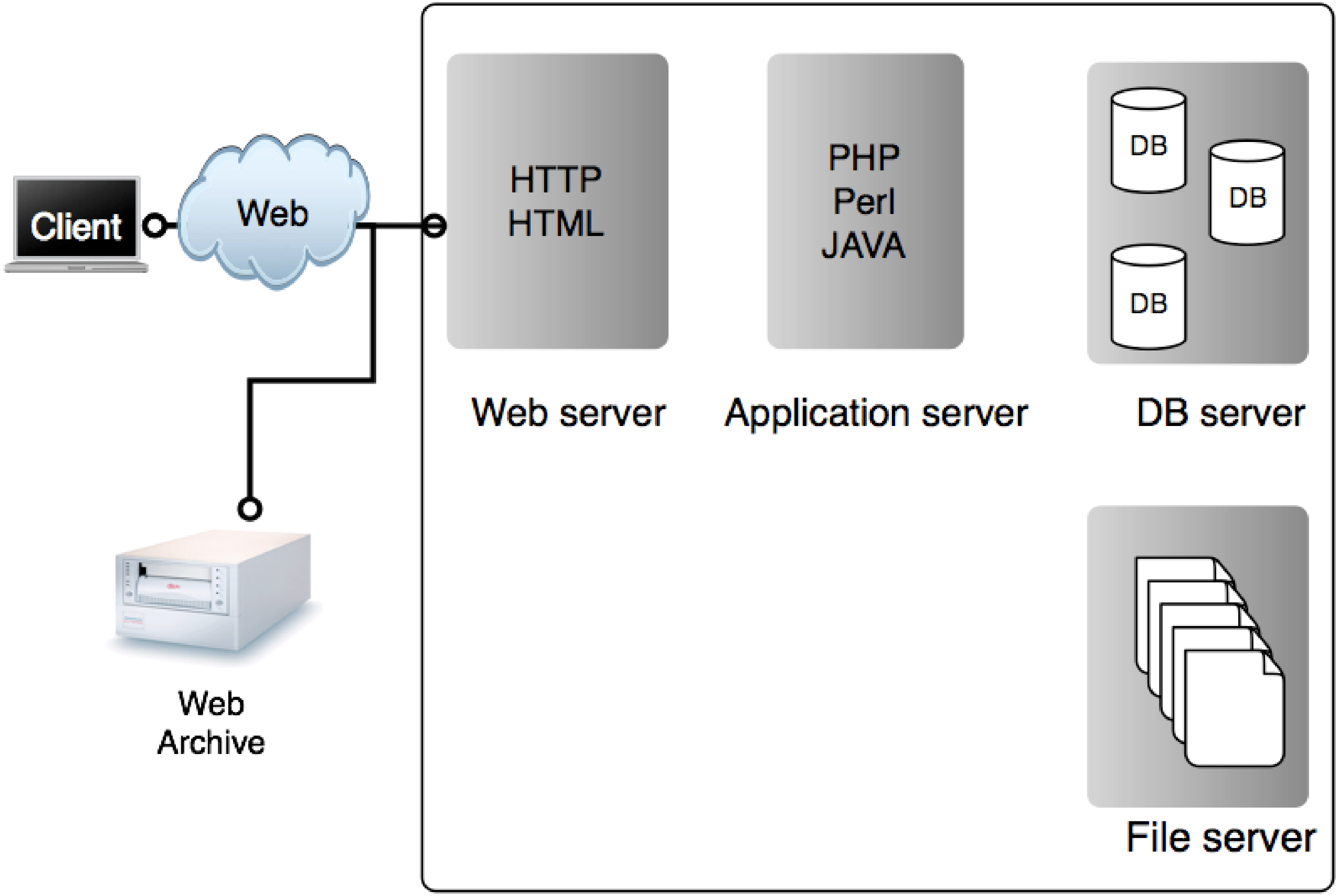


Dr. Marc Spaniol

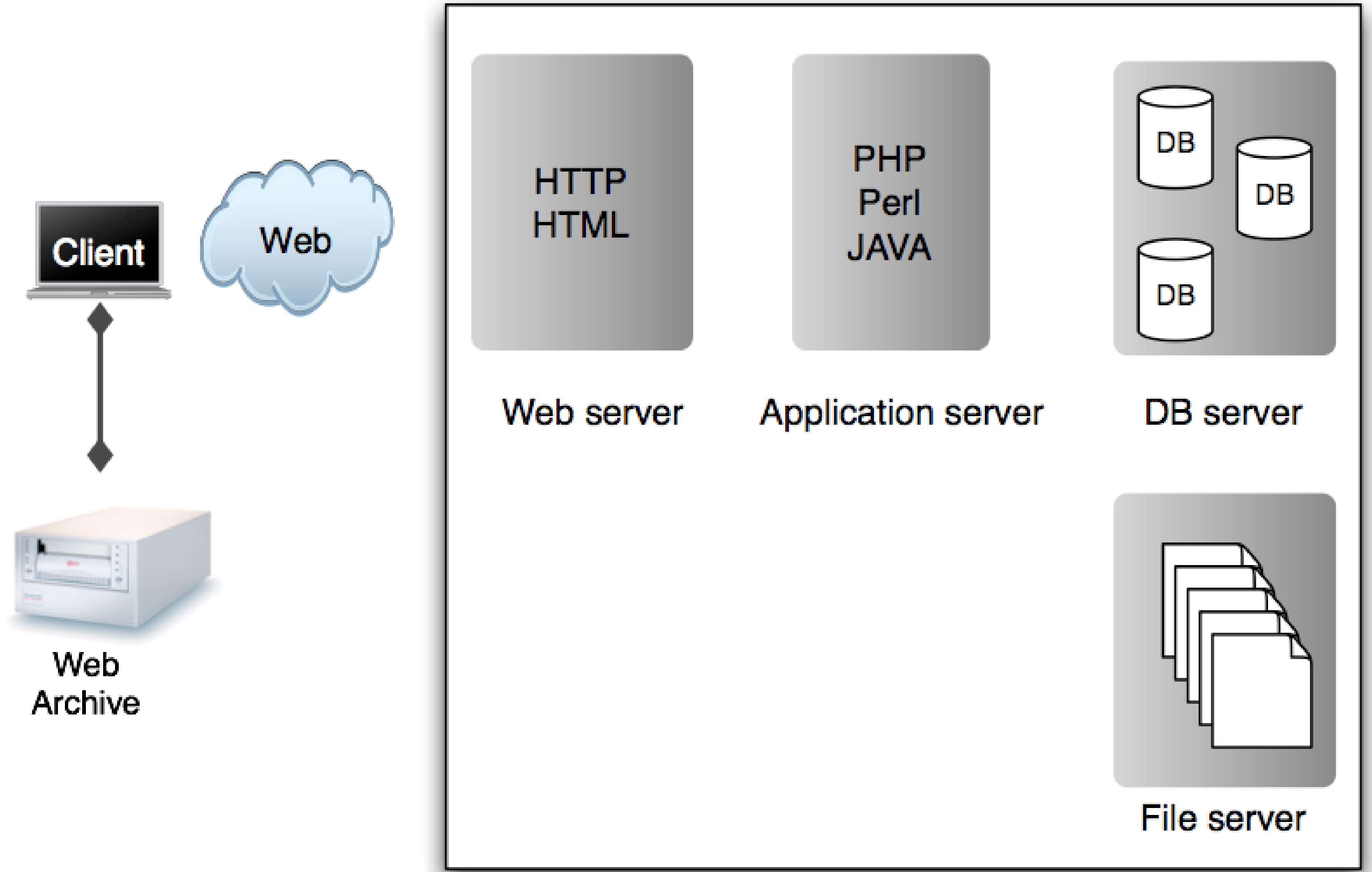
Server Side Archiving



Transaction based Archiving



Client Side Archiving



Capturing Approaches Summary

Approach	Benefits	Drawbacks
Server Side Archiving	+ Extremely comprehensive + Changes are fully traceable + Instantaneous snapshots + No network latency or limitations + Deep Web “compliant”	- Change monitoring may decrease server performance - Needs sophisticated set-up - Requires server access
Transaction based Archiving	+ Comes for “free” + “Smart” coverage achieved by human interaction + Simple maintenance + No server collaboration required	- Unsystematic (requires constant traffic) - Data quality is potentially poor - Needs traffic monitoring - Privacy issues - Potential network latency or limitations
Client Side Archiving	+ No server collaboration needed + Only crawler set-up required + Mostly automated process (daily/weekly/monthly)	- Changes might get lost - Sophisticated crawling strategy needed - Potential network latency or limitations - Computational “expensive”

Temporal Coherence

- What means coherence?
  - "The action or fact of cleaving or sticking together"
  - "Harmonious connexion of the several parts, so that the whole 'hangs together'"

Oxford English Dictionary
[http://dictionary.oed.com]

- Temporal coherence in Web archiving:
  - Capturing Web sites as "authentic" as possible
  - Ensure an "as of time point x (or interval [x, y])" capture of a Web site

⇒ Periodic domain scope crawls of Web sites to obtain a best possible representation with respect to a time point / interval

Assumptions and Notations

- Basic Assumptions
  - Web site to be crawled consists of n Web pages
  - Changes of Web pages occur per time unit and independent of each other
  - Change rates are assumed / given
  - Delay between downloads of pages is the same
  - Download time is neglected

- Basic Notation
  - Crawl: c
  - Web pages: $p_1, \ldots, p_n$
  - Change probability of page p_i: λ_i
  - Time of downloading page p_i: $t(p_i)$
  - Last modified value of page p_i: μ_i
  - Content hash or etag of page p_i: $\theta(p_i)$
  - Crawl interval: $[t_s, t_e]$

