Peer-to-Peer Web Archival and Time-Travel Search

by:
Avishek Anand

supervised by: Srikanta Bedathur, Christos Tryfonopoulos

Motivation

- Web is constantly changing and evolving. “Only 20% of the existing web-pages will exist after a year”
- Web today contains information from all walks of life and capturing these changes is essential
- Web archiving is the process of collecting portions from the web and ensuring the collection is preserved in an archive for future analysis

Motivation

- Searching on the archives makes historical analysis possible. E.g "John McCain @ 9/11/2001"
- Current attempts like Internet Archive (IA) have limited search capability and their scalability is also questionable
- We propose a scalable, decentralized and peer-to-peer(p2p) approach to web archiving which supports time-travel search

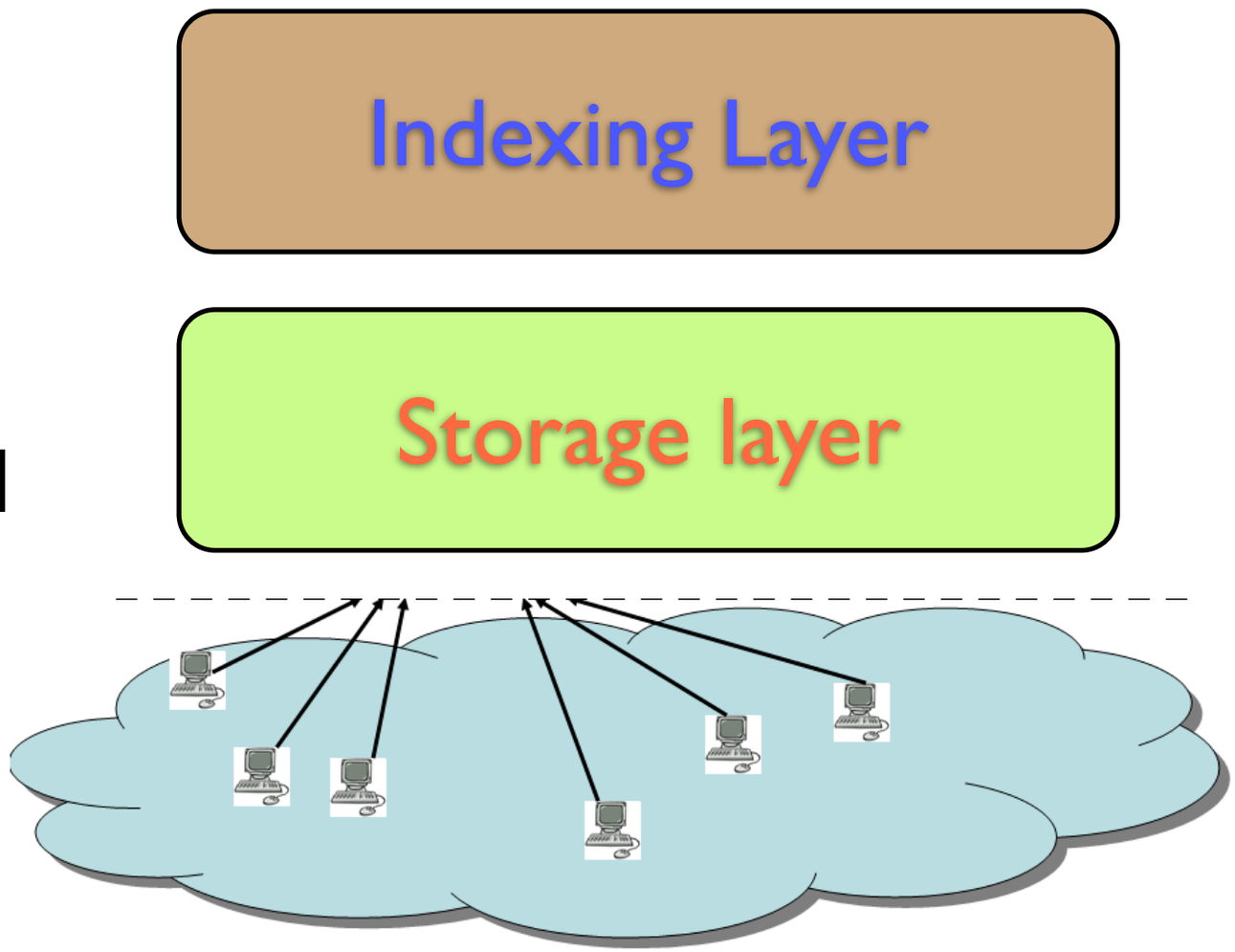
Outline

- P2P Framework for Web Archival
- Index organization for time-travel queries
- Partitioning Strategies for Index lists
- Results
- Conclusion

System Architecture

P2P Architecture for archival and search

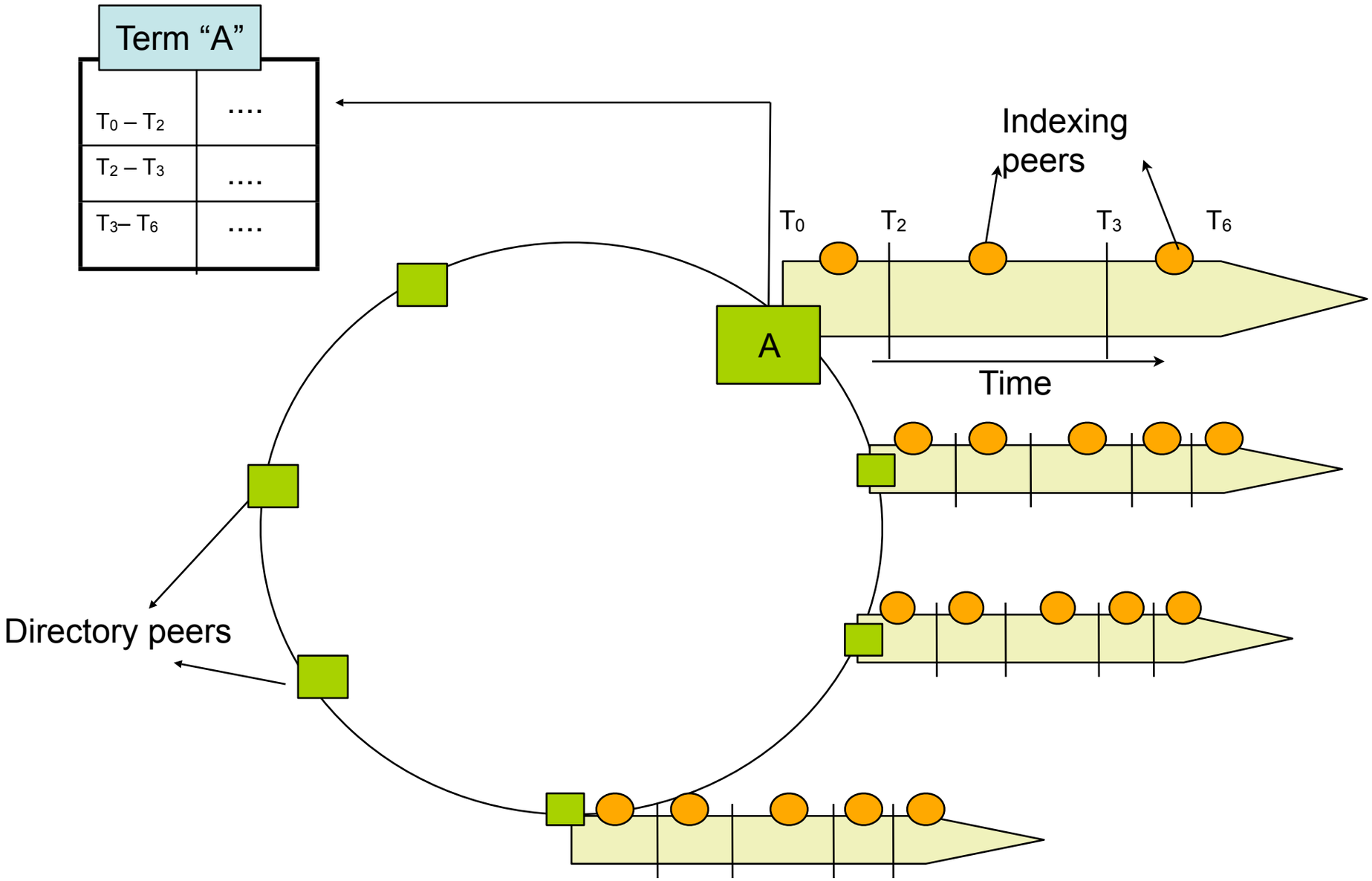
- Crawling peers push content periodically to storage layer
- Storage peers organized to store data in a durable and persistent way
- Indexing peers organized to process queries efficiently



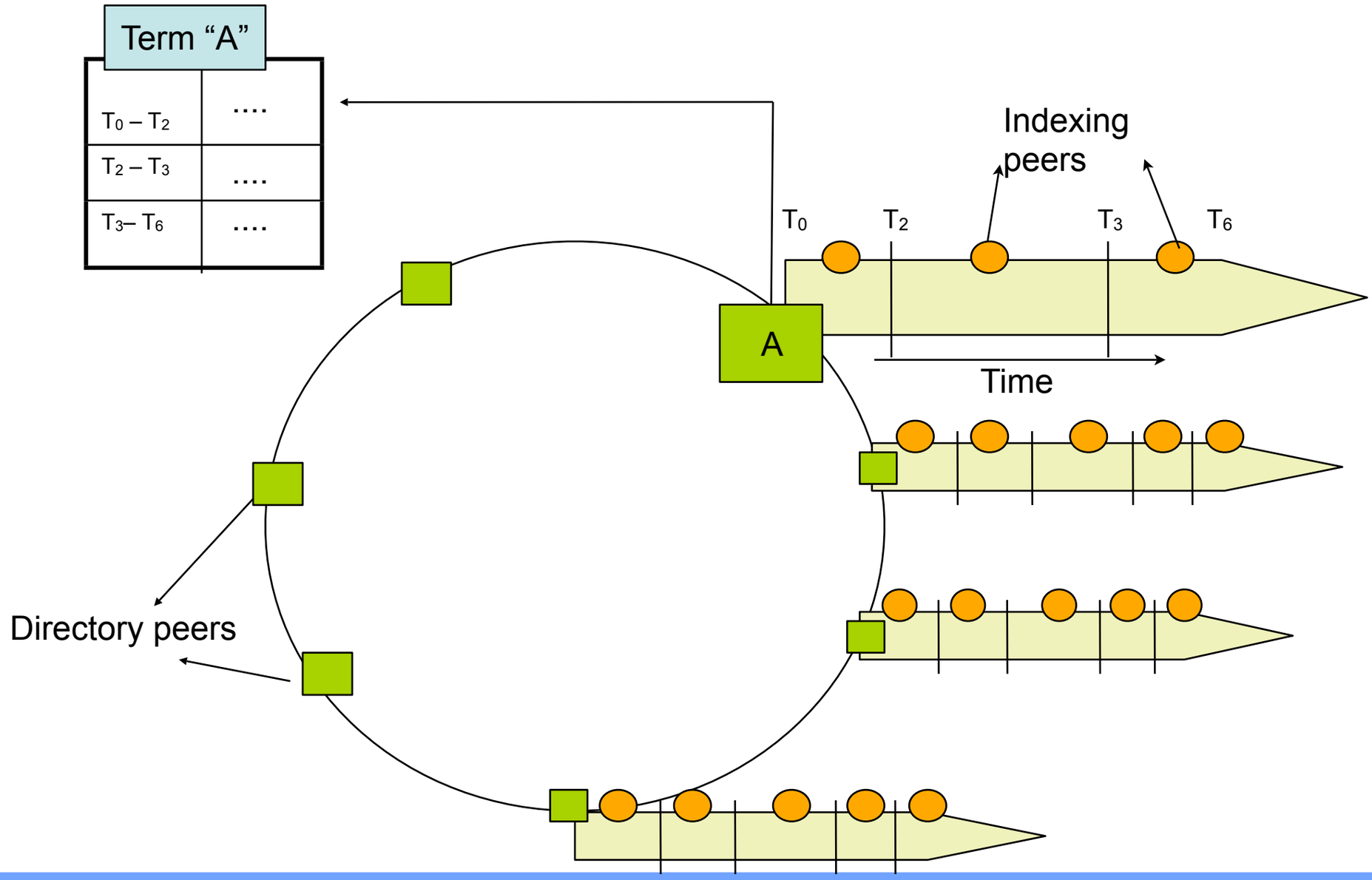
Index Partitioning

- Indexing layer is organized such that peers holds part of an index-list.
- Index are partitioned both along the time-axis and in term-space.
- Term Partitioning based on the hashing used by structured p2p overlays (Chord, Pastry etc.).
- Time Partitioning is partitioning the index list per term in the time axis

Term-Time Partitioning

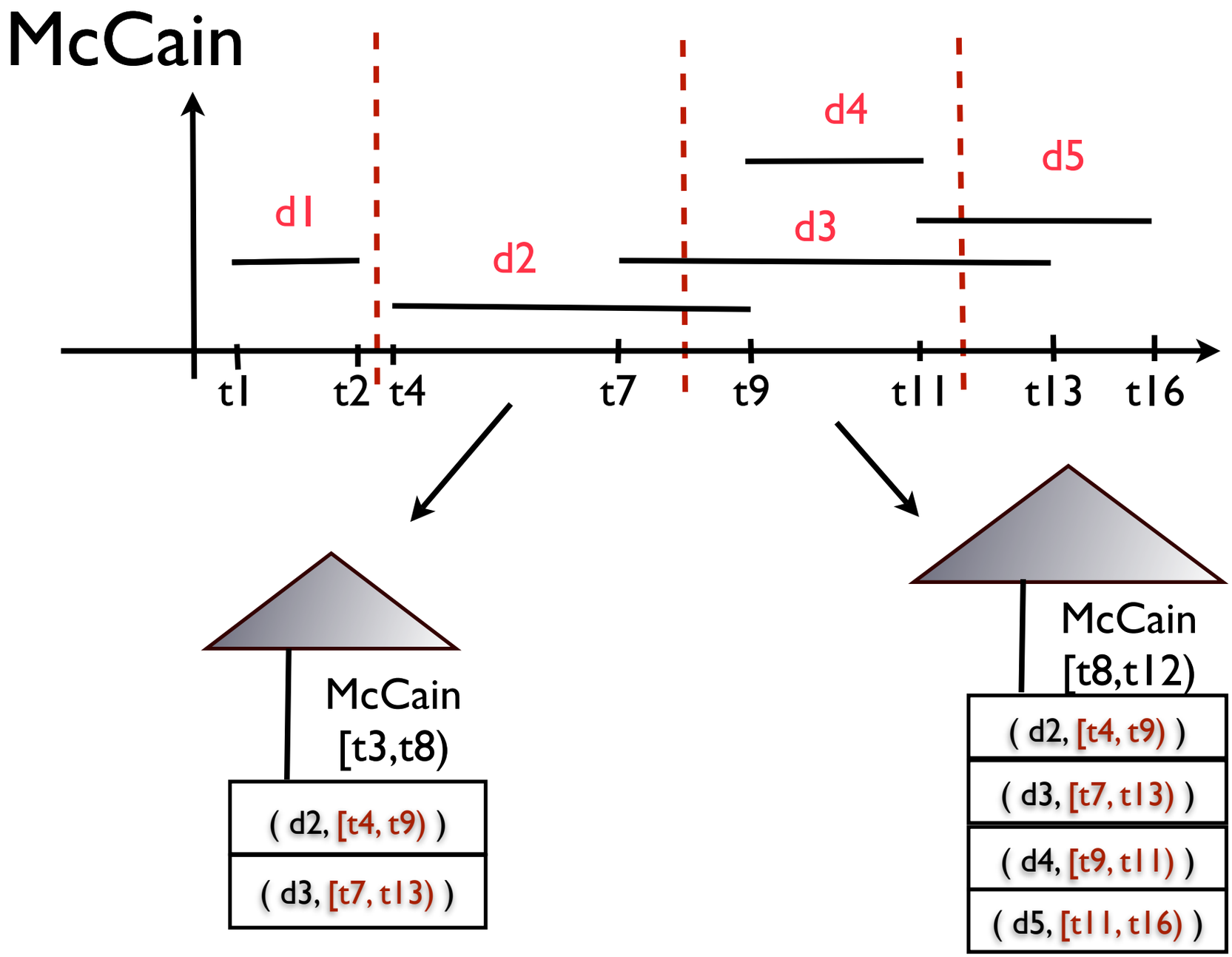


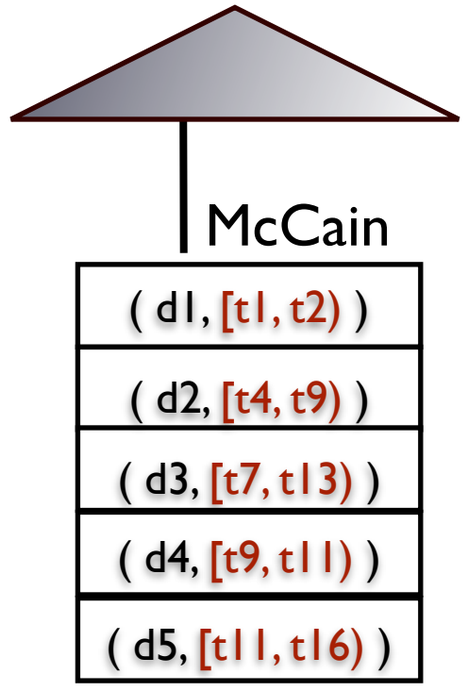
Term-Time Partitioning



A time-travel query “A @ t” routed to the directory peer responsible for “A” and then the indexing peer responsible for “t”