Living Web Archives

Web Archiving

Dr. Marc Spaniol

Coherence

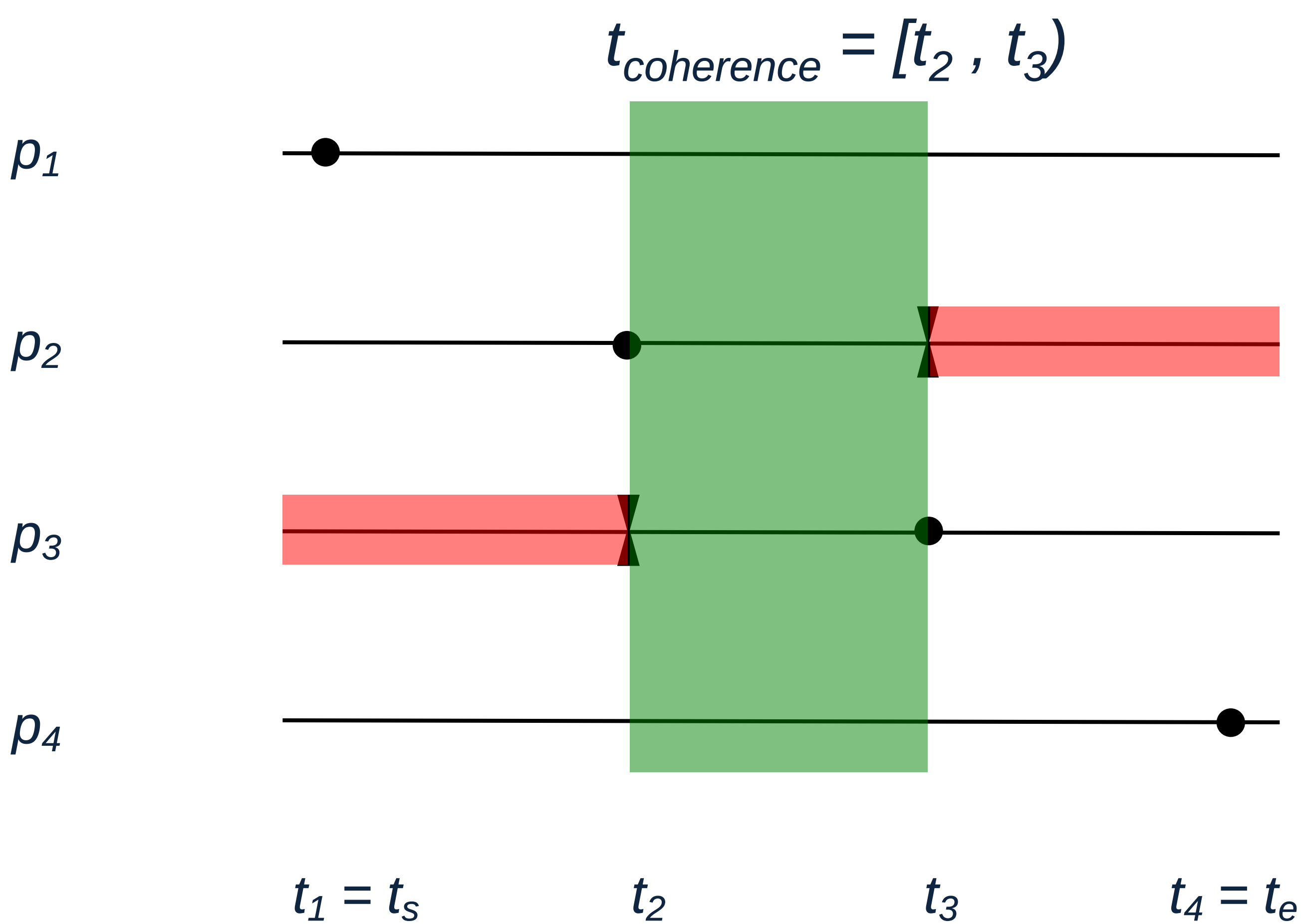
Definition:

1. A single Web page is always coherent.
2. The invariance interval $[\mu_i,\mu_i^*]$ of page p_i lies between the last modified time stamp μ_i at time $t(p_i)$ of downloading p_i ($\mu_i \leq t(p_i)$) and the next change μ_i^* following $t(p_i)$.
3. Two or more pages are coherent if

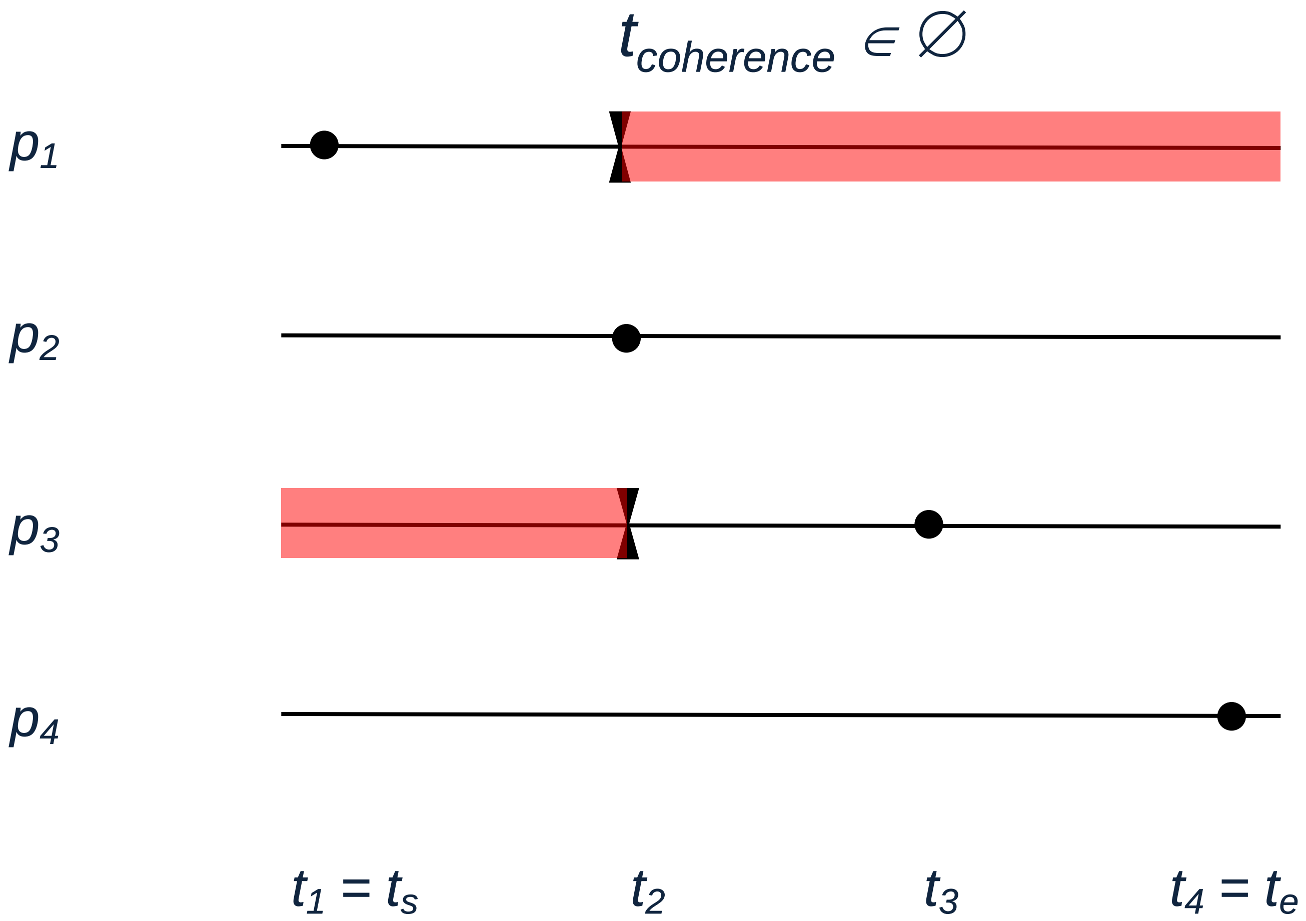
   there is a time point (or interval) $t_{coherence}$ so that a non-empty intersection among the invariance interval of all pages exists:

$$\forall p_i, \exists t_{coherence} : t_{coherence} \in \bigcap_{i=1}^{n} [\mu_i, \mu_i^*] \neq \emptyset$$

Coherence by Example



Coherence by Example



Observable Coherence

Definition:

Two or more pages are observable coherent if

there is a single timepoint (or interval) $t_{coherence}$ so that there is a non-empty intersection of the intervals spanning the respective download time $t(p_i)$ and the corresponding last modified stamp μ_i retrieved at time of download ($\mu_i \leq t(p_i)$):

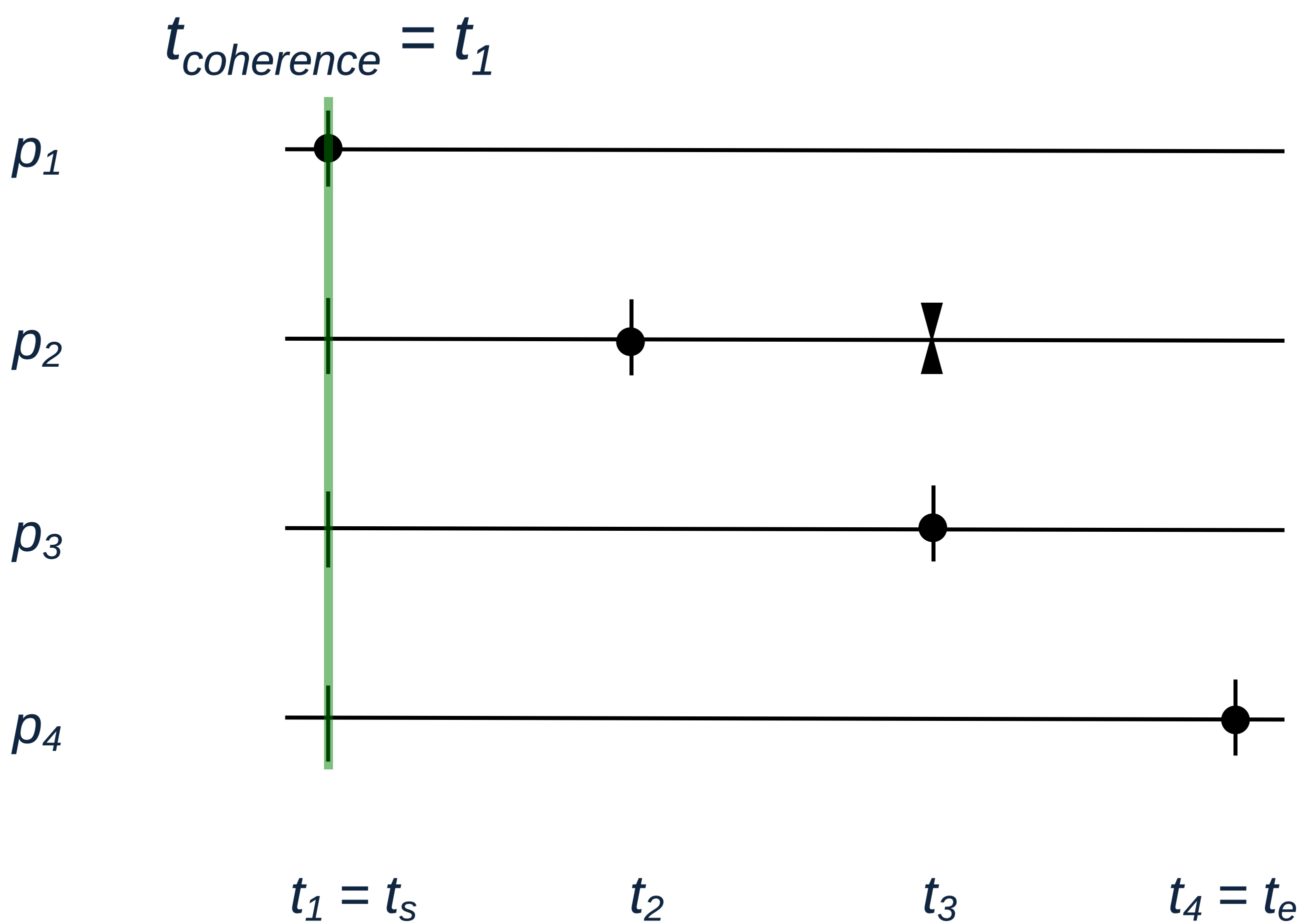
$$\forall p_i, \exists t_{coherence} : \bigcap_{i=1}^{n} [\mu_i, t(p_i)] \neq \emptyset \wedge t_{coherence} \in [\mu_i, t(p_i)]$$

Measurable Coherence

- Specialization of observable coherence
  - Makes observable coherence measurable in a real life scenario
  - Overcomes “right-hand side blindness” of crawlers
- Ability to issue a guaranteed coherence statement
  - Valid for all contents of a Web site
  - “Regardless” of crawl duration
- Suitable coherence time point (or interval) $t_{coherence}$ needed

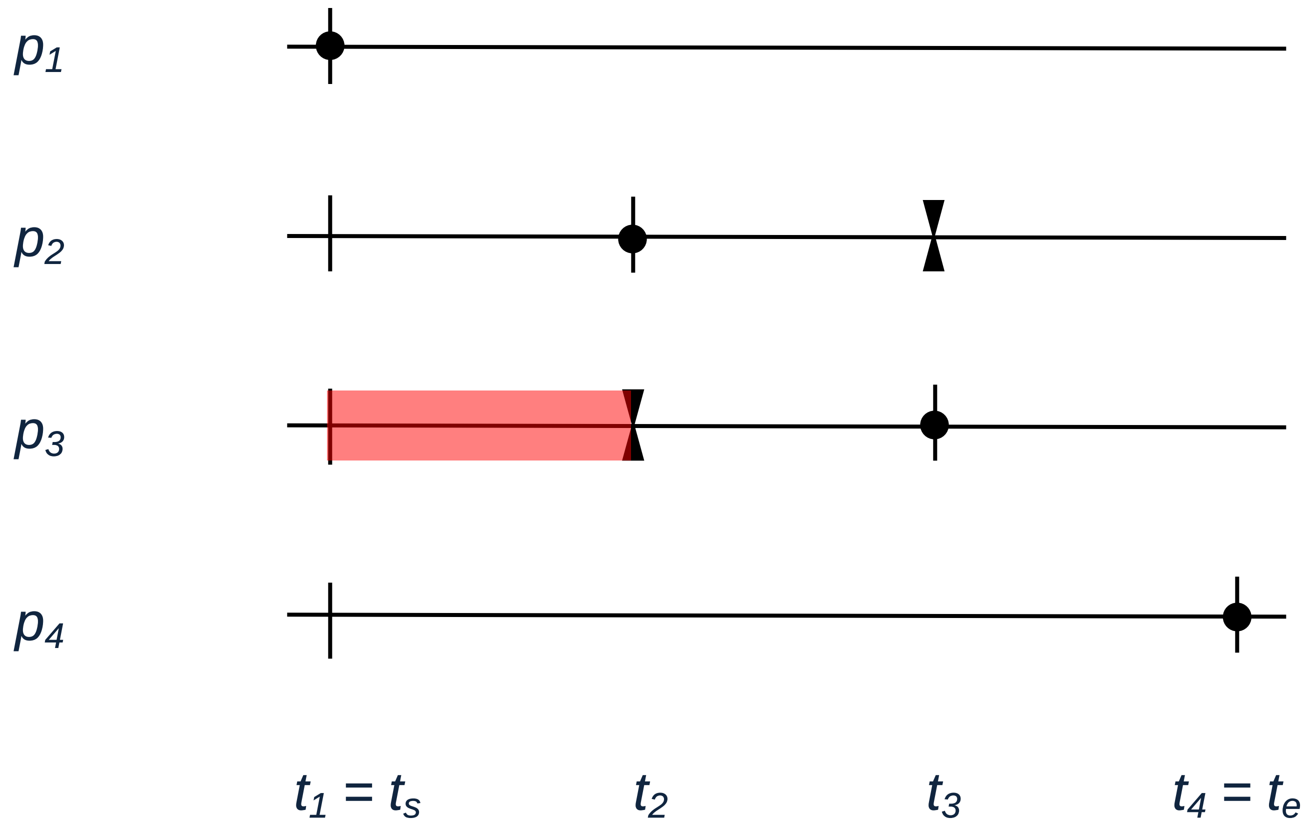
$\Rightarrow$ Full control is only given for $t_{coherence} = t_s$

Measurable Coherence by Example



Measurable Coherence by Example

$t_{coherence} \in \varnothing$



Living Web Archives

Web Archiving

Dr. Marc Spaniol

Quantifying Measurable Coherence

Error function

$$f(p_i) = \begin{cases} 0 & \text{, if } \mu_i \le t_s \\ 1 & \text{, else} \end{cases}$$

Coherence function

$$C(c) = 1 - \frac{\sum_{i=1}^{n} f(p_i)}{n} \quad , n \ge 1$$

Databases and Information Systems
Prof. Dr. G. Weikum

Inducible Coherence

Definition:

Two or more pages are inducible coherent if

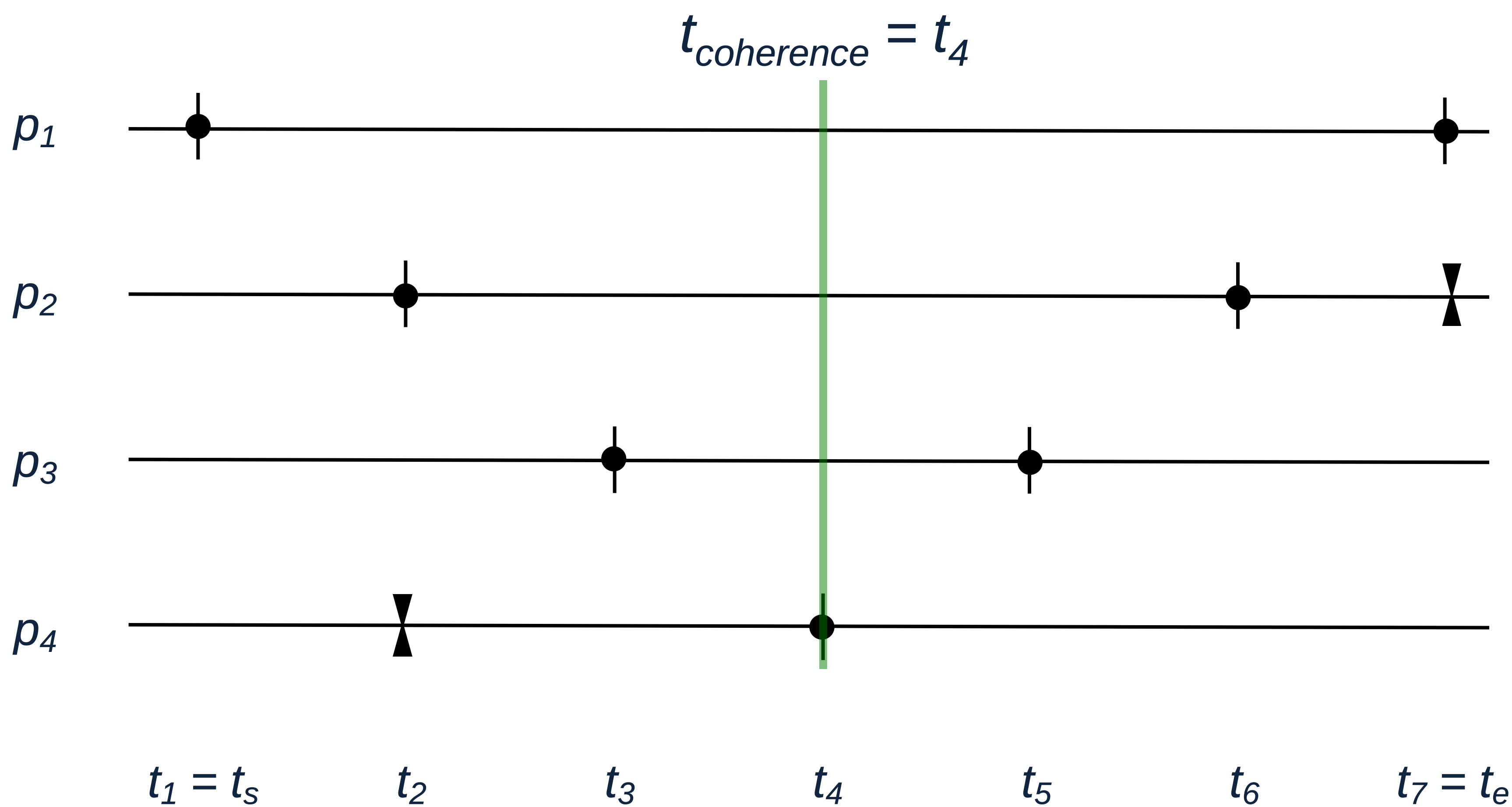
there is a time point $t_{coherence}$ between the visit of pages $t(p_i)$ and the subsequent revisit $t(\tilde{p}_i)$ where the etag or content hash θ of corresponding pages ($\theta(m)$ having $m \in \{p_i, \tilde{p}_i\}$) has not changed:

$$\forall p_i, \exists t_{coherence} : \theta(p_i) = \theta(\tilde{p}_i) \wedge t_{coherence} \in \bigcap_{i=1}^{n} [t(p_i), t(\tilde{p}_i)]$$

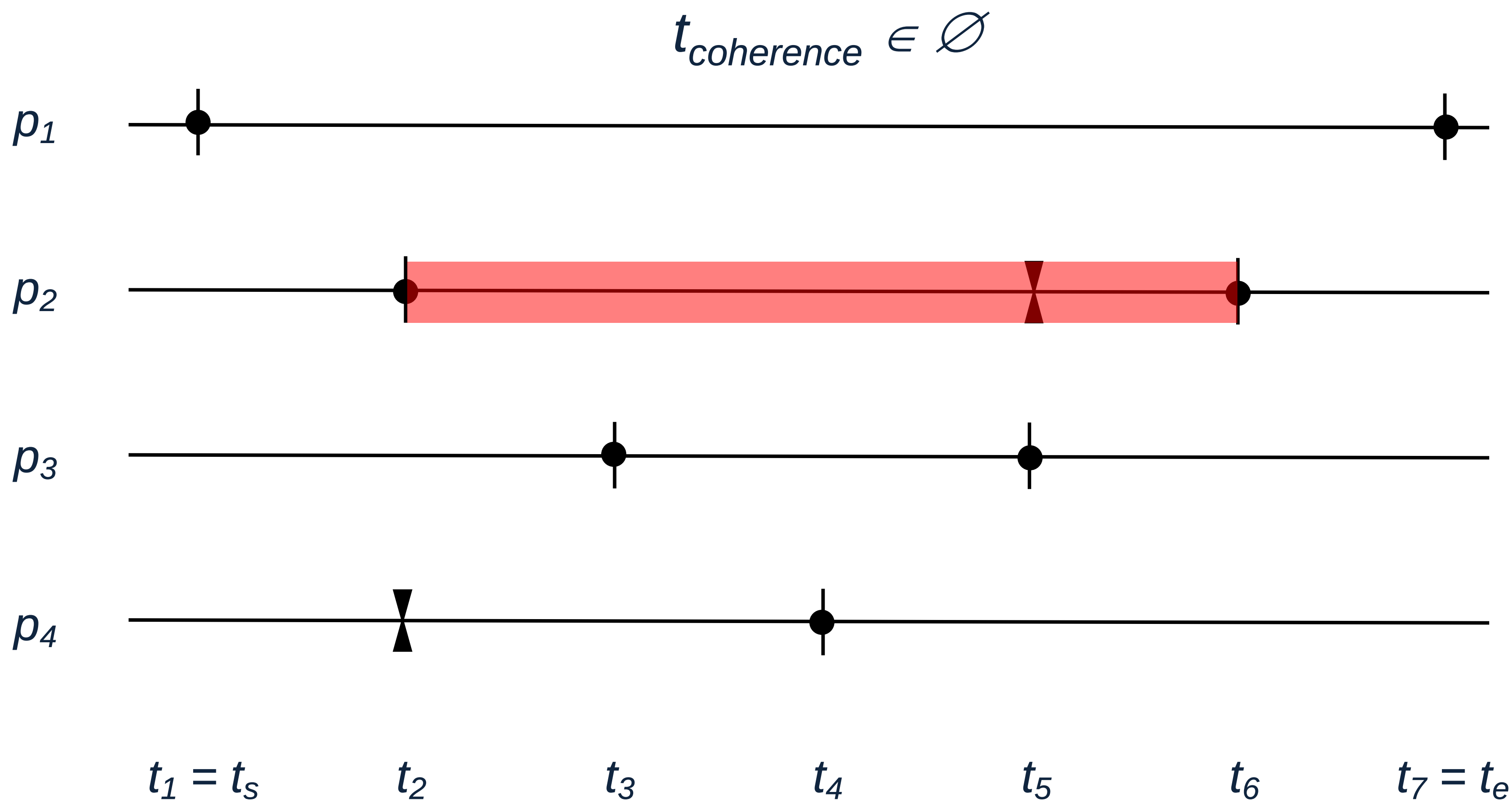
Web Archiving

Dr. Marc Spaniol

Inducible Coherence by Example



Inducible Coherence by Example



Quantifying Inducible Coherence

Error function

$$f(p_i, \tilde{p}_i) = \begin{cases} 0 & \text{, if } \theta(p_i) = \theta(\tilde{p}_i) \\ 1 & \text{, else} \end{cases}$$