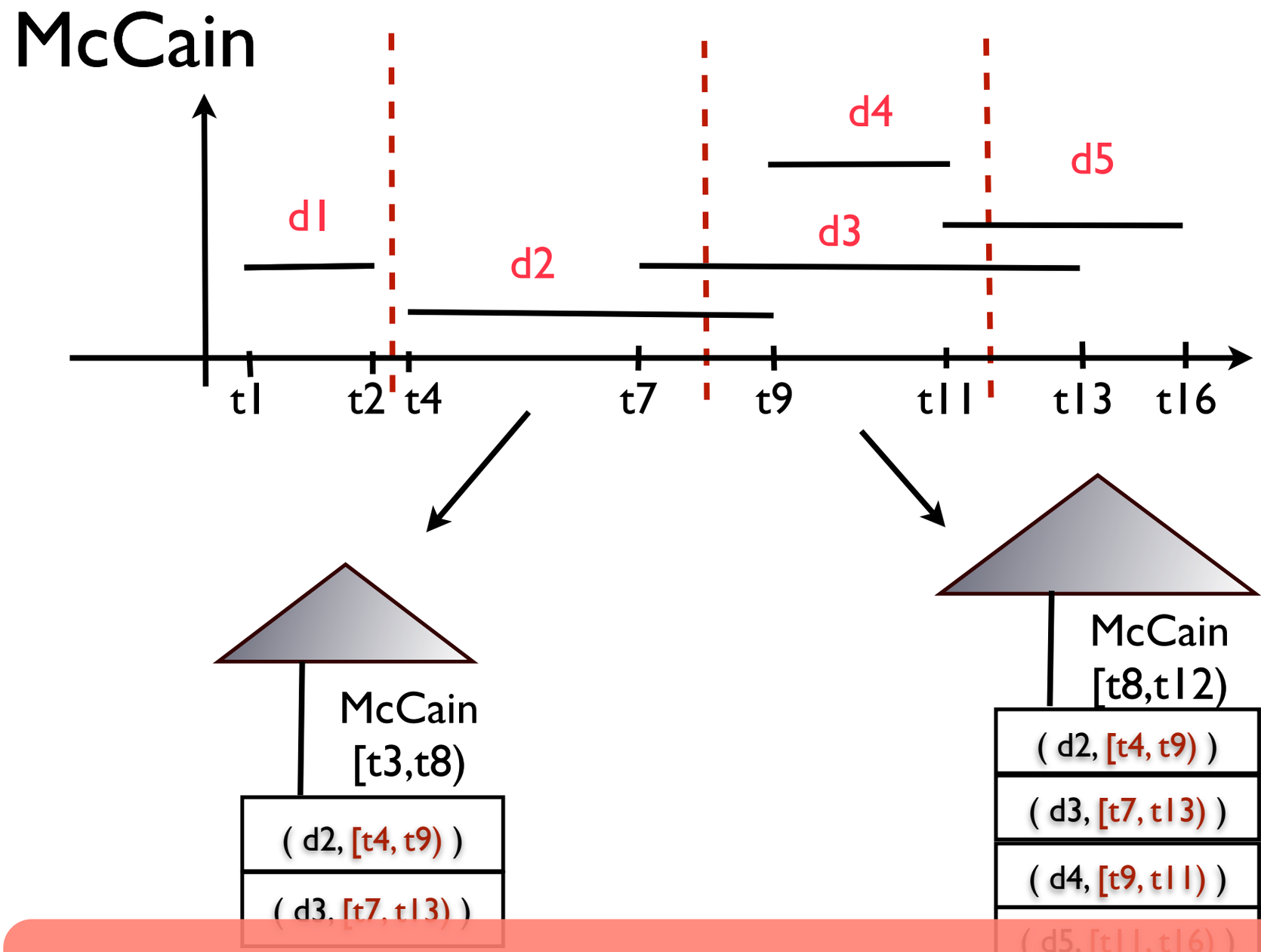
Index Partitioning

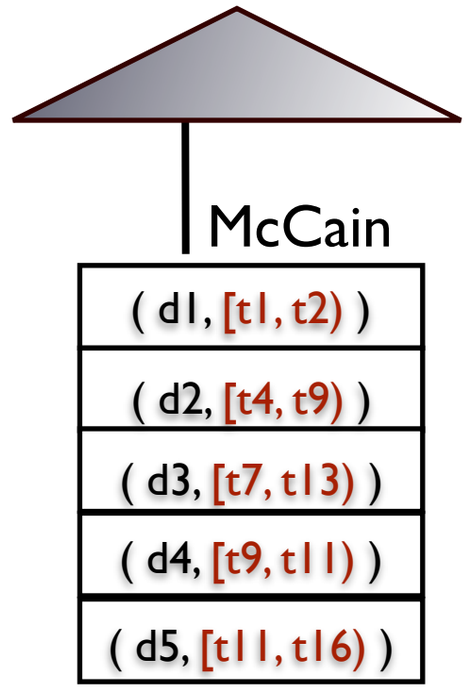




Each version is a new entry, hence long list:
need partitioning

Index Partitioning





Each version is a new entry, hence long list:
need partitioning

Overall size increases due to duplicate entries

Partitioning Strategies

- Centralized Setting : Key issue is to provide performance while partitioning index [Berberich et. al, SIGIR '07]

- Distributed Setting :
  - Loss of data due to churn (leaving of peers)
  - Reconstruction from underlying storage is expensive
  - We leverage the replication due to partitioning to support index reconstruction

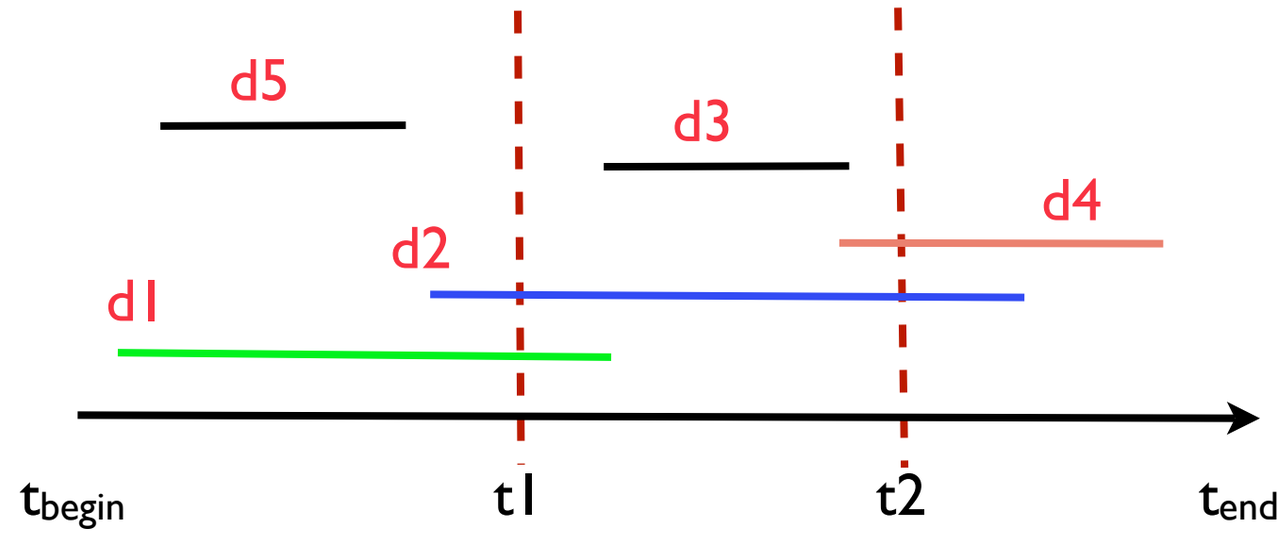
Partitioning Strategies

Live Entries in a Partition

- Live entries in a partition L_v: $[t_i , t_j)$:

$$L_v : [t_i, t_j) = \{(d, p, [t_b, t_e)) \in L_v \mid t_b <= t_j \wedge t_e > t_i\}$$

- Live entries in a partition could be classified into left, right, strikethrough and subsumed.



L_v: [t1, t2) = {d1, d2, d3, d4} L_v: t1 = {d1, d2}

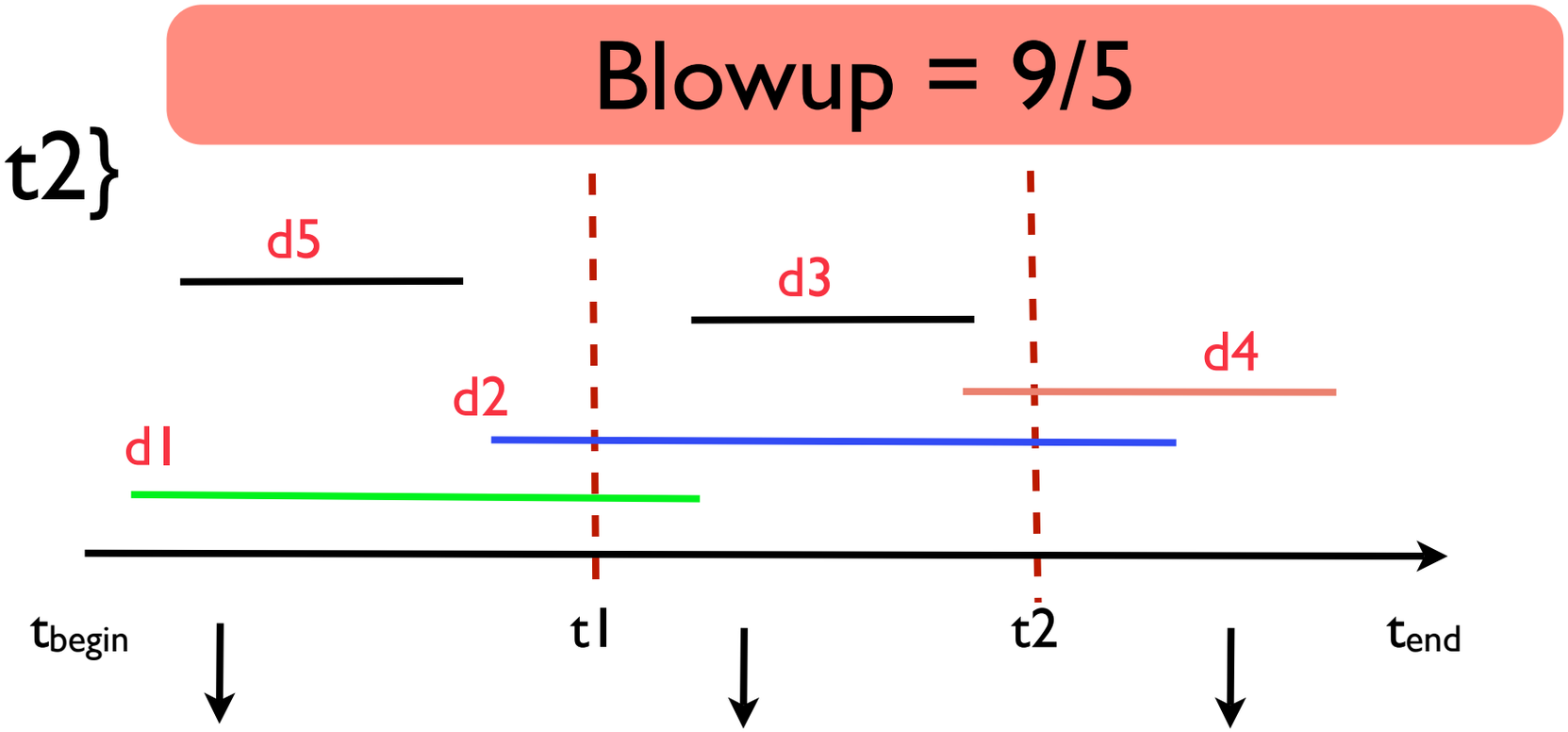
(d, $[t_b , t_e)$) is the posting for the document **d** which begins at **t_b** and ends at **t_e**.