Coherence function

$$C(c) = 1 - \frac{\sum_{i=1}^{n} f(p_i, \tilde{p}_i)}{n} \quad , n \geq 1$$

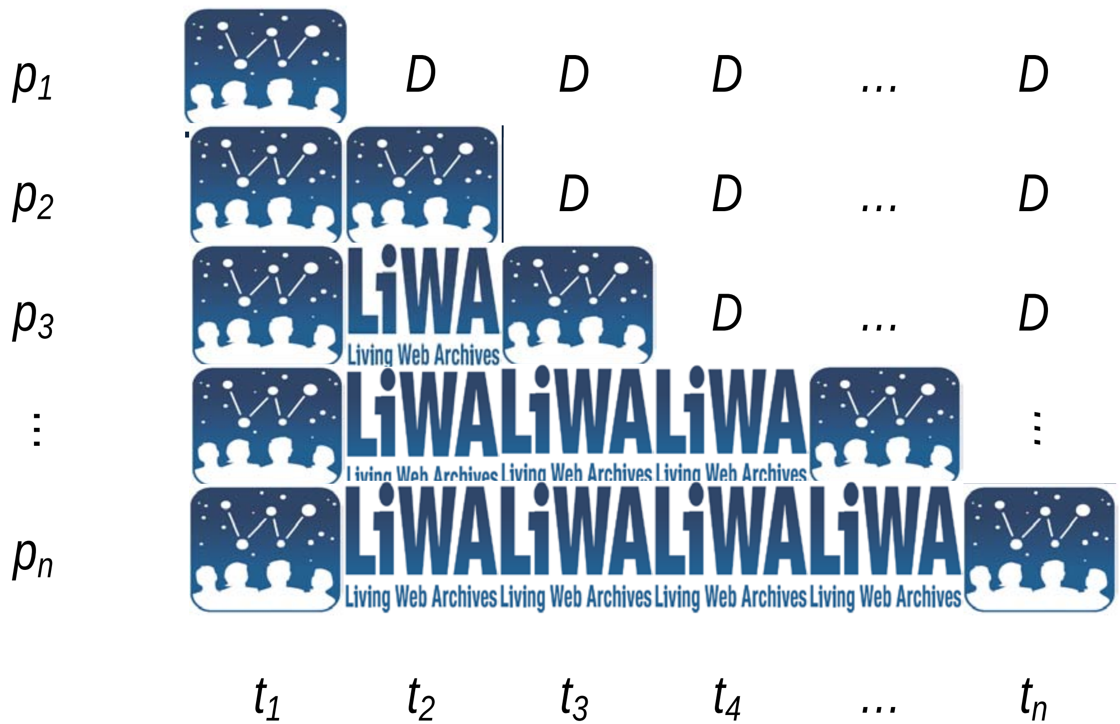
Dr. Marc Spaniol

Coherence Approaches Summary

Approach	Definition	Implementation
Coherence	Requires universal knowledge	- Impractical - Crawlers are unaware of the future (“forward-blind”)
Observable Coherence	Invariance intervals become traceable	- Impractical without suitable reference time-point - Relies on accuracy of last modified stamps
Measurable Coherence	Makes observable coherence become quantifiable relative to start of crawl	+ Ad-hoc verifiable + Efficient + Produces no extra traffic - Relies on accuracy of last modified stamps
Inducible Coherence	Makes coherence of improper dated contents become quantifiable relative to end of crawl / start of revisit	+ Full control on proper dating of contents - Produces extra traffic - More complex + Few “full” downloads + Mostly conditional gets - Politeness delays are the “real” bottleneck

Crawl Improvement: Measurable Coherence

- Conflict probability: $\kappa(p_i) = 1 - (1 - \lambda_i)^{t(p_i) - t_s}$
- Crawling so that conflicts are "tolerable": $\kappa(p_i) < \eta$
- "Slots" to be assigned range from length 0 to t_{n-1}



Improved Measurable Coherence Crawl Scheduling

input: $p_1,\ldots,p_n$ - list of pages in descending order of λ_i,
η - readiness to assume risk threshold

begin
Start with: $slot = 1$
while $slot \leq n$
do
if $\kappa(p_{slot}) < \eta$ **then** **/* no conflict expected */**
Download page p_{slot}
end
Continue with next iteration: *slot* ++
end
Download skipped pages in reversed order of their index
end

Measurable Coherence Scheduling: pos_1

Position: pos_1
Remaining λ_i : $\lambda_{n-1} >> \lambda_{n-2} >> \lambda_{n-2} >> \lambda_1 . > \lambda_1$

Test: $1 - (1 - \lambda_n)^0 < \eta$?
$\Rightarrow$ *Yes!*

Downloaded: λ_n

Skipped:

	t_1	t_2	t_3	t_4	…	t_n
pos_1	D	D	D	D	…	D
pos_2	D	1	D	D	…	D
pos_3	D	1	2	D	…	D
⋮	⋮	⋮	⋮	⋮		⋮
pos_n	D	1	2	3	…	n-1

LiWA Living Web Archives
Web Archiving
Dr. Marc Spaniol
Databases and Information Systems
Prof. Dr. G. Weikum

Measurable Coherence Scheduling: pos_2

Position: pos_2

Remaining λ_i : $\lambda_{n-3} > \ldots \lambda_{n-2} > \lambda_1 \ldots > \lambda_1$

Test: $1 - (1 - \lambda_{n-3})^1 < \eta$?

e.g. *Yes!*

Downloaded: λ_n, λ_{n-3}

Skipped: λ_{n-1} λ_{n-2}

pos_1	D	D	D	D	…	D
pos_2	D	1	D	D	…	D
pos_3	D	1	2	D	…	D
⋮	⋮	⋮	⋮	⋮		⋮
pos_n	D	1	2	3	…	$n-1$
	t_1	t_2	t_3	t_4	…	t_n

Living Web Archives

Web Archiving

Dr. Marc Spaniol

Measurable Coherence Scheduling: pos_3

Position: pos_3

Remaining $\lambda_i : \lambda_{n-6} > \ldots > \lambda_1$

Test: $1 - (1 - \lambda_{n-5})^2 < \eta$?

e.g. Yes!

Downloaded: $\lambda_n, \lambda_{n-3}\ \lambda_{n-5}$

Skipped: $\lambda_{n-1}, \lambda_{n-2}\ \lambda_{n-4}$

	t_1	t_2	t_3	t_4	…	t_n
pos_1	D	D	D	D	…	D
pos_2	D	1	D	D	…	D
pos_3	D	1	2	D	…	D
⋮	⋮	⋮	⋮	⋮		⋮
pos_n	D	1	2	3	…	n-1

Measurable Coherence Scheduling: pos_k

Position: pos_k
Remaining λ_i : λ_2 / λ_1 > λ_1

Test: $1 - (1 - \lambda_2)^{k-1} < \eta$?
e.g. Yes! / No!

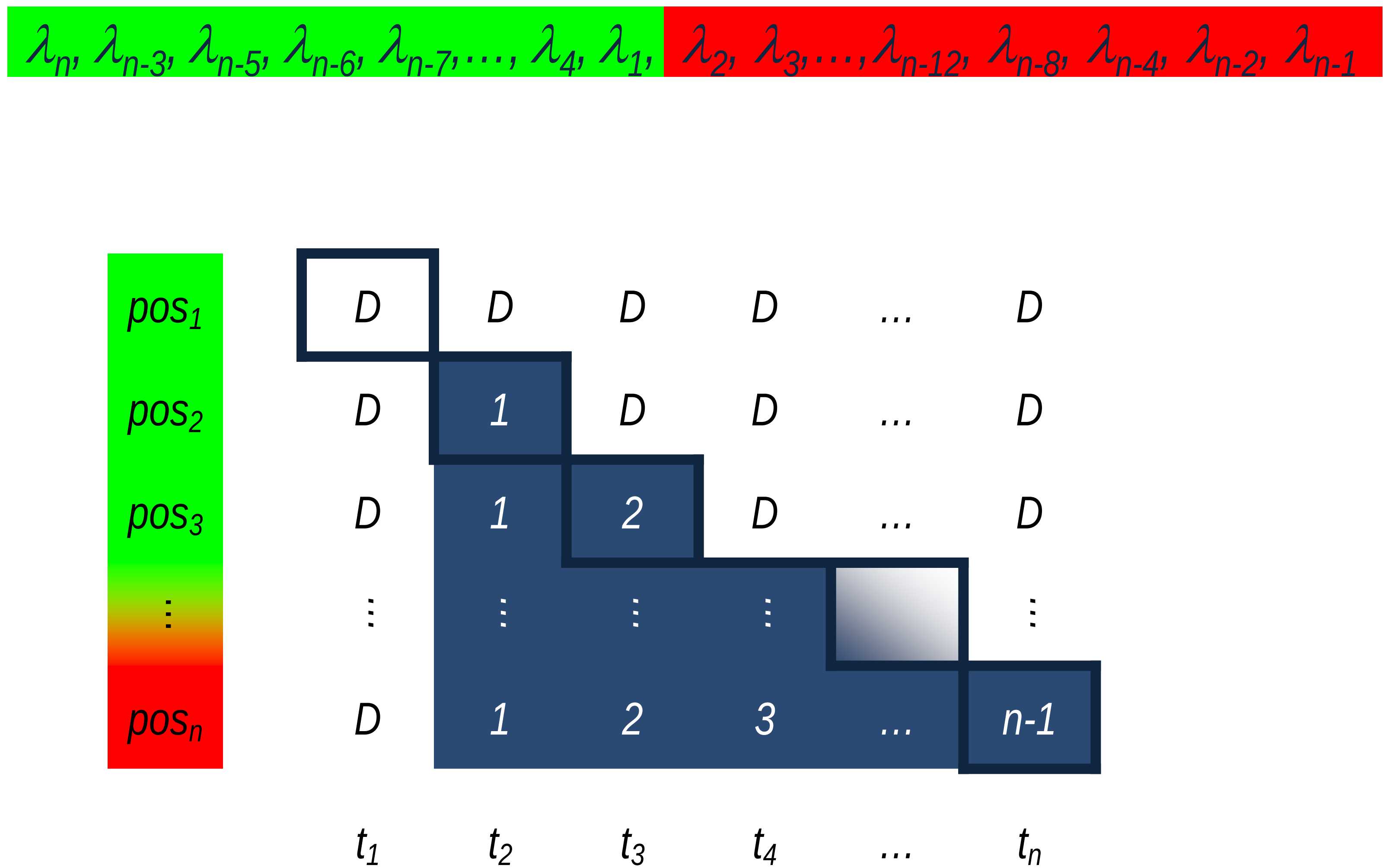
Downloaded: $\lambda_n, \lambda_{n-3}, \lambda_{n-5}, \lambda_{n-6}, \lambda_{n-7}, \ldots, \lambda_4\ \lambda_1$

Skipped: $\lambda_{n-1}, \lambda_{n-2}, \lambda_{n-4}, \lambda_{n-8}, \lambda_{n-12}, \ldots, \lambda_3\ \lambda_2$

	t_1	t_2	t_3	t_4	…	t_n
pos_1	D	D	D	D	…	D
pos_2	D	1	D	D	…	D
pos_3	D	1	2	D	…	D
⋮	⋮	⋮	⋮	⋮		⋮
pos_n	D	1	2	3	…	n-1

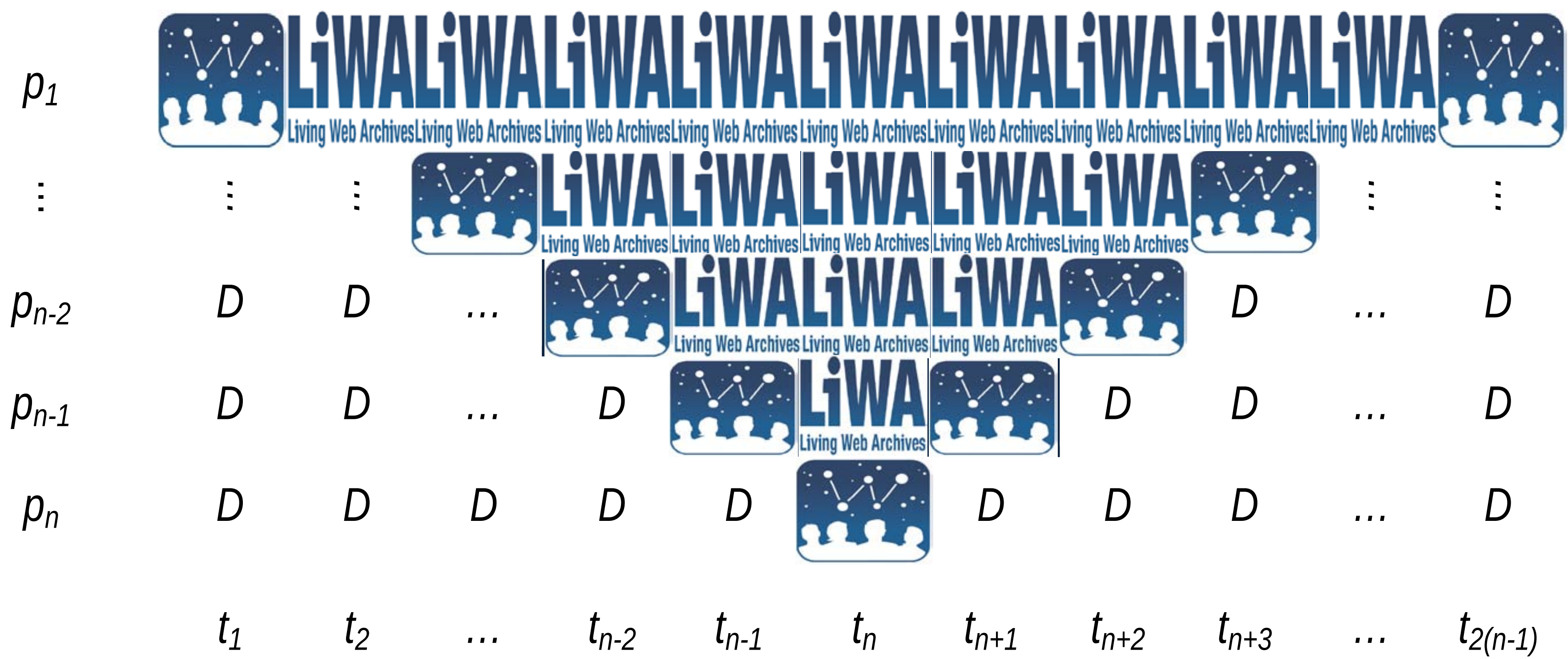
Dr. Marc Spaniol

Final Crawl Sequence



Crawl Improvement: Inducible Coherence

- Conflict probability: $\kappa(p_i) = 1 - (1 - \lambda_i)^{t(\tilde{p}_i) - t(p_i)}$
- Crawling so that conflicts are “tolerable”: $\kappa(p_i) < \eta$
- “Slots” to be assigned range from length 0 to $t_{2(n-1)}$



Improved Inducible Coherence Crawl Scheduling

input: $p_1,\ldots,p_n$ - list of pages in descending order of λ_i,
η - readiness to assume risk threshold

begin
  Start with: *slot* = *1*, *lastpromising* = *n*
  while $slot \leq lastpromising$
  do
    if $\kappa(p_{slot}) \geq \eta$ **then** **/* conflict expected! */**
    Move p_{slot} to position *lastpromising*
    Decrease promising boundary: *lastpromising* −−
end
else
    Increase promising boundary: *promising* ++
end
  end
  $slot = n$ **while** $slot \geq 1$
  do **/* visit from hopeless to promising */**
Download page p_{slot}
Decrease slot counter: *slot* −−
  end
  $slot = 2$ **while** $slot \leq n$
  do **/* revisit from promising to hopeless */**
Revisit page p_{slot}
Increase slot counter: *slot* ++
  end
end

Inducible Coherence Scheduling: pos_n

Position: pos_n
Remaining λ_i : $\lambda_{n-1} > \lambda_{n-2} > \ldots > \lambda_1$

Test: $1 - (1 - \lambda_n)^0 < \eta$?
$\Rightarrow$ *Yes!*

Promising: λ_n

Hopeless:

pos_1	D	1	…	n-i-2	n-i-1	n-i	n-i+1	n-i+2	n-i+3	…	2(n-1)
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮	⋮
pos_{n-2}	D	D	…	D	1	2	3	4	D	…	D
pos_{n-1}	D	D	…	D	D	1	2	D	D	…	D
pos_n	D	D	D	D	D	D	D	D	D	…	D
	t_1	t_2	…	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	…	t_{2n-1}

LiWA Living Web Archives

Web Archiving

Dr. Marc Spaniol

Inducible Coherence Scheduling: pos_{n-1}

Position: pos_{n-1}
Remaining λ_i : $\lambda_{n-3} > \lambda_{n-2} > \ldots > \lambda_1$

Test: $1 - (1 - \lambda_{n-3})^2 < \eta$?
e.g. Yes!

Promising: λ_{n-3}, λ_n

Hopeless: λ_{n-1} λ_{n-2}

pos_1	D	1	…	n-i-2	n-i-1	n-i	n-i+1	n-i+2	n-i+3	…	2(n-1)
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮	⋮
pos_{n-2}	D	D	…	D	1	2	3	4	D	…	D
pos_{n-1}	D	D	…	D	D	1	2	D	D	…	D
pos_n	D	D	D	D	D	D	D	D	D	…	D
	t_1	t_2	…	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	…	t_{2n-1}

Inducible Coherence Scheduling: pos_{n-2}

Position: pos_{n-2}
Remaining λ_i : $\lambda_{n-5} > \ldots > \lambda_1$

Test: $1 - (1 - \lambda_{n-5})^4 < \eta$?
e.g. No! Yes!

Promising: $\lambda_{n-3}, \lambda_n, \lambda_n$

Hopeless: $\lambda_{n-1}, \lambda_{n-2}, \lambda_{n-4}$

pos_1	D	1	…	n-i-2	n-i-1	n-i	n-i+1	n-i+2	n-i+3	…	2(n-1)
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮	⋮
pos_{n-2}	D	D	…	D	1	2	3	4	D	…	D
pos_{n-1}	D	D	…	D	D	1	2	D	D	…	D
pos_n	D	D	D	D	D	D	D	D	D	…	D
	t_1	t_2	…	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	…	t_{2n-1}

Inducible Coherence Scheduling: pos_{n-k}

Position: pos_{n-k}

Remaining λ_i : $\lambda_2 > \lambda_1$

Test: $1 - (1 - \lambda_2)^{2(n-(n-k))} < \eta$?

e.g. Yes! / No

Promising: $\lambda_4, \ldots, \lambda_{n-7}, \lambda_{n-6}, \lambda_{n-5}, \lambda_{n-3}, \lambda_n\ \lambda_n$

Hopeless: $\lambda_{n-1}, \lambda_{n-2}, \lambda_{n-4}, \lambda_{n-8}, \lambda_{n-12}, \ldots, \lambda_3\ \lambda_2$

	t_1	t_2	…	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	…	t_{2n-1}
pos_1	D	1	…	n-i-2	n-i-1	n-i	n-i+1	n-i+2	n-i+3	…	2(n-1)
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮	⋮
pos_{n-2}	D	D	…	D	1	2	3	4	D	…	D
pos_{n-1}	D	D	…	D	D	1	2	D	D	…	D
pos_n	D	D	D	D	D	D	D	D	D	…	D

Final Crawl Sequence

Visit: $\lambda_{n-1}, \lambda_{n-2}, \lambda_{n-4}, \lambda_{n-8}, \lambda_{n-12}, \ldots, \lambda_3, \lambda_2, \lambda_1, \lambda_4, \ldots, \lambda_{n-7}, \lambda_{n-6}, \lambda_{n-5}, \lambda_{n-3}, \lambda_n$

Revisit: $\lambda_{n-3}, \lambda_{n-5}, \lambda_{n-6}, \lambda_{n-7}, \ldots, \lambda_4, \lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{n-12}, \lambda_{n-6}, \lambda_{n-5}, \lambda_{n-3}, \lambda_{n-1}$

pos_1	D	1	…	n-i-2	n-i-1	n-i	n-i+1	n-i+2	n-i+3	…	2(n-1)
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮		⋮	⋮
pos_{n-2}	D	D	…	D	1	2	3	4	D	…	D
pos_{n-1}	D	D	…	D	D	1	2	D	D	…	D
pos_n	D	D	D	D	D	D	D	D	D	…	D
	t_1	t_2	…	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	…	t_{2n-1}

LiWA Living Web Archives

Web Archiving

Dr. Marc Spaniol

Databases and Information Systems
Prof. Dr. G. Weikum

Incoherence Detection

- Multistage change measurement procedure
  1) *Conditional GET (etag comparison)*
  2) *Check content timestamp (last modified comparison)*
  3) *Compare a hash of the page with a stored hash*
  4) *Non-significant differences (ads, fortunes, request timestamp)*
     - *only hash text content, or "useful" text content*
     - *compare distribution of n-grams (shingling)*
     - *compute edit distance with previous version*

Incoherence Categories

- Removed contents

- Structural (changed link structure)
  - Links to new contents added
  - Links to removed contents deleted

- Content wise
  - Major changes: Text added or deleted (in “large” sections)
  - Minor changes: Text changed (in “small” sections)

⇒ Comparison of document similarities (syntactically not semantically)

Document Similarity

- In general:
  - Given a body of documents, e.g., the Web
  - Find pairs of docs that have a lot of text in common

  ⇒ Identify mirror sites, approximate mirrors, plagiarism, quotation of one document in another, “good” document with random spam, etc.