L_v set of all the documents in the collection having term v

Blowup

- Blowup (ϒ): Ratio of sum of live entries in each partition to the total number of entries

P (M) = {t1, t2}



Live entries : 3 4 2

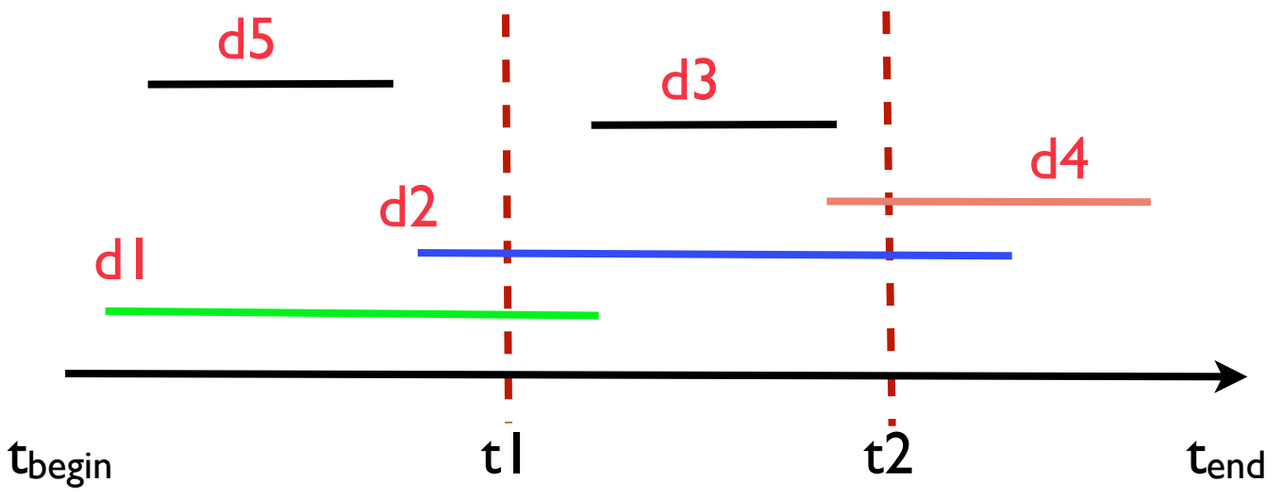
Replication

- Replication(R) measures the fraction of index entries that are ever replicated or replicated at least once.

$$\mathcal{R}(\mathcal{M}) = \frac{|\bigcup_{\mathbf{t_i} \in \mathbf{P}(\mathcal{M})} L_v : t_i|}{|L_v|}.$$

P(M) is the set of time-points where the partitions exists

Replication R(M) = 3/5

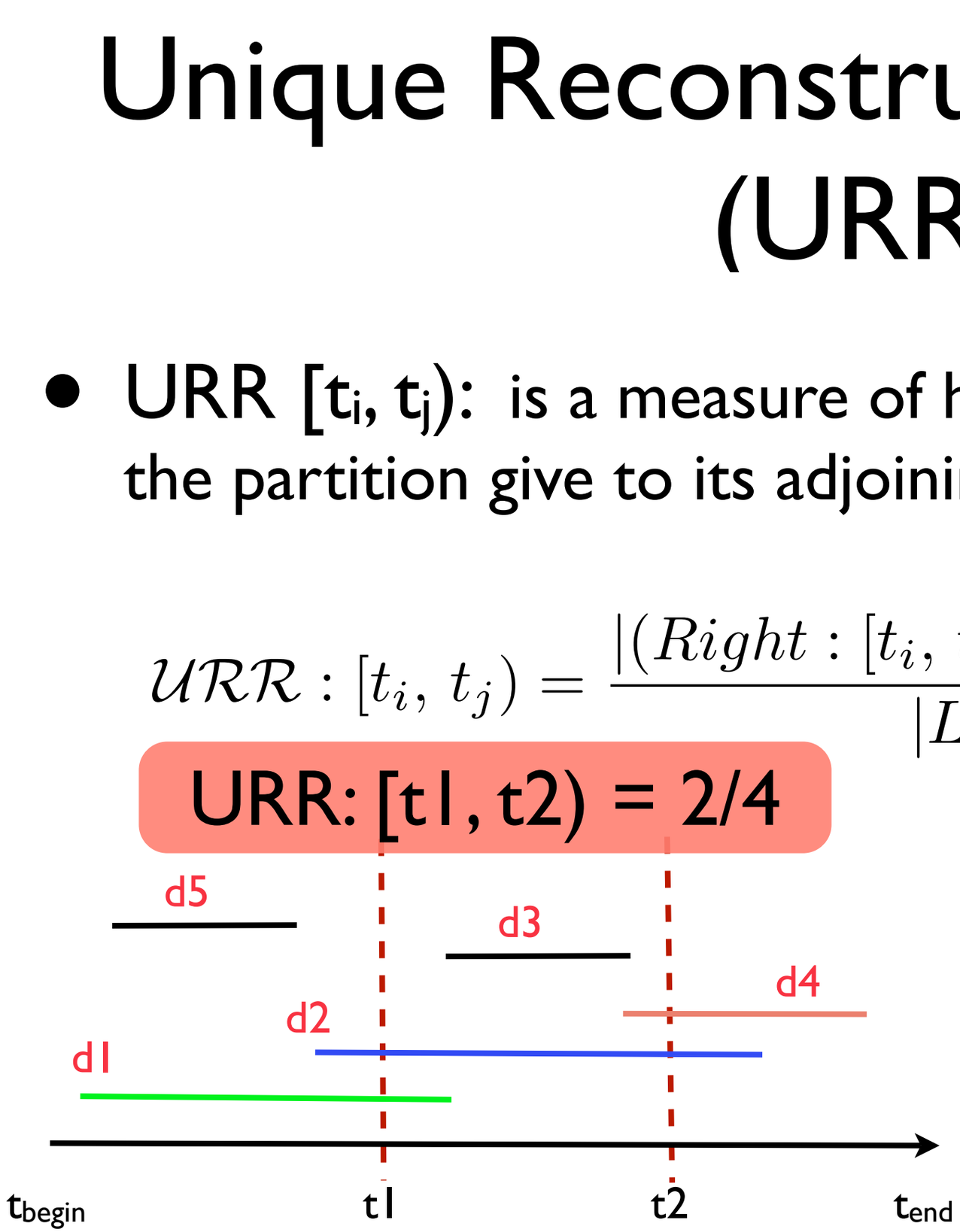


P (M) = {t1, t2}

Unique Reconstructibility Ratio (URR)

- URR $[t_i, t_j)$: is a measure of how much benefit does the partition give to its adjoining partitions

$$\mathcal{URR} : [t_i, t_j) = \frac{|(Right : [t_i, t_j)) \cup (Left : [t_i, t_j))|}{|L_v : [t_i, t_j)|}$$



Right: [t1, t2) = {d4}

Left: [t1, t2) = {d1}

L_v: [t1, t2) = {d1,d2,d3,d4}

Maximum Replication

- **Maximum Replication (Max-Rep):** Find a partitioning to Maximize replication keeping the blowup in check.

- Max-Rep is NP-Hard : Reduction from subset-sum problem
- Optimal implementation (Branch-and-Bound) for a limited time-range

$$\arg\max_{\mathcal{M}} \mathcal{R}(\mathcal{M}) \quad \text{s.t.}$$

$$\sum_{m \in \mathcal{M}} |L_v : m| \leq \gamma |L_v| \, .$$

Maximum Minimum Reconstructibility

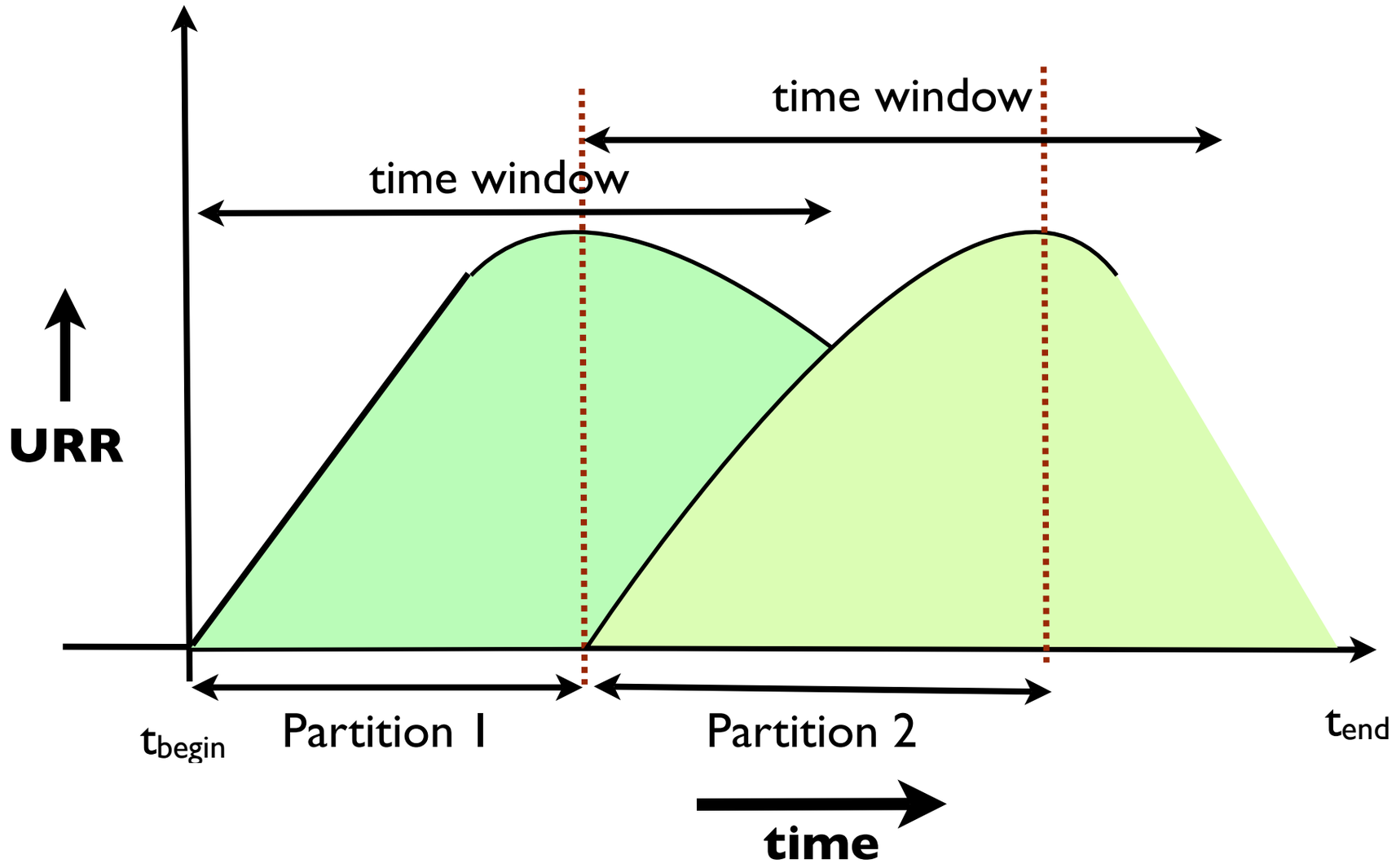
- **Max-Min Reconstructibility:** Find a partitioning to Maximize the minimum URR keeping the blowup in check.

- Complexity yet to be established
- Heuristic approaches like Greedy considered
- Importance of approaches which do not change the past

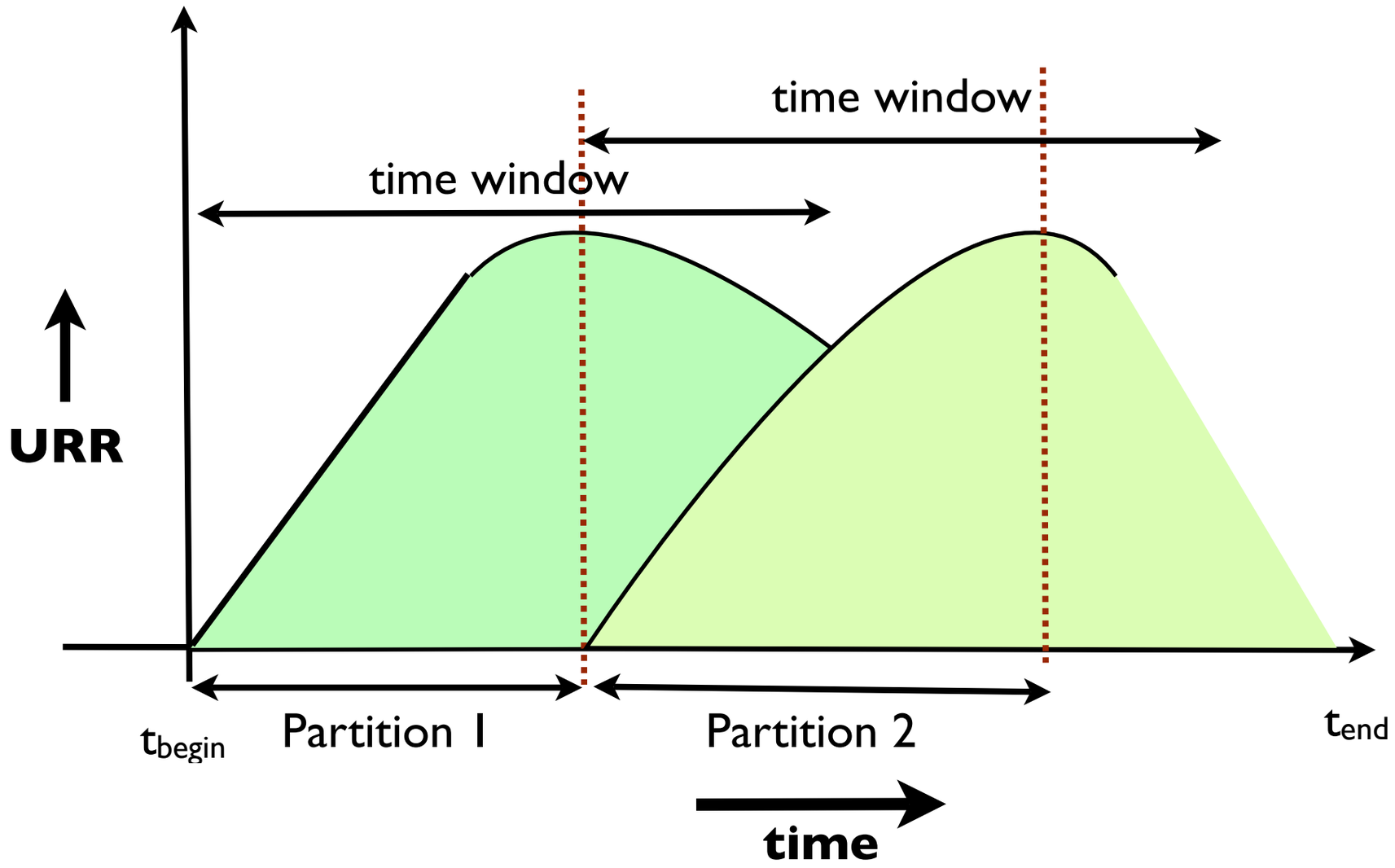
$$\arg\max_{\mathcal{M}} \min_{\mathbf{m}\in\mathcal{M}} \mathcal{URR}(\mathbf{m}) \quad \text{s.t.}$$

$$\sum_{m\in\mathcal{M}} |L_v : m| \leq \gamma\,|L_v| \ .$$

Greedy Approach



Greedy Approach



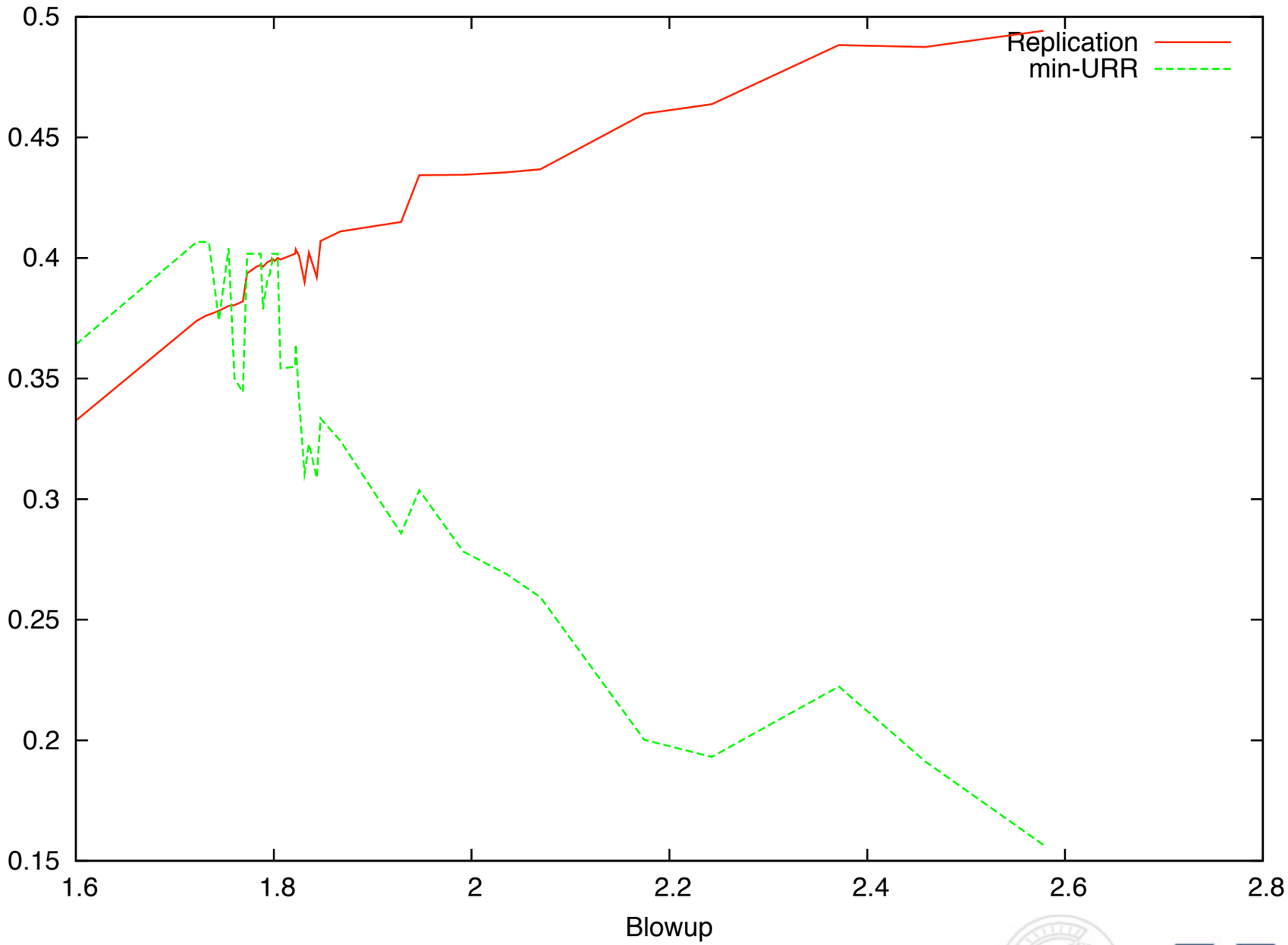
Time-points with the highest values of URR in a given time-window are chosen as partitions

Experimental Setup

- We use the version lifetimes from **Wikipedia**
- Each version considered as a separate document
- We consider lifetimes in day-level granularity
- Plot response of min-URR, Replication (R) with varying blowup values

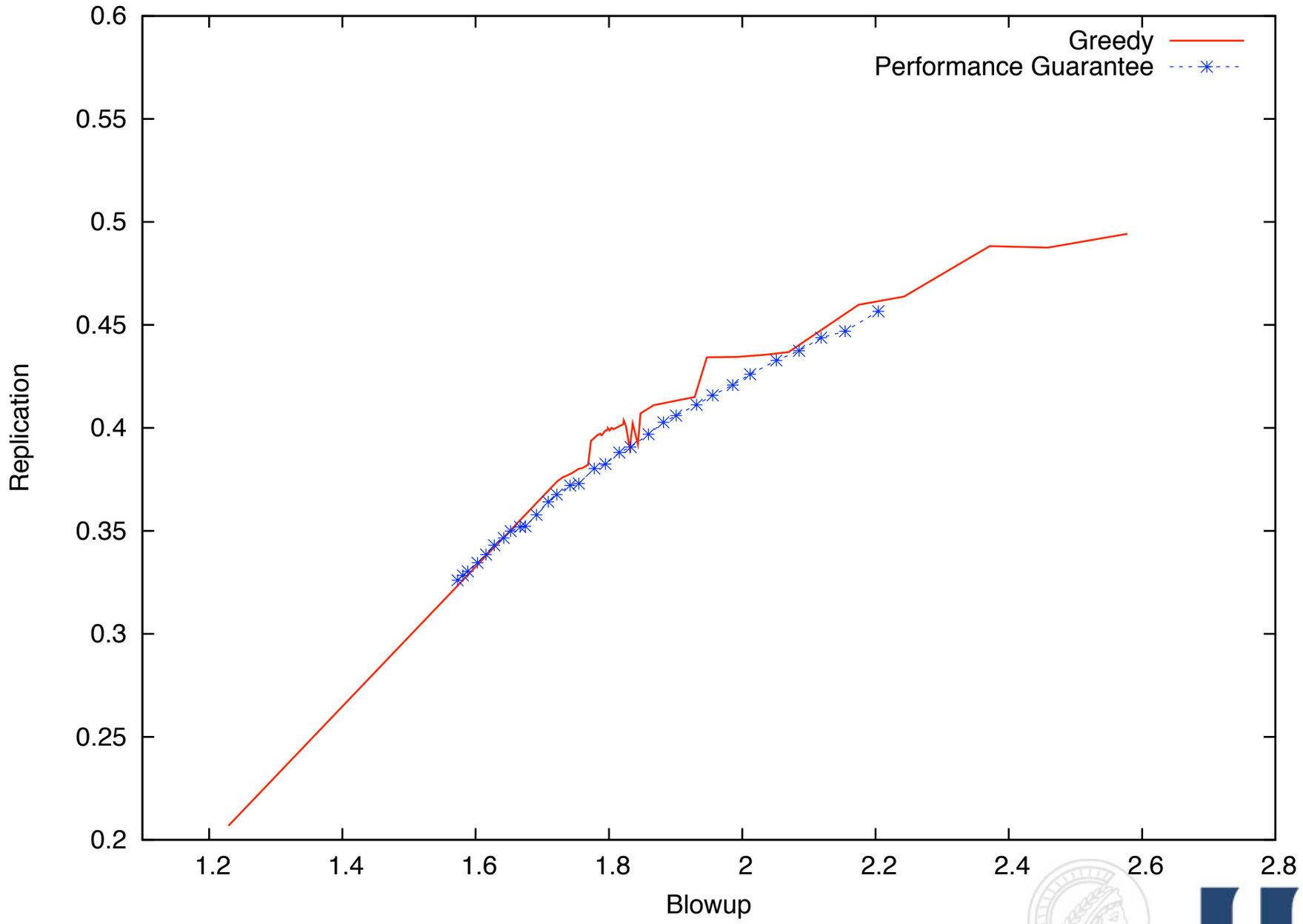
Results

Analysis of min-URR, blowup and R using Greedy approach



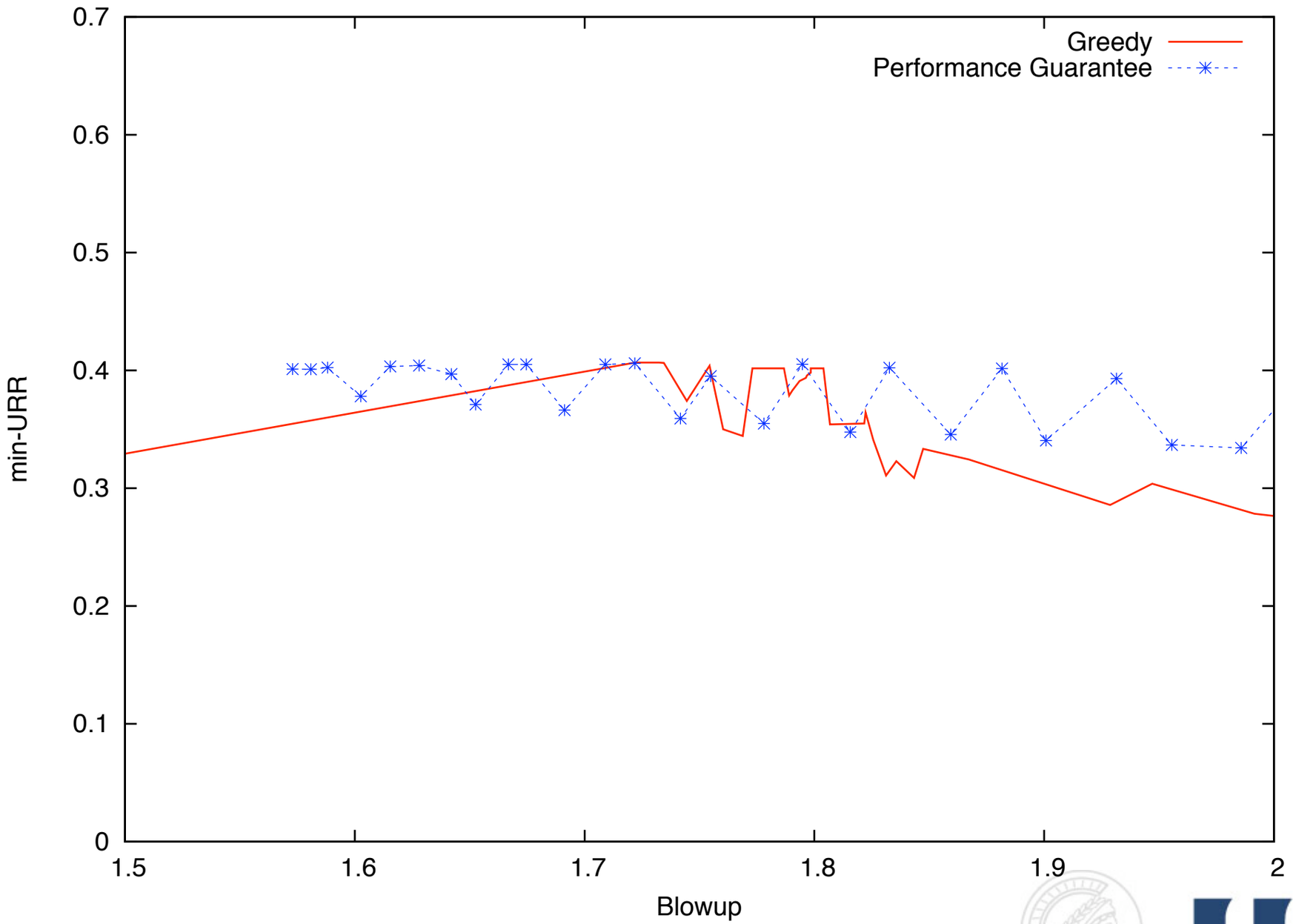
Results

Analysis of Performance-Guarantee vs Greedy for Replication



Results

Analysis of Performance-Guarantee vs Greedy for min-URR



Future Work

- To determine complexity of MMR and come up with centralized solutions
- To look at combination of both Min-URR and Replication as an optimization problem
- Explore possible applications of partitioning strategies to other areas involving continuous attributes

Thank You

References

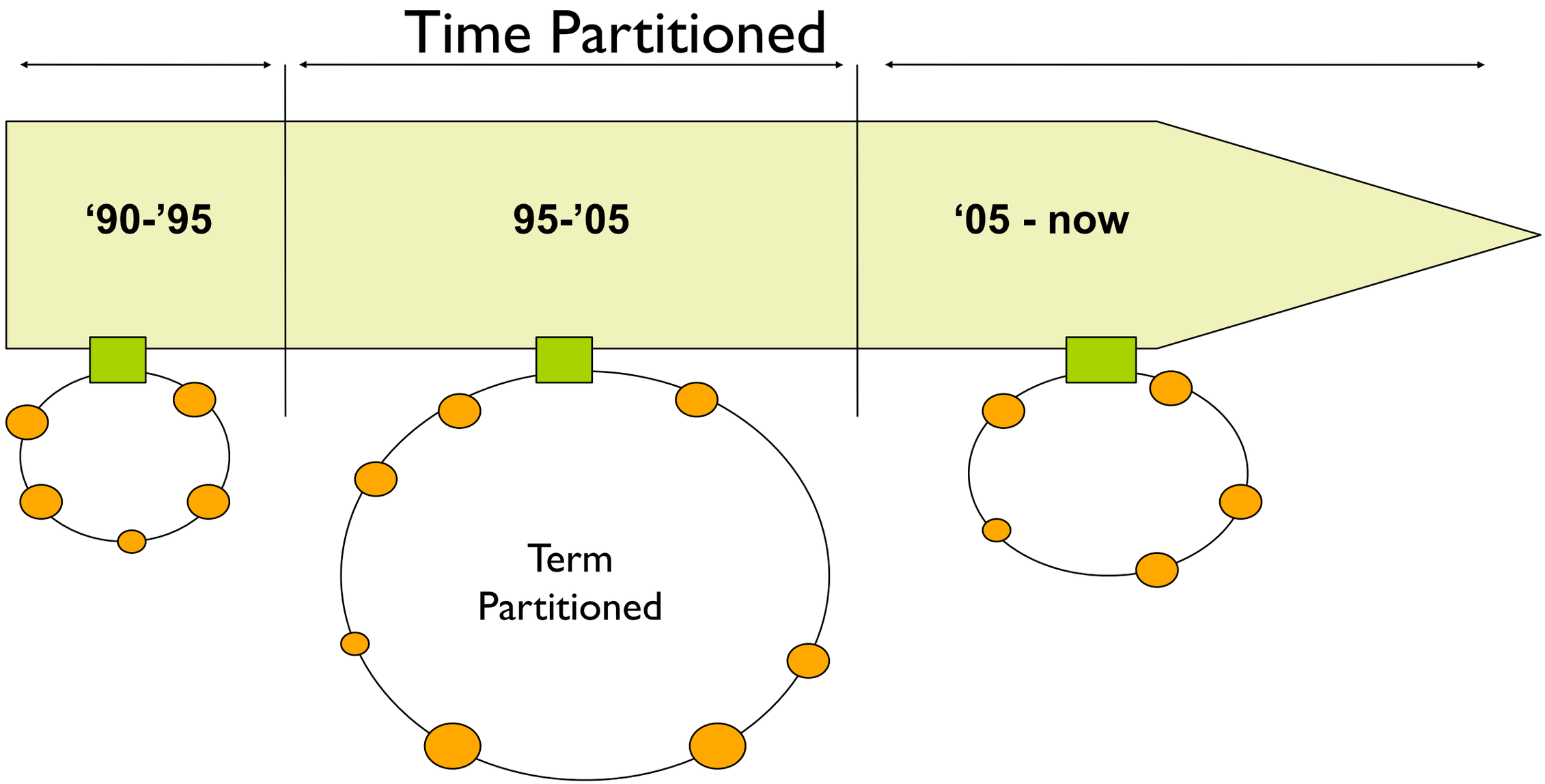
- [1] Klaus Berberich, Srikanta Bedathur, Thomas Neumann, Gerhard Weikum: *A Time Machine for Text Search* SIGIR 2007, July 2007

- [2] Srikanta Bedathur, Avishek Anand, Klaus Berberich, Ralf Schenkel, Christos Tryfonopoulos: Realizing an Everlasting Web, under submission CIDR 2009.

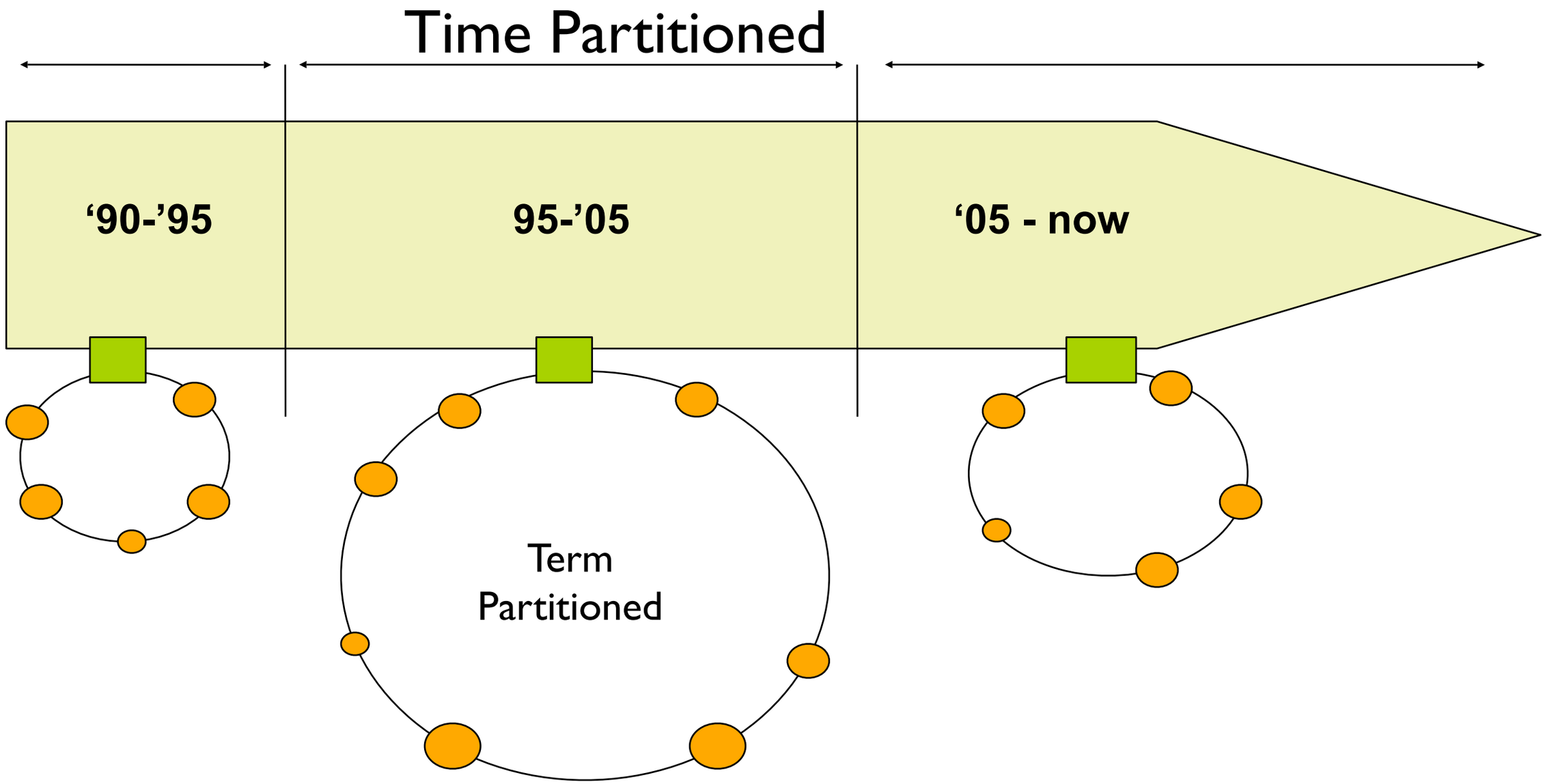
- [3] Klaus Berberich, Srikanta Bedathur, Thomas Neumann, Gerhard Weikum: *A Time Machine for Text Search* MPII Technical Report MPI-I-2007-5-002, July 2007