- In the case of data quality in Web archiving:
  - Characterize change (diff) between two versions of page
  - Identify relevant aspects of changes to web pages and sites
    - Content: full, − banners, − links, − photos, − style, etc.
    - Links: all, non-navigational, intra-site, etc.
  - Quantify the amount of changes

  ⇒ Filtering of irrelevant changes

Shingles – k-grams

- Representation of a document by its set of shingles (or k-grams)
  - Documents that have lots of shingles in common have similar text
  - The text may even appear in different order

$\Rightarrow$ Similar documents are very likely to have many shingles in common

- Selection of k having a "useful" size is crucial:
  - If k is too small, documents might have too many shingles in common
  - If k is too large, compression is not good

- Heuristic experience:
  - k = 5 is OK for short documents
  - k = 10 is better for long documents

Basic Data Model: Sets

- Many similarity problems can be expressed as finding subsets of some universal set that have significant intersection
- A k-shingle (or k-gram) of a document is a sequence of k characters that appears in the document
- Documents are represented by their k-shingles
  - All possible k-shingles represent universe *U*
  - Degree of "overlap" between shingle sets represents similarity of documents

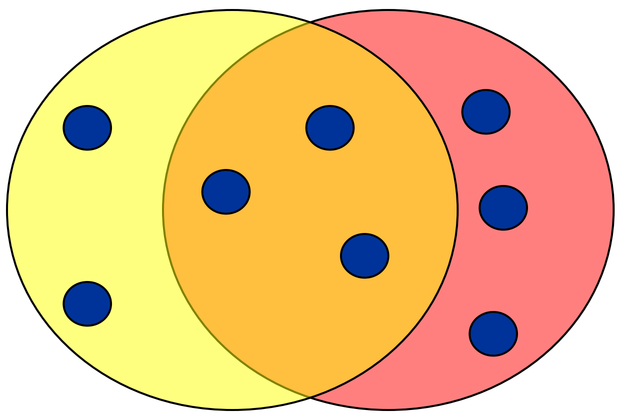
- Example:
  - Document = abcab
  - $k = 2$
  - Set of 2-shingles = {ab, bc, ca}.
- Option: Regard shingles as a bag → count "ab" twice

Jaccard Similarity of Sets

- The Jaccard similarity of two sets C_1 and C_2:

  Size of their intersection divided by the size of their union:

  $$Sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



3 elements in intersection
8 elements in union

$\Rightarrow$ Jaccard similarity = 3/8

From Sets to Boolean Matrices

- Matrix representation of data in the form of subsets of a universal set
  - Rows = elements of the universal set (shingles)
  - Columns = sets (documents)
  - 1 in row *r*, column *c* iff document *c* contains shingle *r*
  - Column similarity is the Jaccard similarity of the sets of their rows with 1

⇒ Typically the matrix is sparse

- Implementation
  - Might not really represent the data by a boolean matrix
  - List of places where there is a non-zero value.

⇒ Matrix illustration is conceptually useful

Row Types

- Given columns C_1 and C_2, rows may be classified as:

C_1	C_2	
1	1	← a
1	0	← b
0	1	← c
0	0	← d

- *a* = # rows of type *a* , etc.
- Note: $Sim(C_1, C_2) = a / (a + b + c)$

Example

C_1	C_2		
0	1	←	
1	0	←	
1	1	←	✓
0	0		
1	1	←	✓
0	1	←	

$Sim(C_1, C_2) = 2/5 = 0.4$

Problems

- Computational complexity
  - Creation of shingles
  - Comparison of columns (often pair wise)
- "Compression" of columns wanted, so that
  - Similar documents obtain related signatures
  - Dissimilar documents receive discriminative signatures
- Idea:
  - Pick *m* (*m* << *k*) rows at random
  - Let the signature of column C be the m bits of C in those rows

  ⇒ Matrix is sparse

  ⇒ Many columns will have 00. . .0 as a signature

  ⇒ "Everything" is dissimilar because their 1's are in different rows

Minhashing

- Key idea: "Hash" each column C_i to a small *signature* S_i

- Basic goals:
  - S_i is sufficiently small (e.g. to be processed in the main memory)
  - $Sim(C_1, C_2)$ corresponds to $Sim(S_1, S_2)$

- Basic idea:
  - Imagine the rows permuted randomly and equally distributed
  - Define hash function $h(C)$ to compute smallest number of the (in the permuted order) row in which column C has 1
  - Use several ($m << k$) independent hash functions to create a signature
  - Optional: Check that columns with similar signatures are really similar

Implementation

- For each column c and each hash function h_i, keep a "slot" $M(i, c)$
- $M(i, c)$ becomes the smallest value of $h_i(r)$ having 1 in column c at row r
- $h_i(r)$ gives order of rows for i th permuation

for each row r
  for each column c
    if c has 1 in row r
      for each hash function h_i **do**
        if $h_i(r)$ is a smaller value than $M(i, c)$ **then**
          $M(i, c) := h_i(r)$;

- Optimization:
  - Our case: columns = documents, rows = shingles
  - Sort matrix once so it is by row
  - Compute $h_i(r)$ only once for each row

Minhashing Example

Input matrix → Signature matrix M

h_1	h_2	h_3	C_1	C_2	C_3	C_4
1	4	3	1	0	1	0
3	2	4	1	0	0	1
7	1	7	0	1	0	1
6	3	6	0	1	0	1
2	6	1	0	1	0	1
5	7	2	1	0	1	0
4	5	5	1	0	1	0

S_1	S_2	S_3	S_4
2	1	2	1
2	1	4	1
1	2	1	2

Similarities:

	$C_1\,C_3$	$C_2\,C_4$	$C_1\,C_2$	$C_3\,C_4$
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Challenges

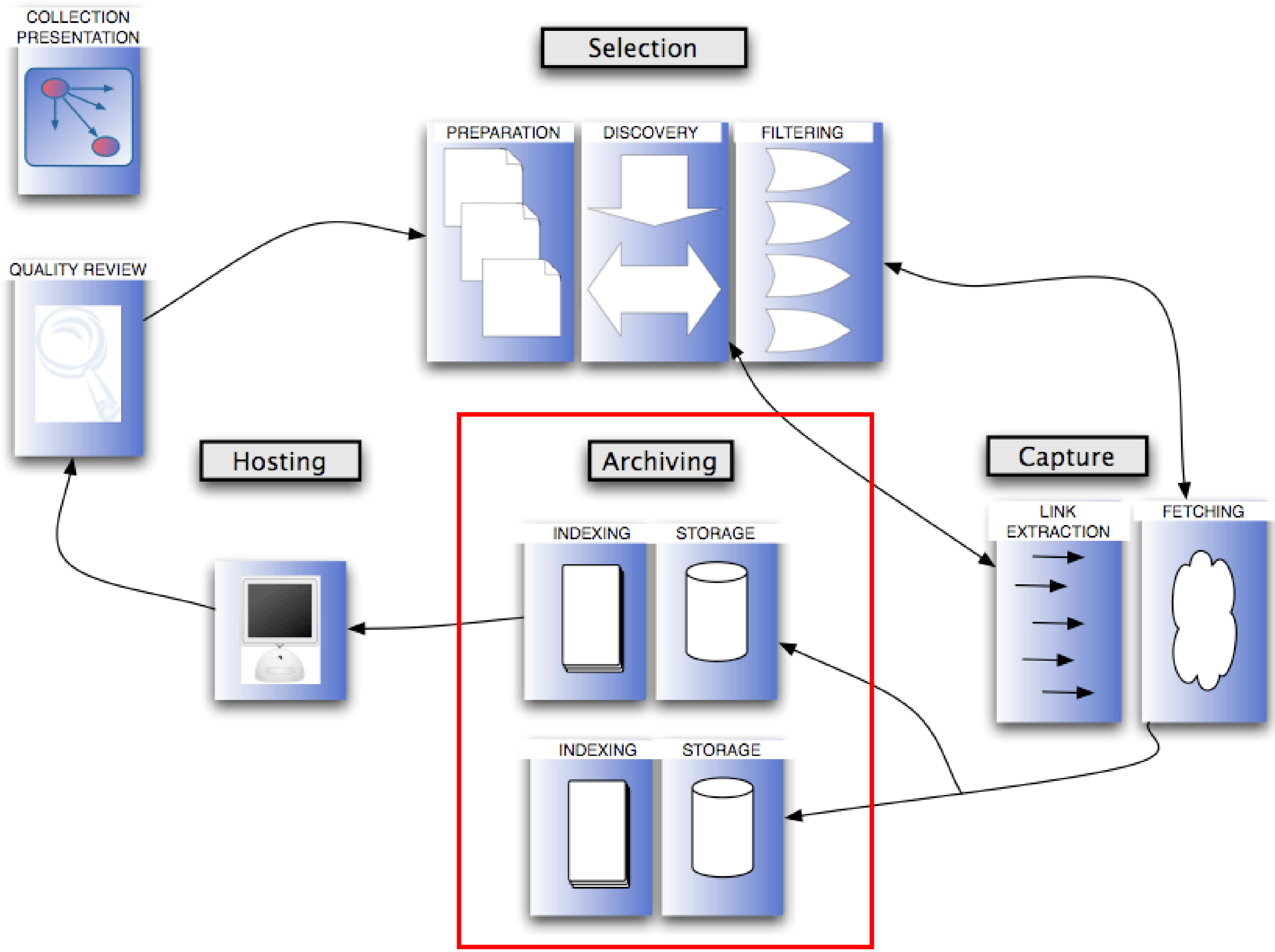
- In general:
  - Suppose huge documents (e.g., 1 billion rows)
  - Hard to pick a random permutation from 1…billion
  - Representing a random permutation requires 1 billion entries
  - Accessing rows in permuted order leads to thrashing

  $\Rightarrow$ A good approximation to permuting rows: Pick *m* << *k* hash functions
- In the case of data quality in Web archiving:
  - Document size less problematic
  - Usually only few pair wise comparisons needed
  - Random contents or automatically generated contents in CMS ruin all efforts

  $\Rightarrow$ Data scrubbing is crucial

$\Rightarrow$ Defining "good" hash functions are an own research topic!