- [4] http://www.wikipedia.org/

Time-Term Partitioning

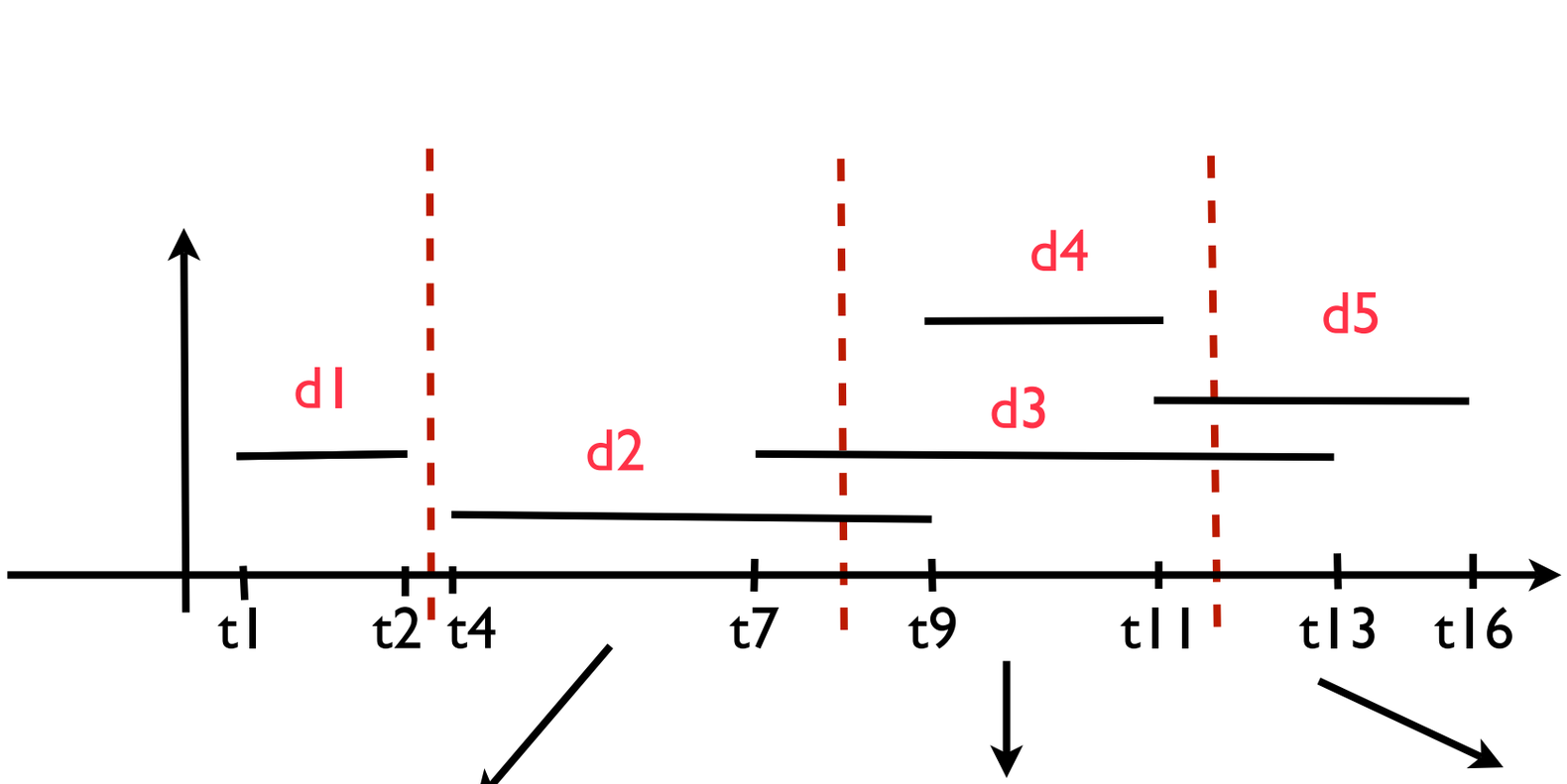


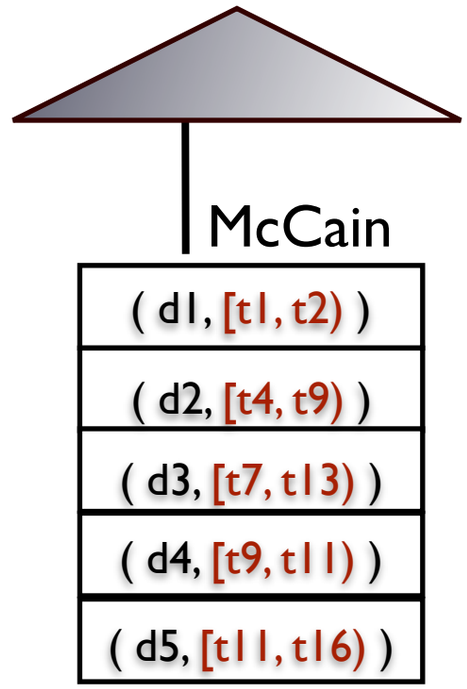
Time-Term Partitioning

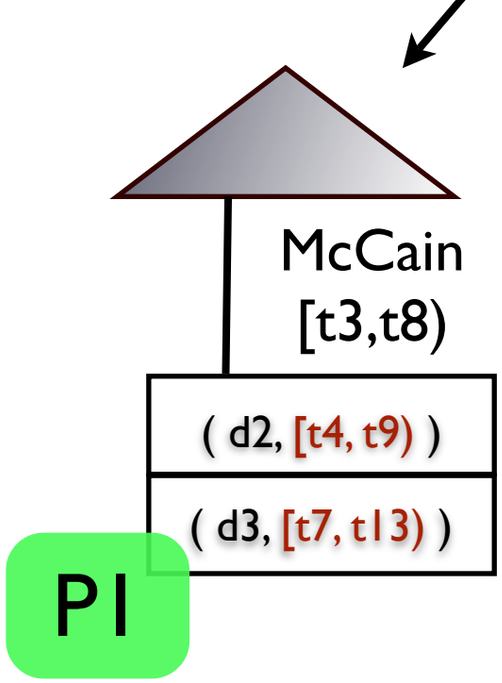


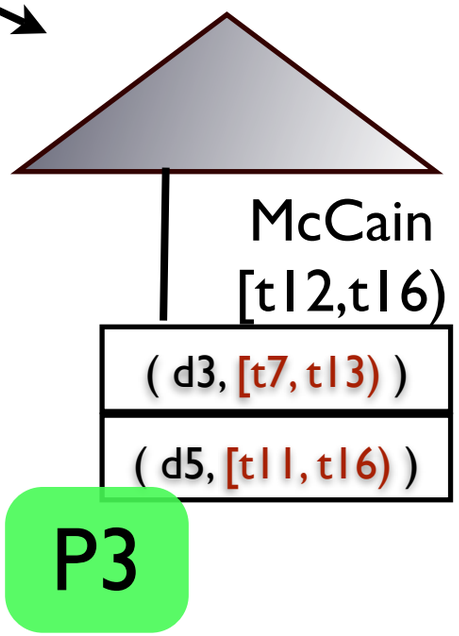
A query "q @ t" routed to gateway peer responsible for "t" and then the peer responsible for "q"

Index Partitioning

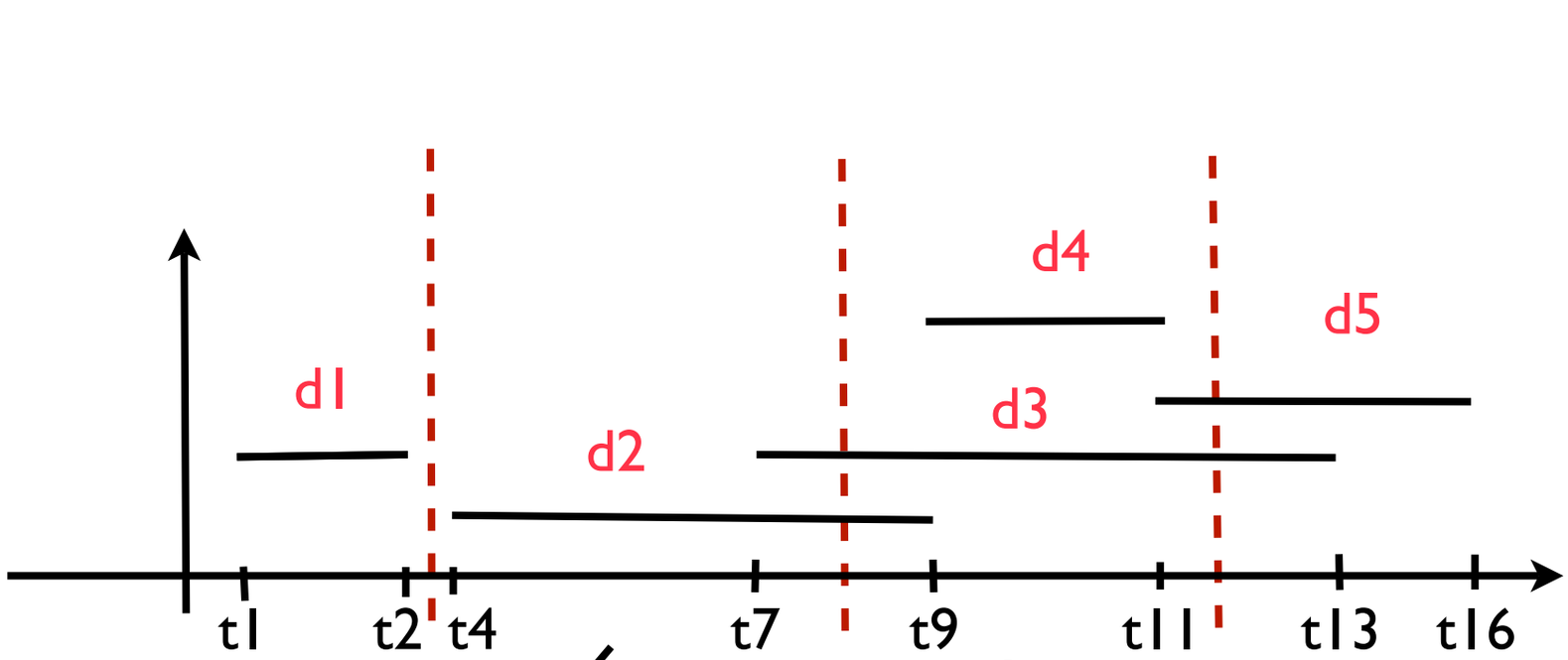


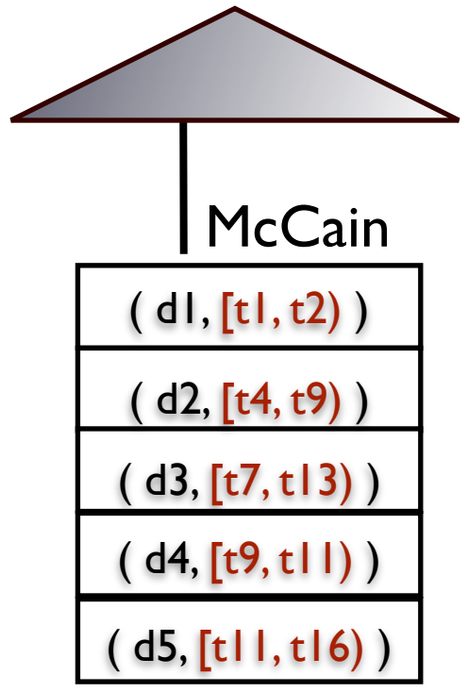


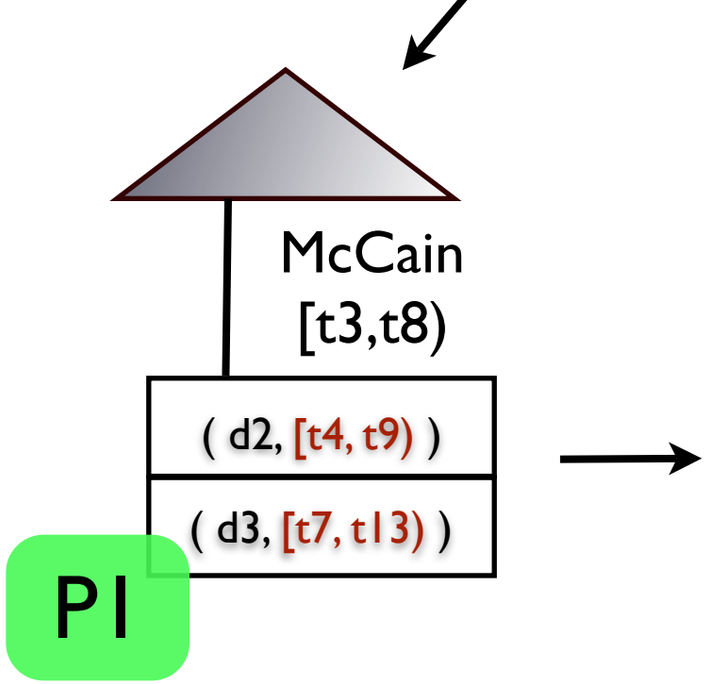


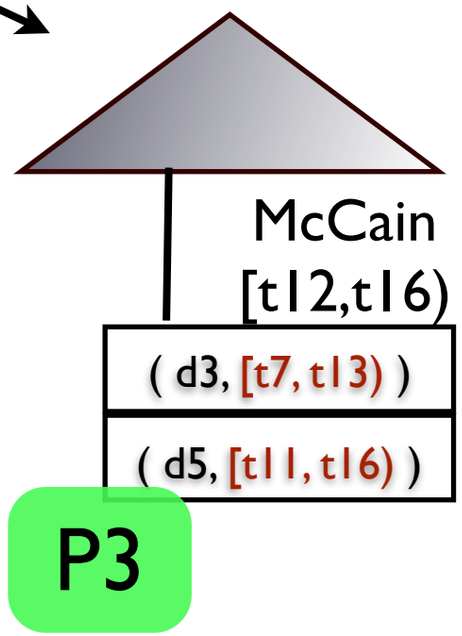


Index Partitioning

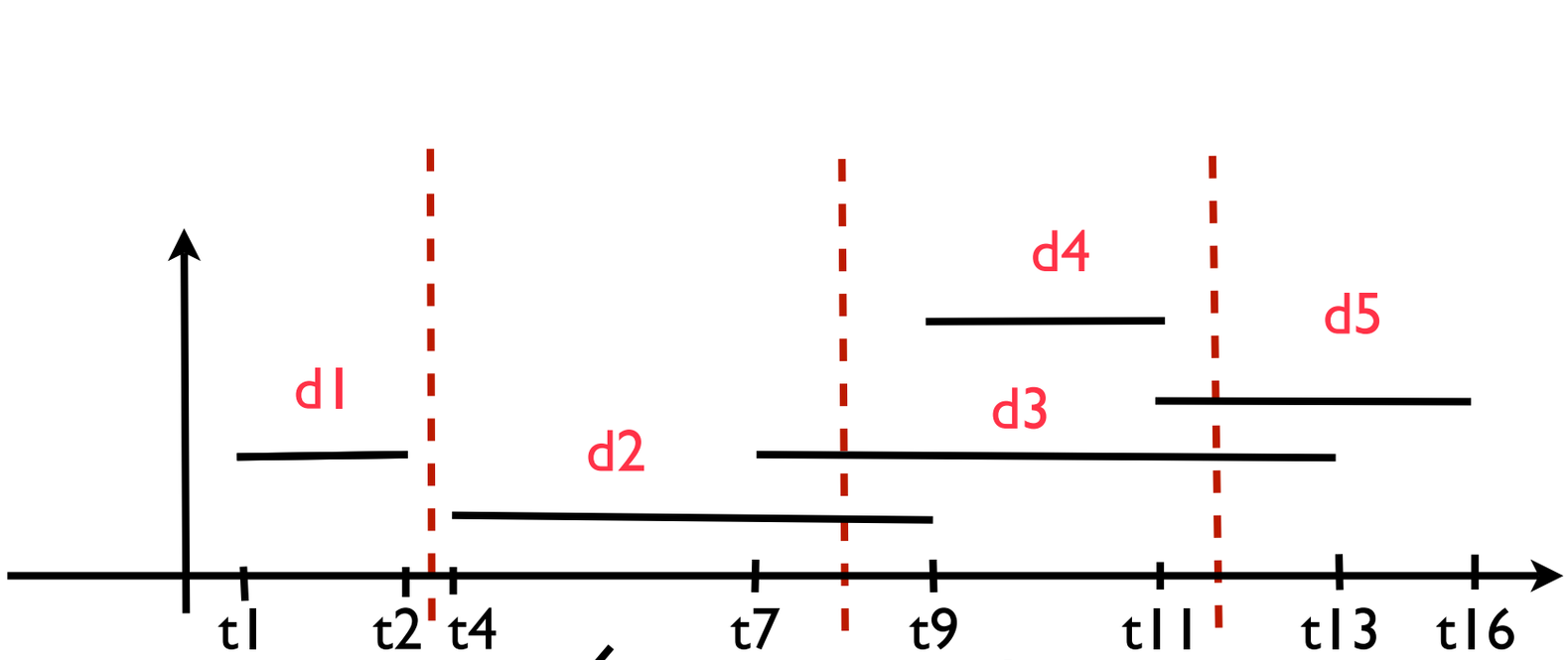


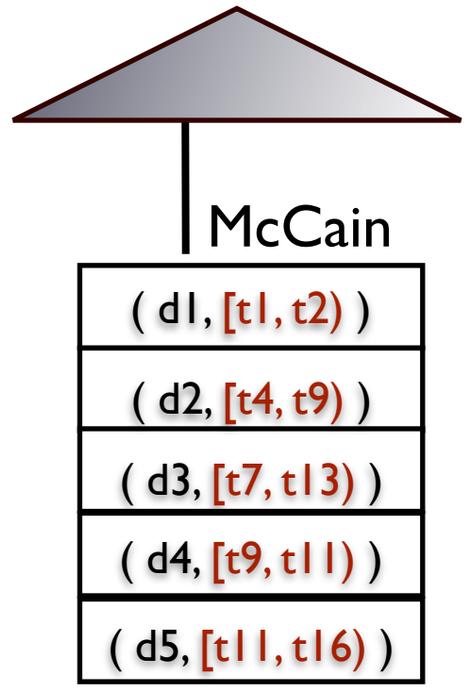


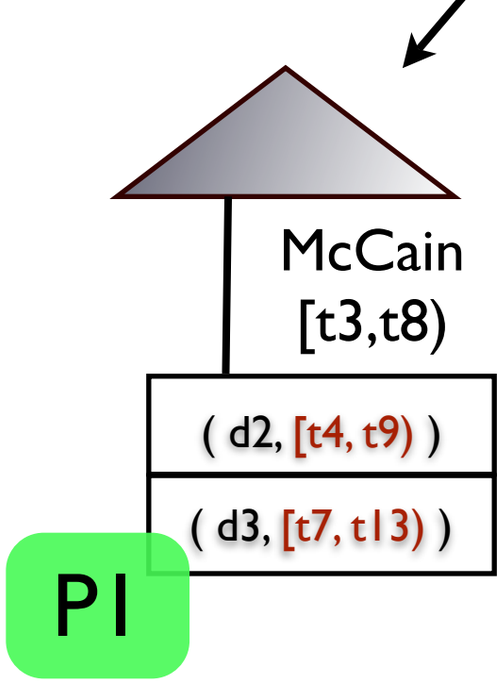


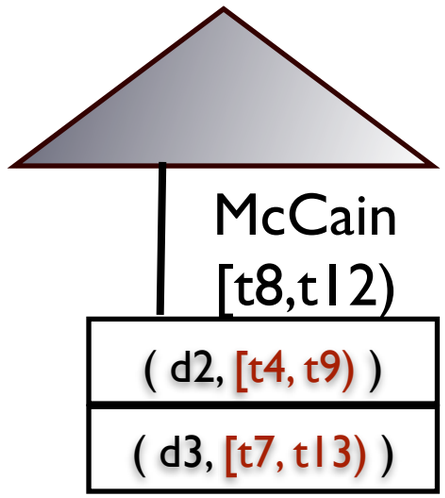


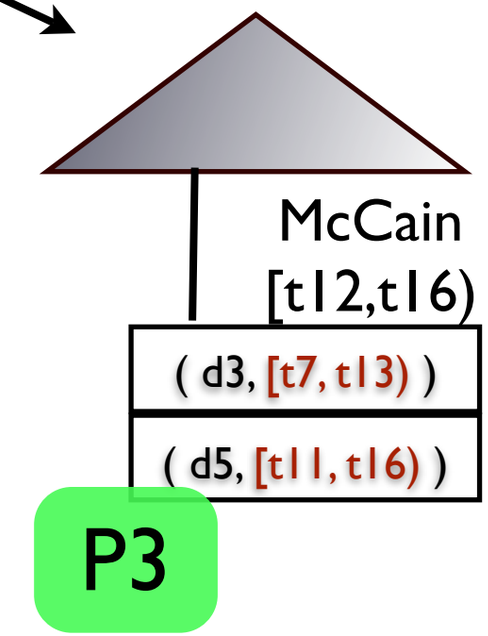
Index Partitioning



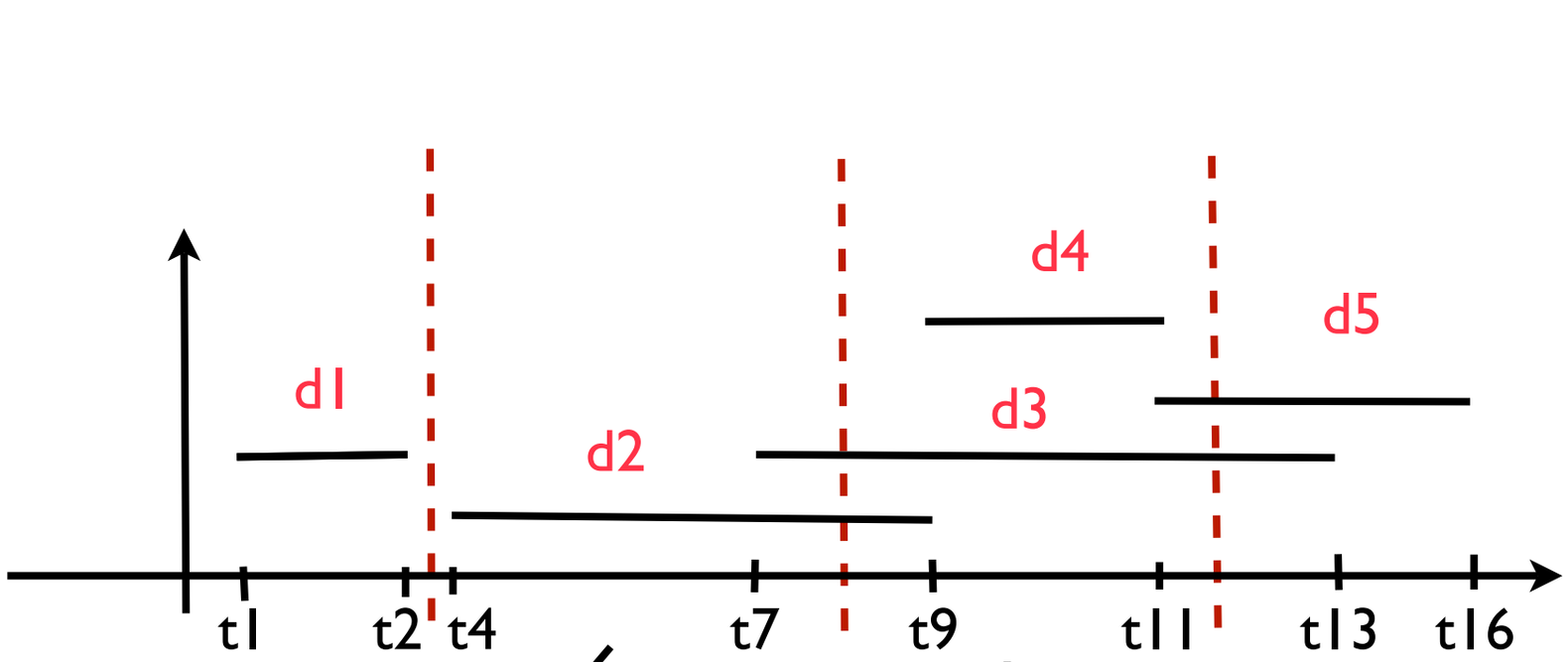


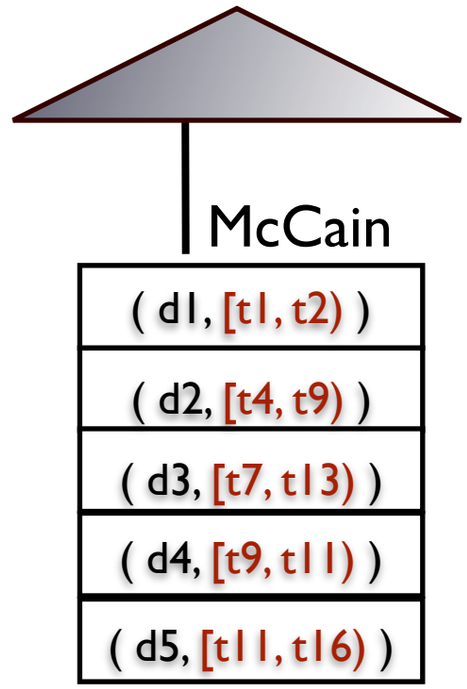


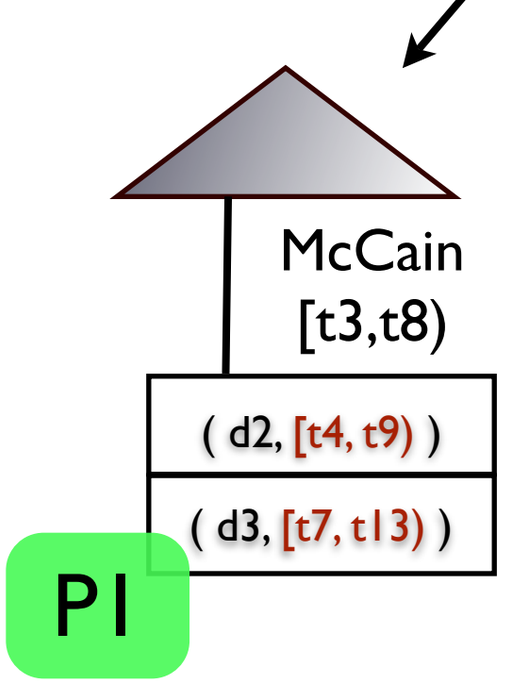


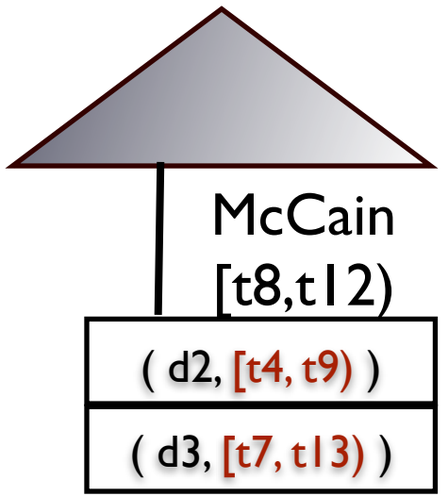


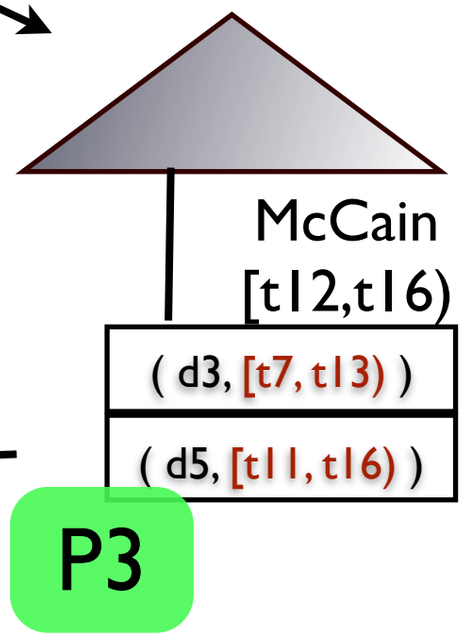
Index Partitioning



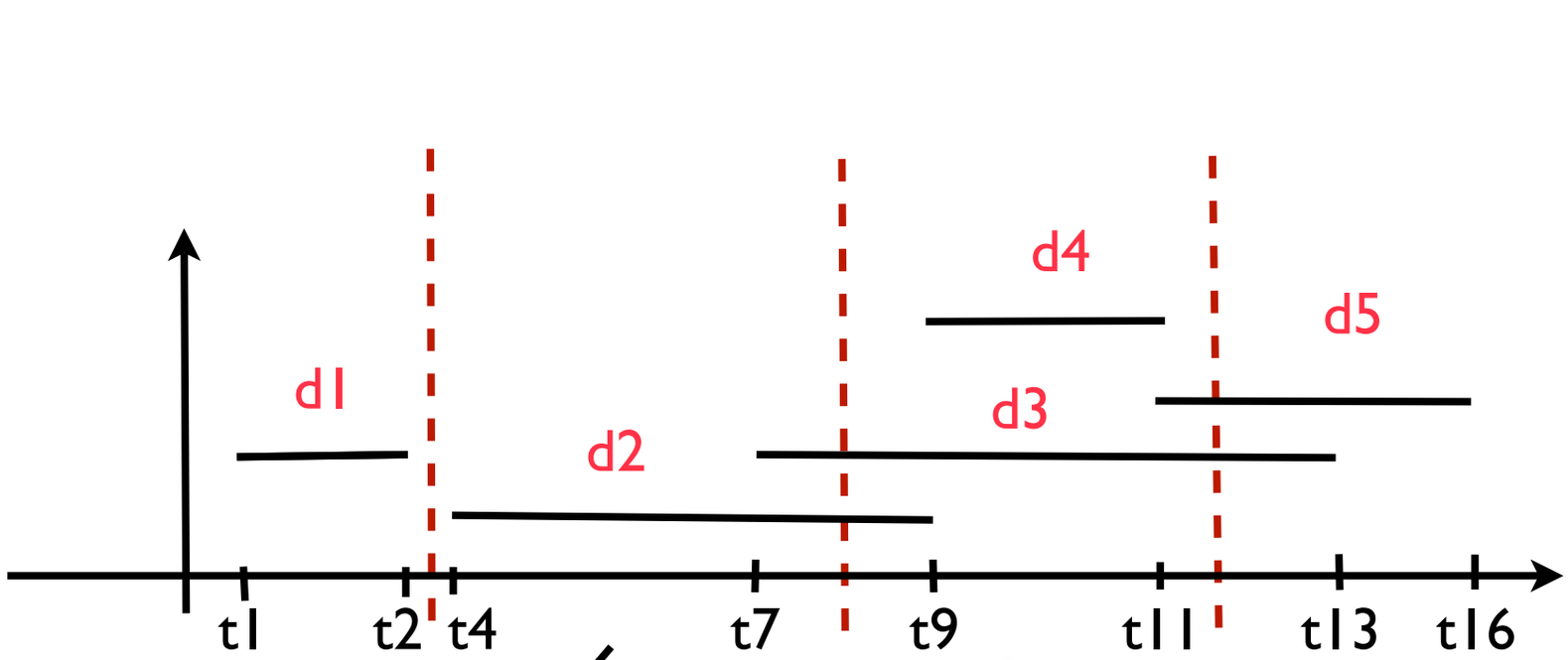


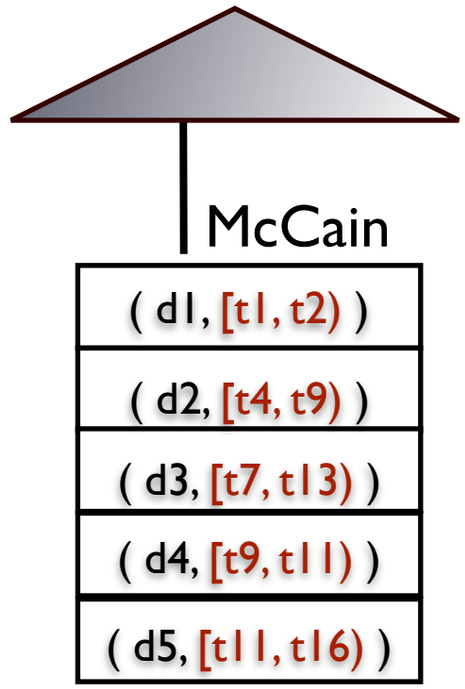


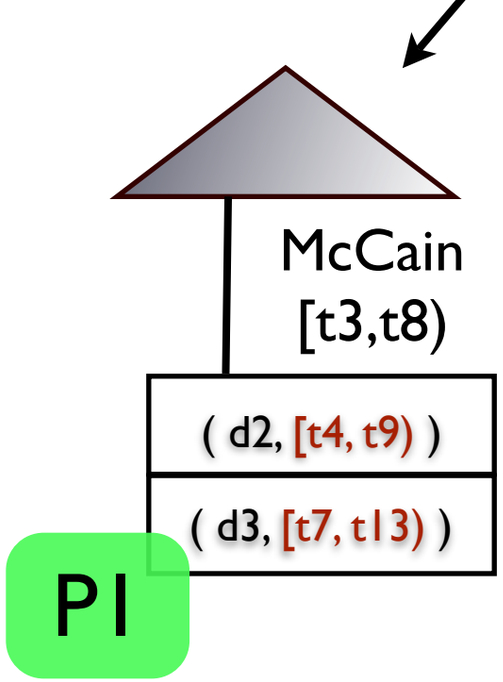


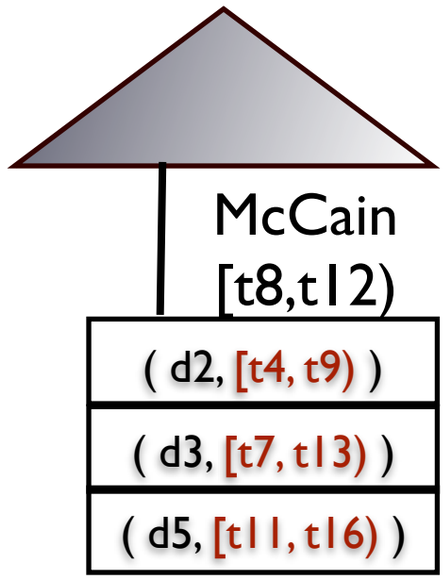


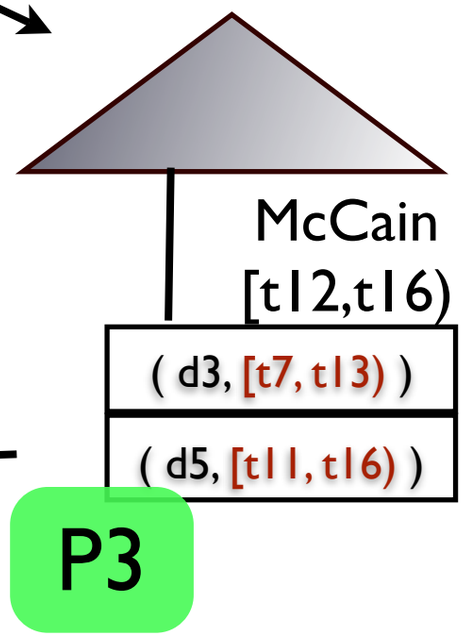
Index Partitioning



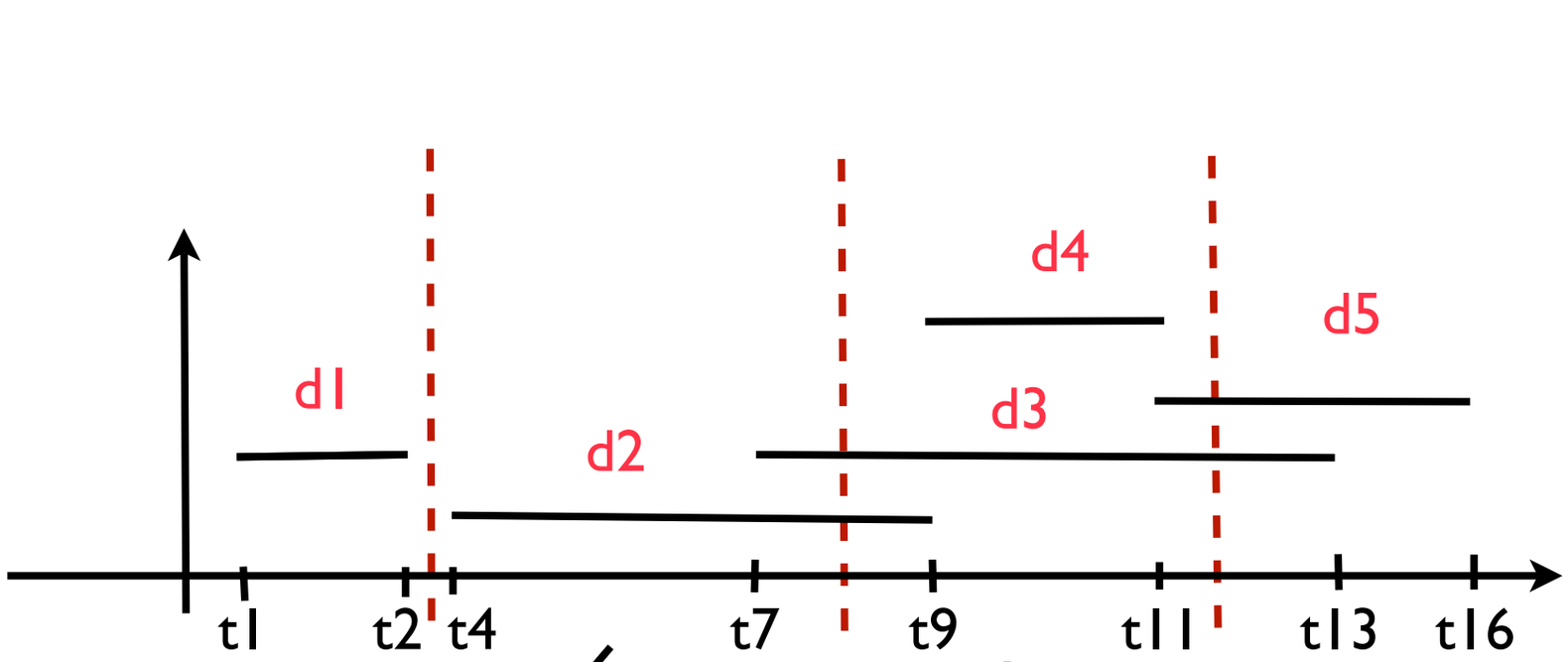