Archiving


COLLECTION PRESENTATION
Selection
PREPARATION
DISCOVERY
FILTERING
QUALITY REVIEW
Hosting
Archiving
Capture
INDEXING
STORAGE
LINK EXTRACTION
FETCHING
INDEXING
STORAGE

Living Web Archives

Web Archiving

Dr. Marc Spaniol

ARC/WARC Files

- ARC/WARC files (Web ARChive) ~ 500 MB – 1 GB each
- “ZIP files” of content(s) and some metadata

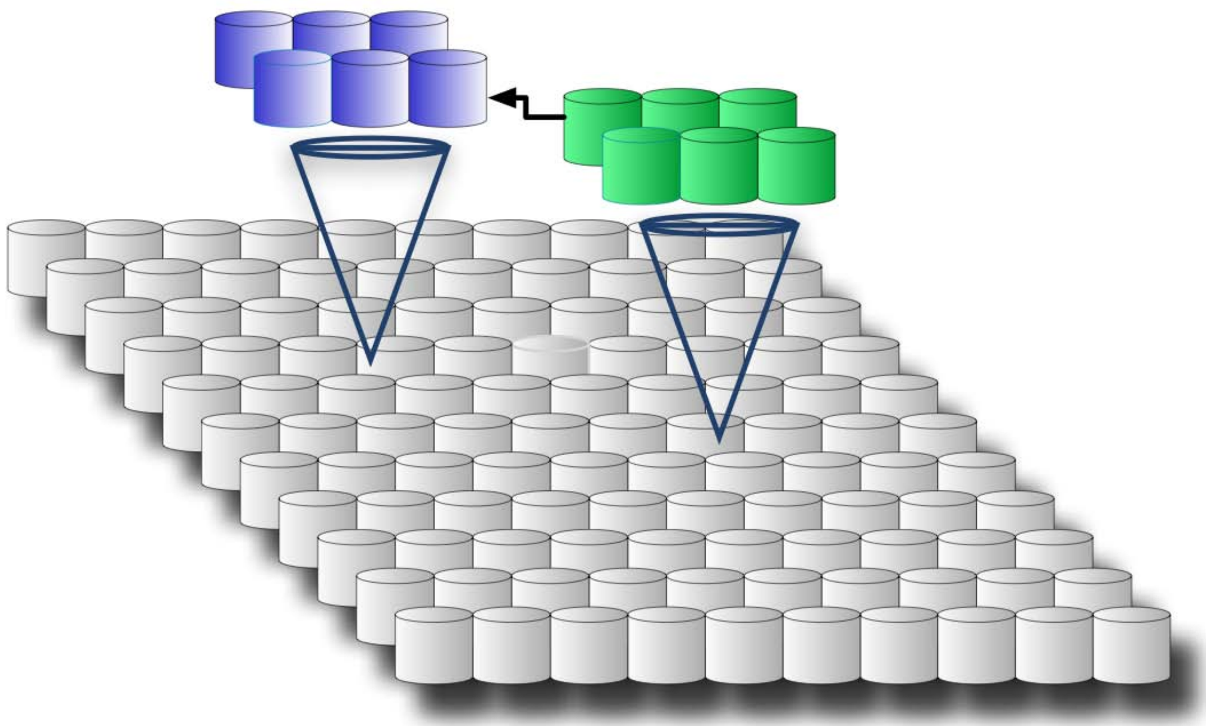
```
WARC/0.17          Record Type
WARC-Type: response                    URL
WARC-Target-URI: http://www.fedoa.unina.it/1305/01/seminar%5FMSU%5F07.pdf
WARC-Date: 2008-04-16T14:30:15Z
WARC-Payload-Digest: sha1:UOEZTJ4I3CGAQ6DNSY7ZFJ3BAVIHUMKD
WARC-IP-Address: 192.132.34.124
WARC-Record-ID: <urn:uuid:2becb774-9327-4d2c-83d3-b8d831691857>
Content-Type: application/http; msgtype=response
Content-Length: 188907 Record Size

HTTP/1.1 200 OK
Date: Wed, 16 Apr 2008 15:29:57 GMT
Server: Apache/2.0.53 (Unix) mod_perl/1.999.21 Perl/v5.8.0
Last-Modified: Tue, 18 Mar 2008 10:00:23 GMT
ETag: "141f7b-2e0ca-3470dbc0"
Accept-Ranges: bytes
Content-Length: 188618
Connection: close
Content-Type: application/pdf

%PDF-1.4
3 0 obj <<                                  Payload
/Length 735
/Filter /FlateDecode
>>
stream
x<DA><95>T<DB>R<DB>0^P}<E7>+<FC>CO<91><D0><CD><BA><94><97>R<88><99><F4>^R^X^R<A6><D
^S<D4>^Y<B3>^Yp^Z\U<E0><CC>SoH^KLE=/<BB>y^CDN<91><90><ED><AA><83><BF><BB>&<D2><B8>^
<A2>eESC7$<E7>v<F0>s<B6>q"<D0>i<CE>`<8C>^B<DD><E7>^R<<A3><85>"<DF><9A>6<CD>^D<B9><E
<FB>c^NO<D9>w<E8><94>^G^W<8A><B7>e;<B8>^P<8B><8E>o<D3><93><D9>[<84><C6><DF>^MV<9B><
<89><A8><CC><A2>^O<F9><AE><AB>^L<F3><DA>ll<85><A3>(<F9>1<B7><87>k<F3><9E>^H<FE>^X<E
```

Web Archives Grid

- Many “connected” servers
- WARC files spread among several servers
- Indexing of WARC files for access by URL and date



SURT Prefix
Sort-friendly URI Reordering Transform

- Transformation applied to URIs
  - Left-to-right representation matching the natural hierarchy of domain names
  - Useful when comparing or sorting URIs

- Conversion to SURT prefix:

1. Convert the URI to its SURT form.
2. If there are ≥ 3 slashes ('/') in the SURT form, remove everything after the last slash

   <http://(org,example,www,)/main/subsection/> ✓

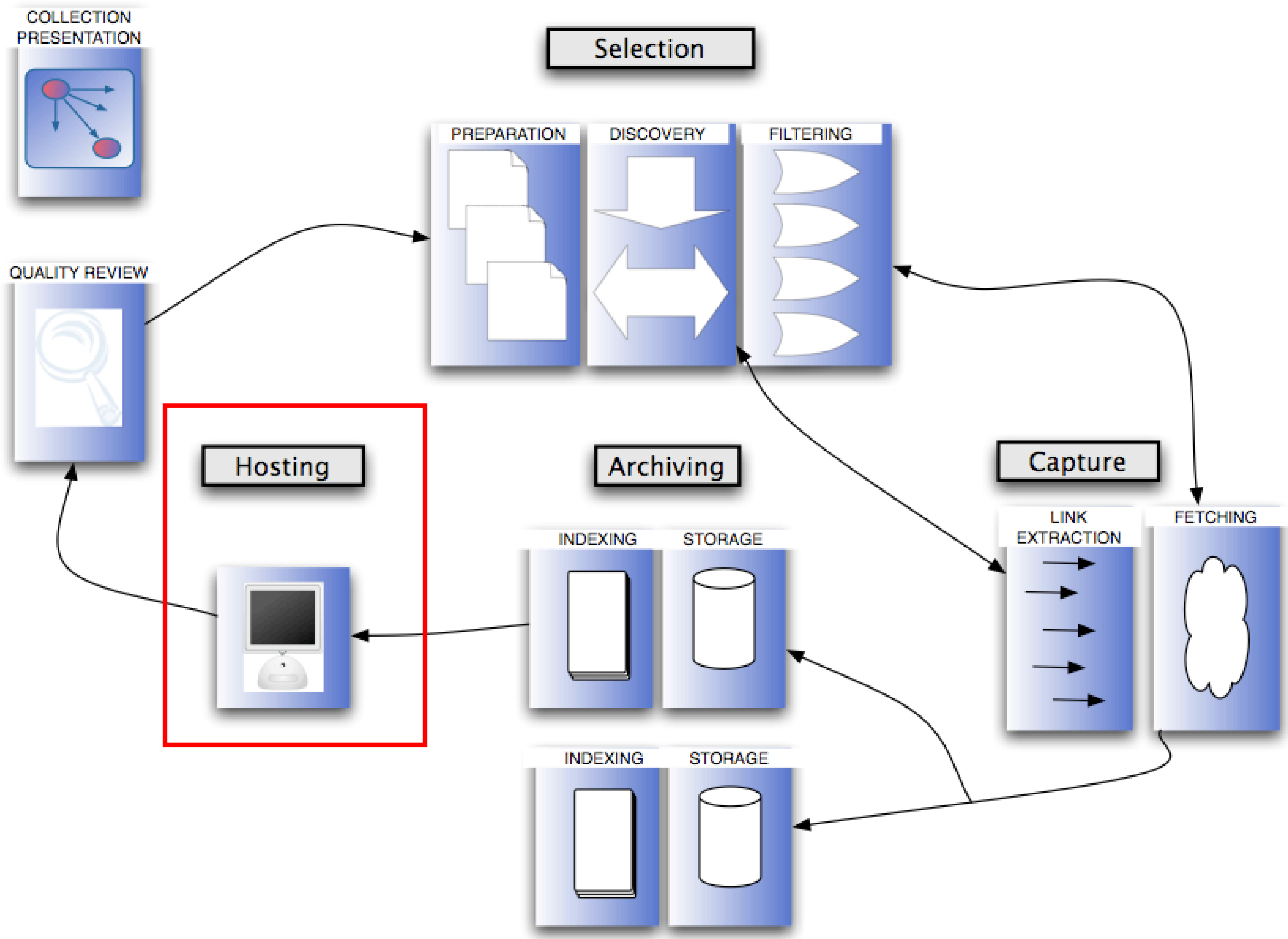
   <http://(org,example,www,)/main/subsection> → <http://(org,example,www,)/main/>

   <http://(org,example,www,)/> ✓

   <http://(org,example,www,)> ✓
3. If the resulting form ends in an off-parenthesis ')', remove the off-parenthesis

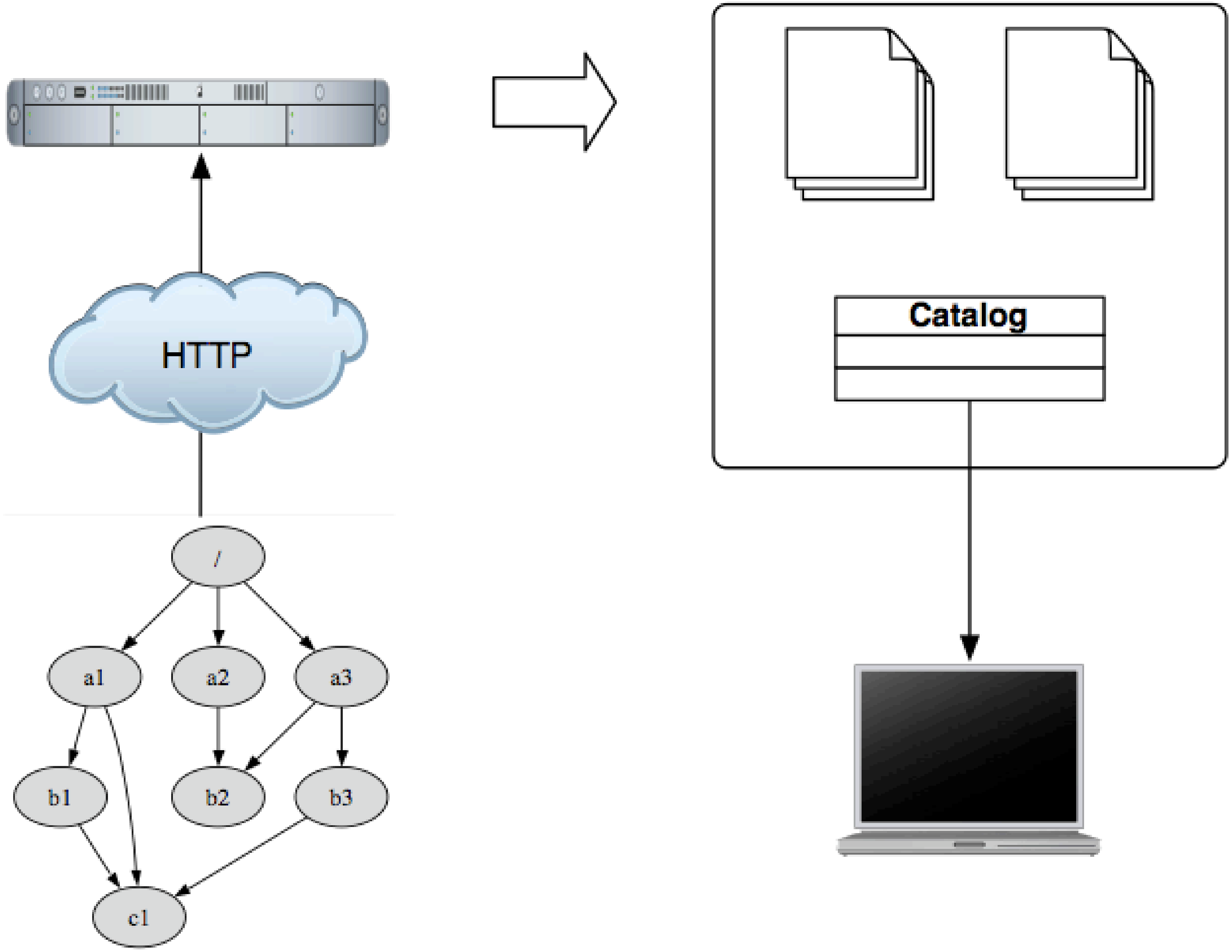
   <http://(org,example,www,)> → <http://(org,example,www,>