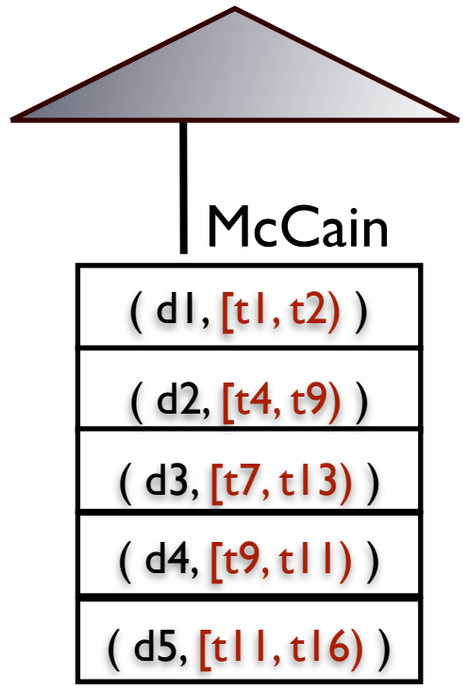


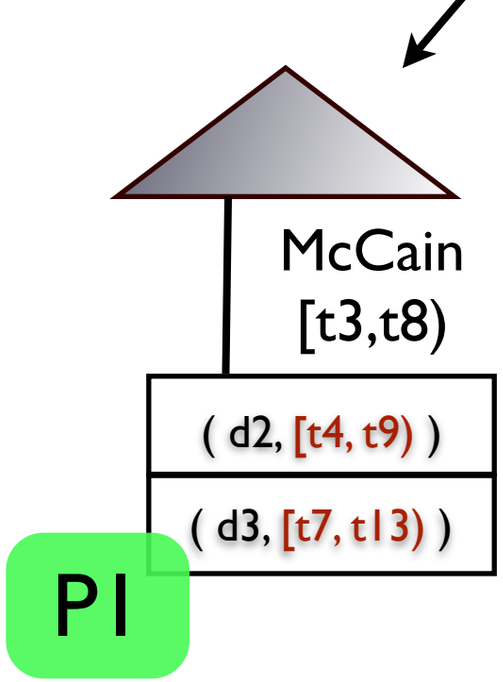


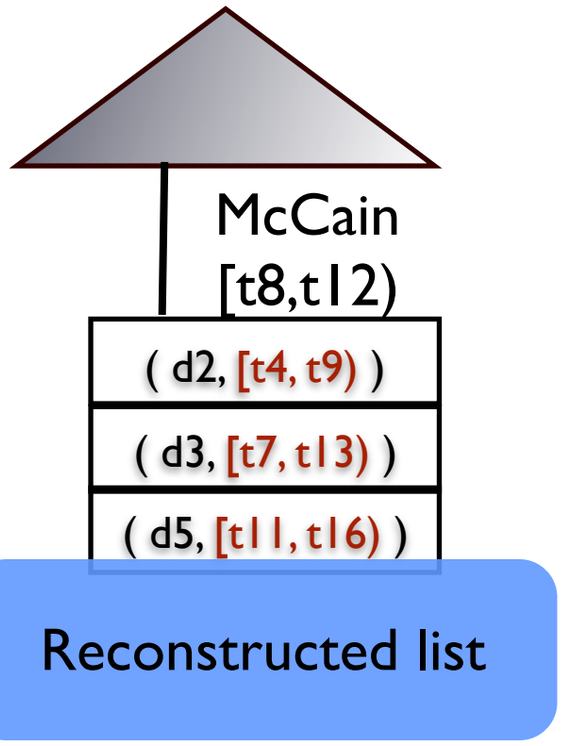


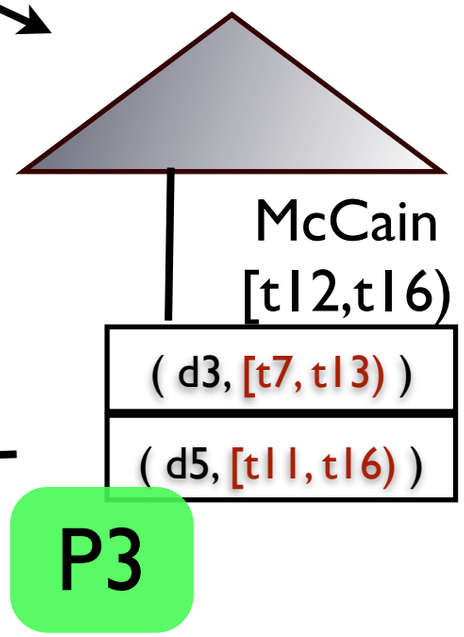
Index Partitioning





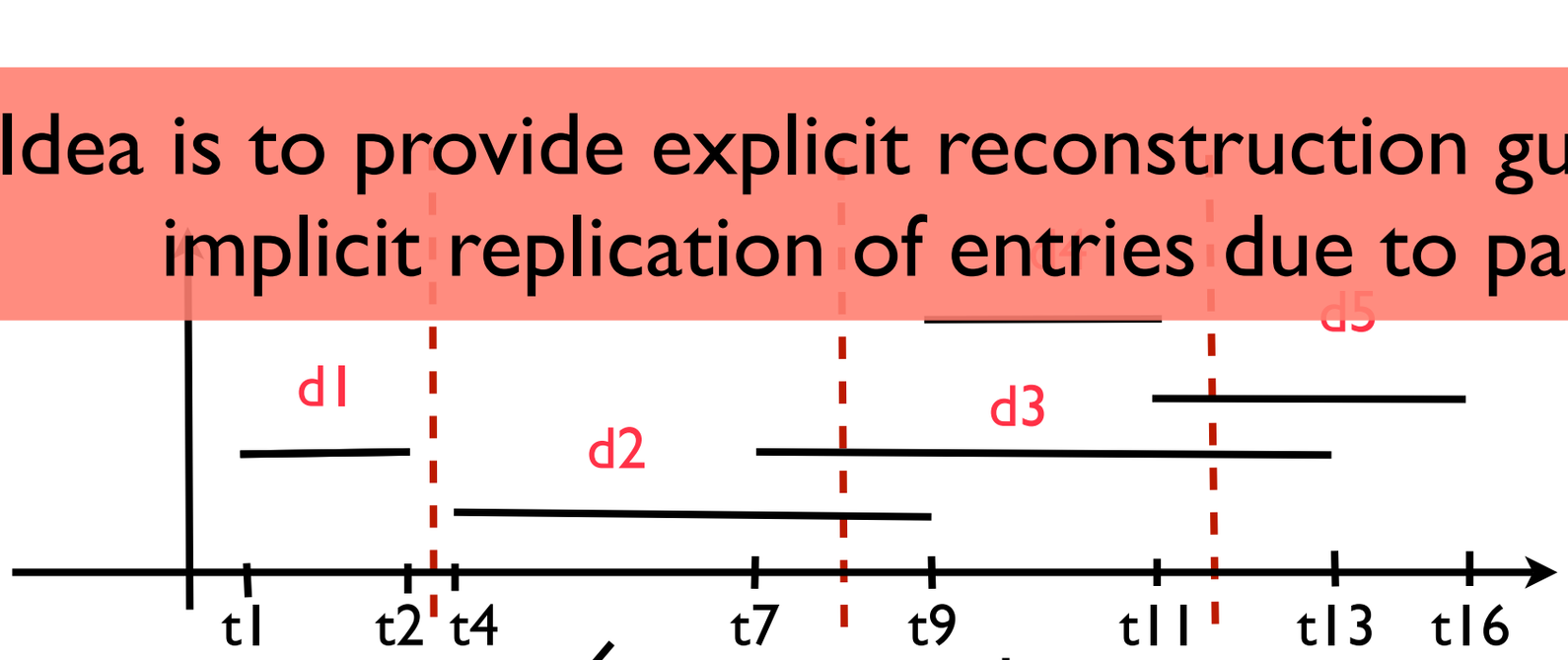


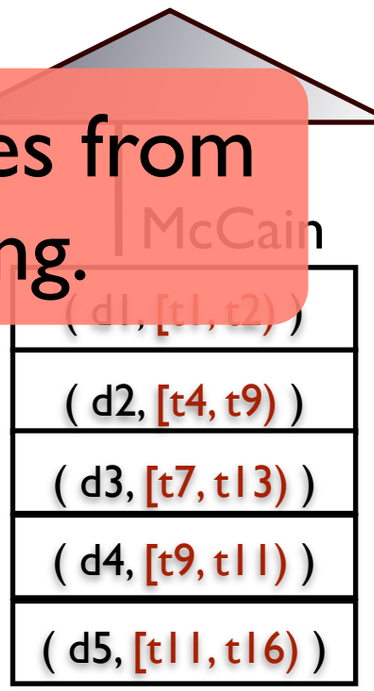


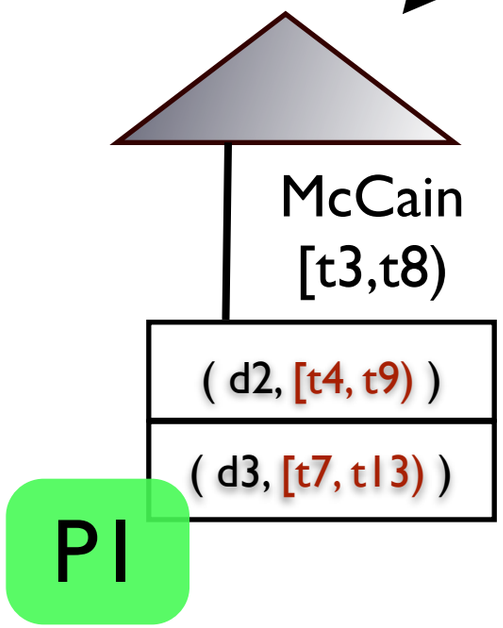


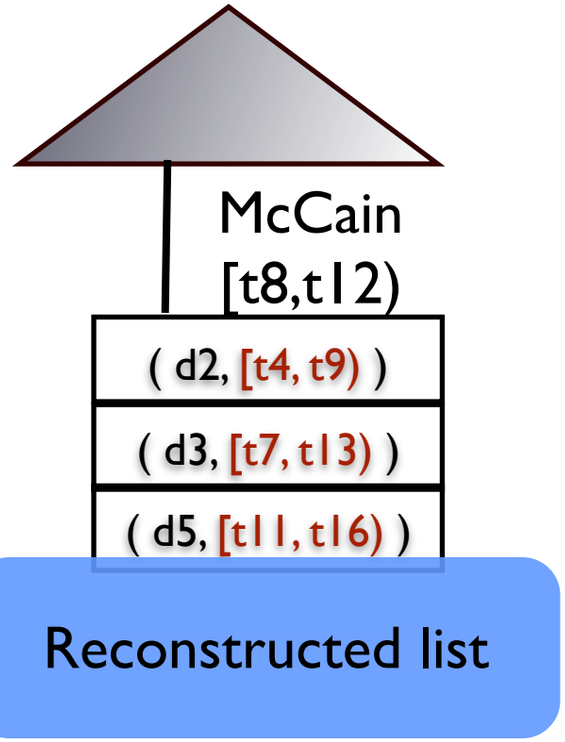
Index Partitioning

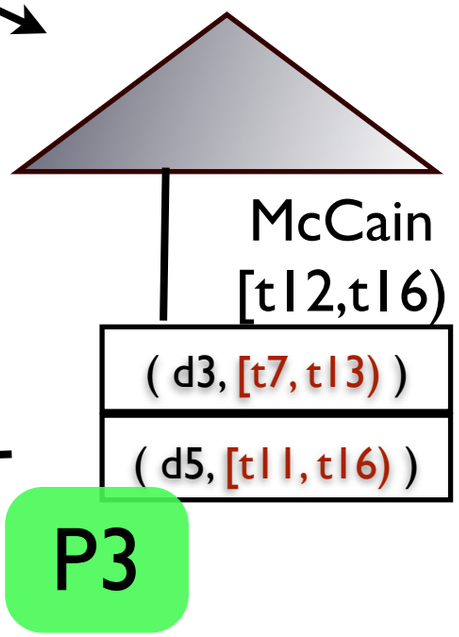
Idea is to provide explicit reconstruction guarantees from implicit replication of entries due to partitioning.





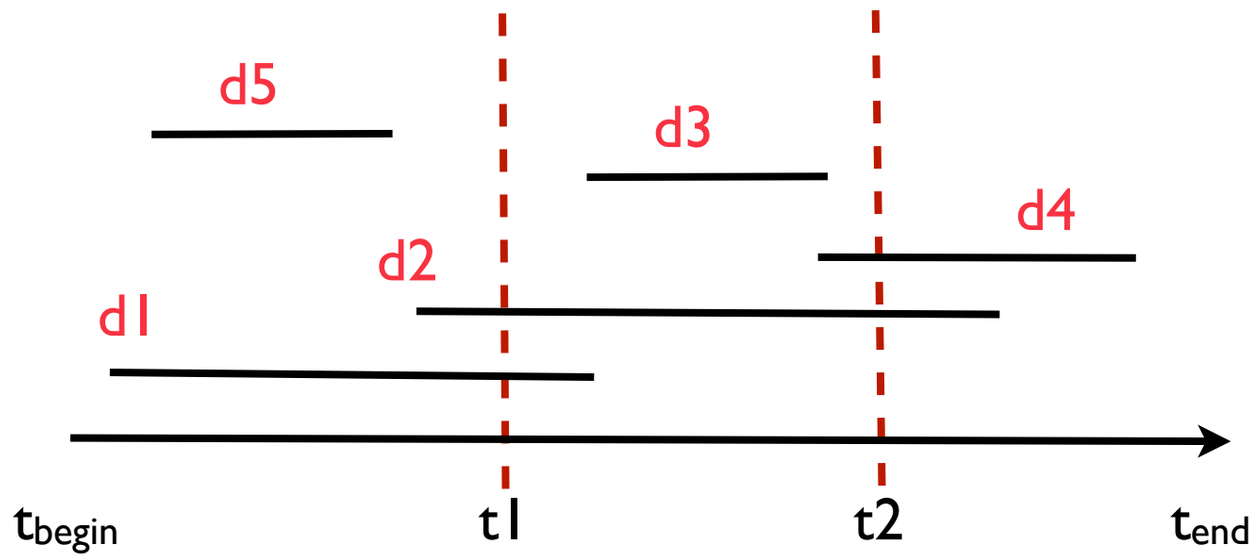






Live Entries

- Live entries in a partition could be classified into left, right, strikethrough and subsumed.



left:[t1, t2] = {d1}

right:[t1, t2] = {d4}

str:[t1, t2] = {d2}

sub:[t1, t2] = {d3}

N_{t1}= {d1, d2} N_{t2} = {d2, d4}

L_v: [t1, t2) = {d1, d2, d3, d4}

L_v = {d1, d2, d3, d4, d5}

Live Entries in a Partition

- Live entries at a time-point t_i is a set of all entries which exist at t_i.

$$N_{t_i} = \{(d, p, [t_j, t_k)) \in L_v \mid t_j \leq t_i \wedge t_k > t_i\} .$$

- Live entries in a partition L_v: $[t_i , t_j)$:

$$L_v : [t_i, t_j) = \{(d, p, [t_b, t_e)) \in L_v \mid t_b < t_j \wedge t_e > t_i\} .$$

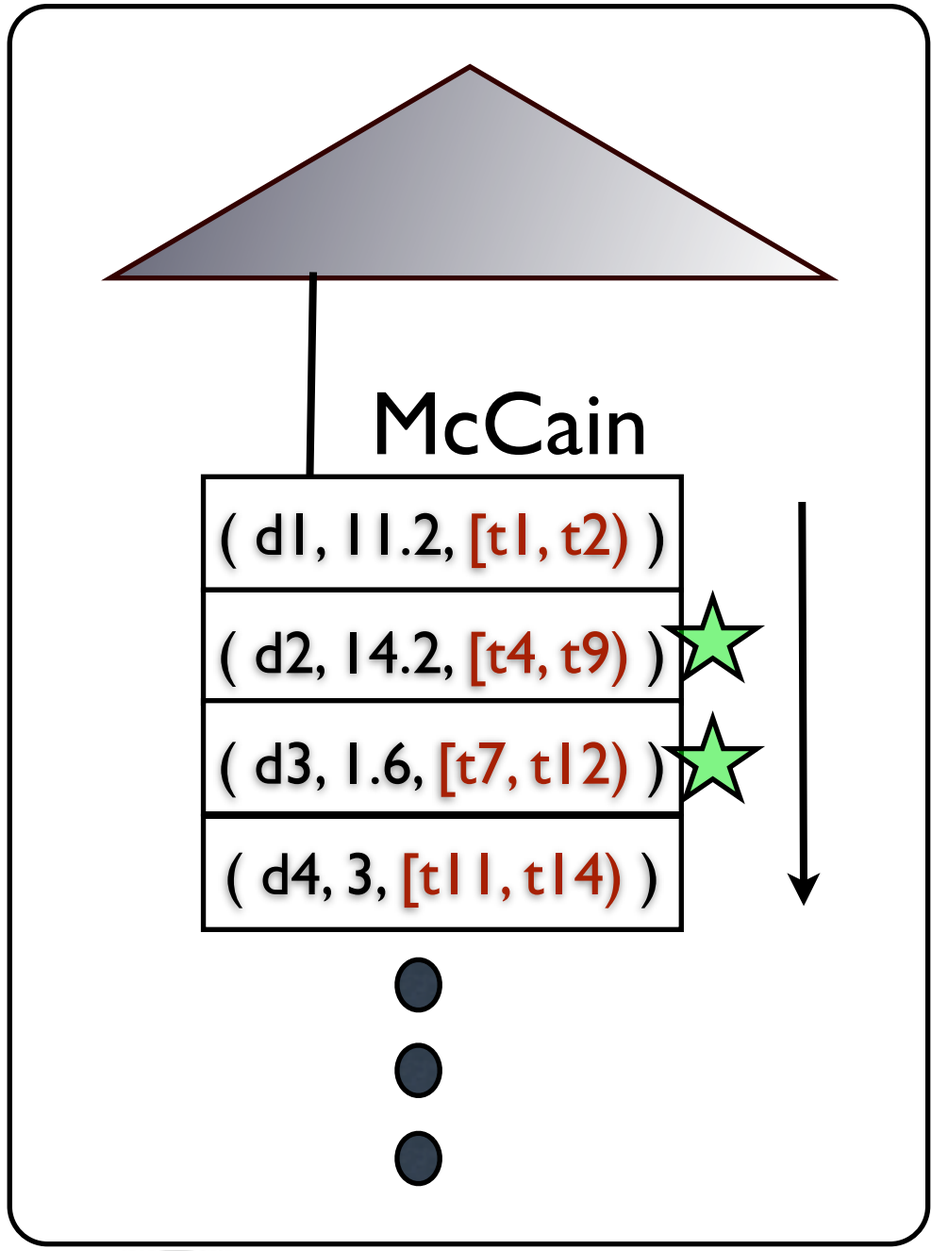
(d, p, [t_b , t_e)) is the posting for the document **d** which begins at **t_b** and ends at **t_e**.
L_v denotes all the documents in the collection.

Backup Slides

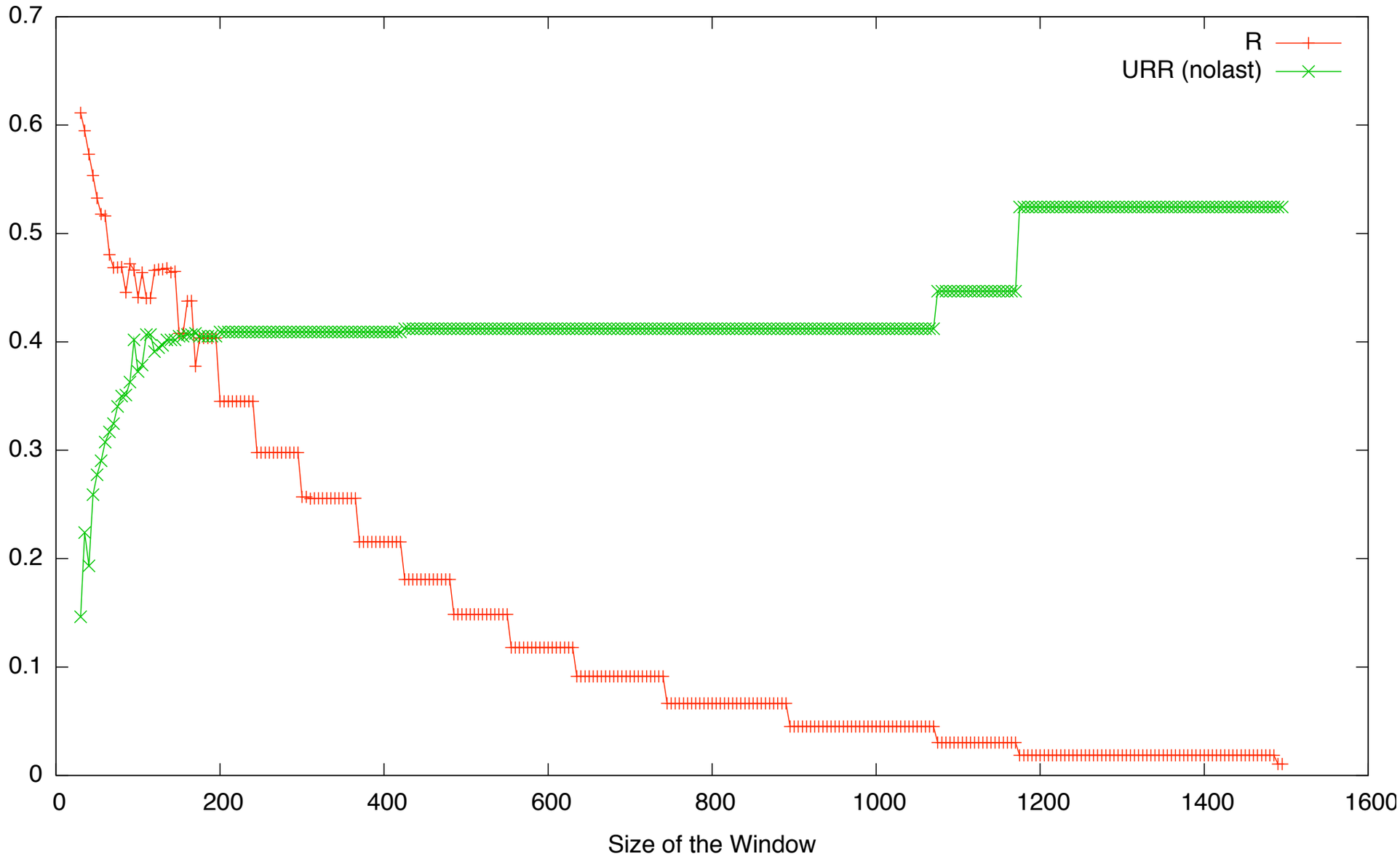
- Everlast as Web Archival system
- Other lifetime analysis results about Wikipedia

Time-travel queries

- Time-travel queries are keyword queries with a temporal context. E.g "McCain @ t5"
- The conventional inverted index structures are extended by addition of validity time-interval
- "McCain @ t8" => d2 , d3
- Every version is considered as a new entry resulting in long lists.



Results



Values of R and Min URR for different window values on wikipedia data