Hosting

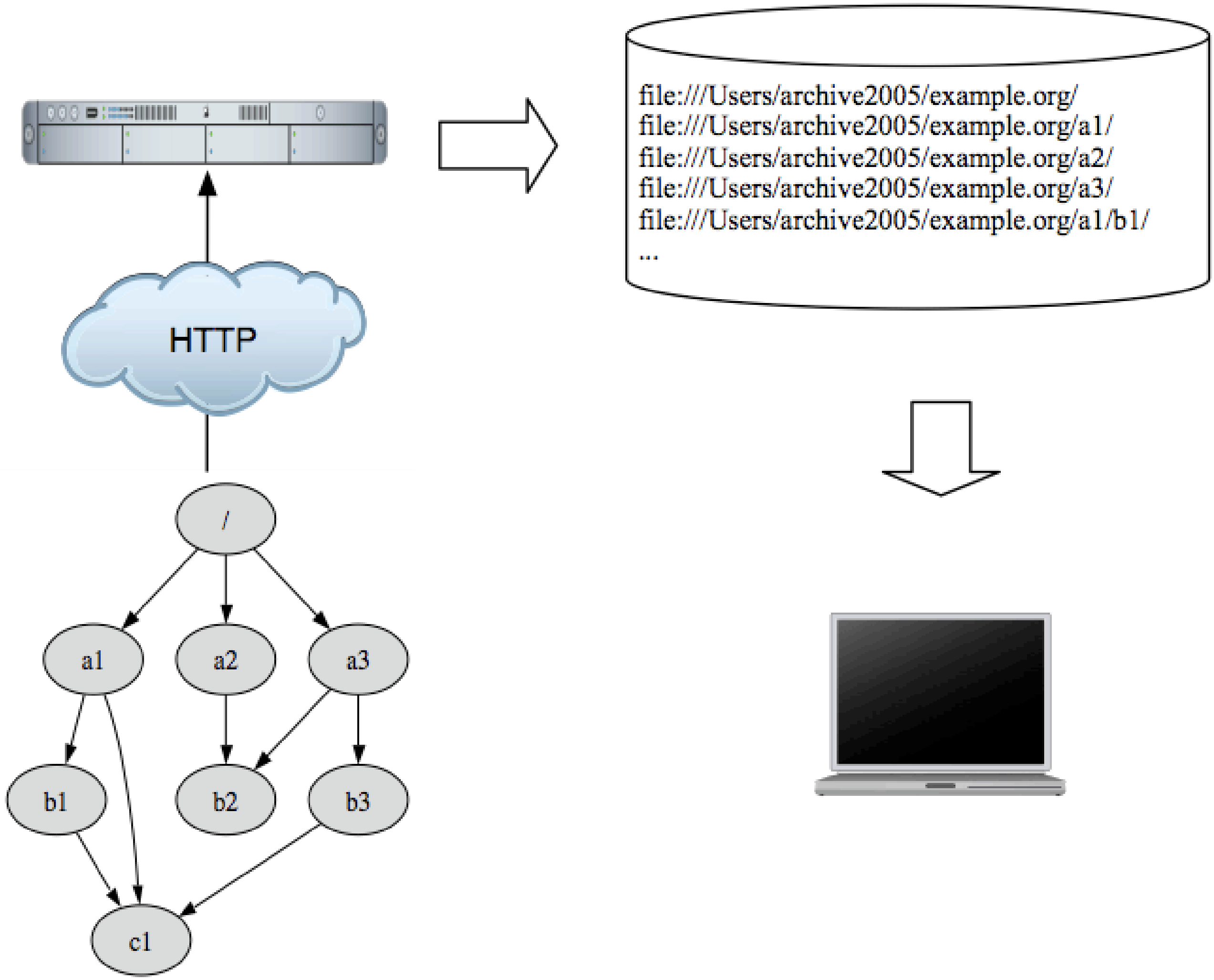


Dr. Marc Spaniol

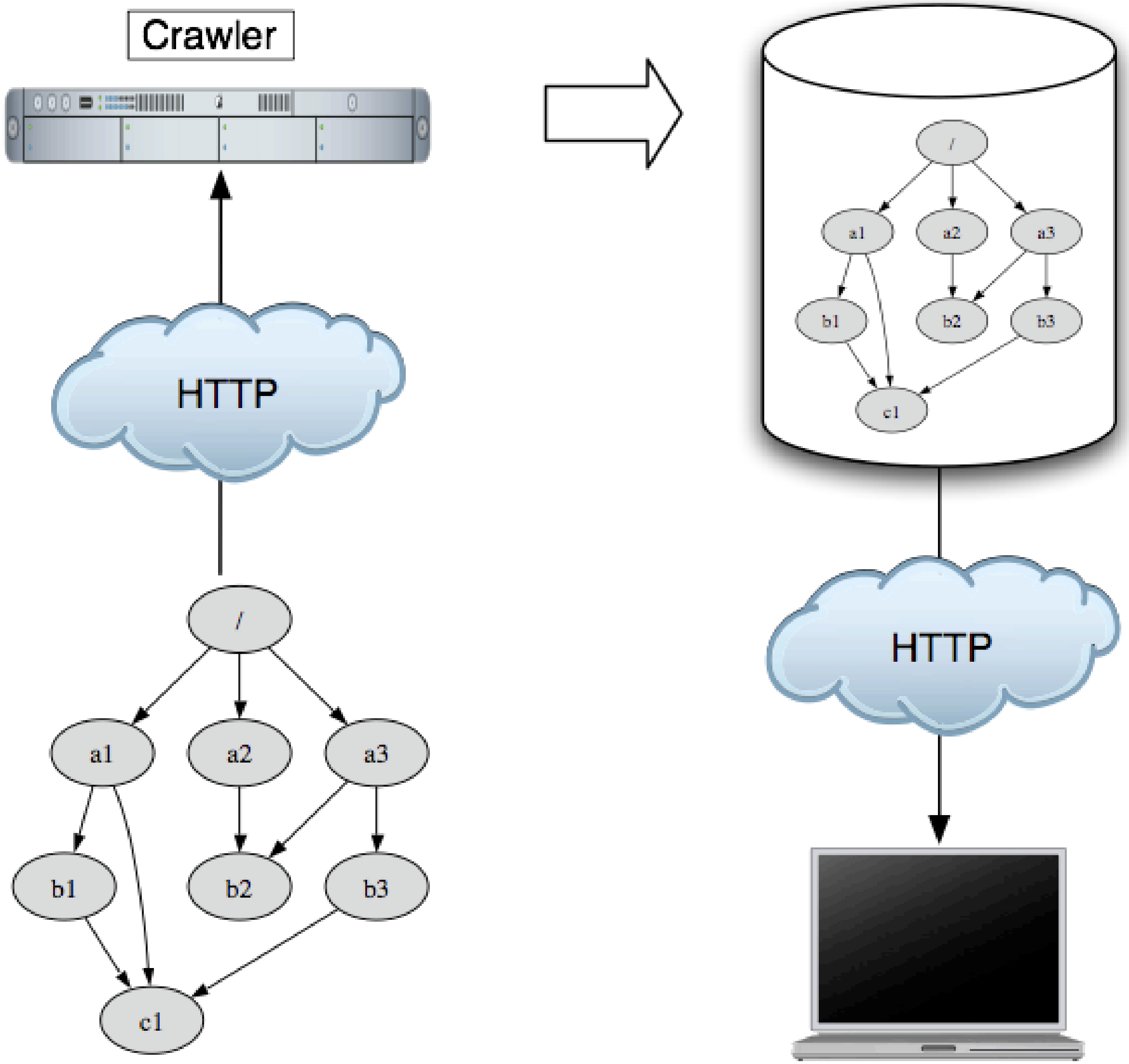
Non-Web Archive



Local File Navigation



Web-served Archive



Living Web Archives

Dr. Marc Spaniol

Hosting Approaches Summary

Approach	Benefits	Drawbacks
Non-Web Archive	+ Designed for archiving of specific (non-Web) collections + Potentially fast data access	- Cataloging (usually) does not resemble hyperlink structure - Implementation cost for cataloging logic - Special search interface required
Local File Navigation	+ Cheap + Simple + No additional infrastructure needed + Fast	- Limited accessibility - Small scale only - Links are converted in relative ones - Copying only
Web-served Archive	+ Realistic “look&feel” + Convenient navigation + Time-travel also for non-technical experienced users possible	- Web server needed - WARC/ARC file access required - Indexing tool for WARC/ARC files necessary - Time consuming sequential reads of WARC/ARC files

Summary

- Web archiving is different from Web indexing
- Important aspects of Web archiving
  - Scope of archiving requires a clear definition
  - Seeds need to be carefully selected
  - Preservation of hidden or dynamically generated contents is almost impossible
  - Sitemaps rarely exist
  - WARC file processing is the bottleneck in retrieval
  - Capturing takes a long time (!!!) and contents may not fit to each other
- Identification of coherence
  - HTTP time stamps $\rightarrow$ Measurable coherence
  - "Virtual" time stamps $\rightarrow$ Inducible coherence
- Measuring data quality in Web archives requires
  - Identification of relevant coherence defects
  - Data scrubbing in order to compare relevant document (sub-)sections
  - Efficient computations
    - Shingling
    - Minhashing

References

[Chak03] S. Chakrabarti: "Mining the Web". Morgan Kaufmann, 2003.

[Masa06] J. Masanès: "Web Archiving". Springer, New York, Inc., Secaucus, NJ, 2006.

[Ullm00] J. Ullman: "Correlated Items". CS345 --- Lecture Notes, 2000.
http://www-db.stanford.edu/~ullman/mining/minhash.pdf
[last access: May 25, 2010]

[SDM*09] M. Spaniol, D. Denev, A. Mazeika, P. Senellart and G. Weikum: "Data Quality in Web Archiving". Proceedings of the 3rd Workshop on Information Credibility on the Web (WICOW 2009), pp. 19-26, 2009.
http://www.dl.kuis.kyoto-u.ac.jp/wicow3/papers/p19-spaniolA.pdf
[last access: May 25, 2010]