Data Currency in Replicated DHTs

Reza Akbarinia Esther Pacitti Patrick Valduriez

Presented by John Harris
CS 755, Fall 2013

Background

- P2P systems are great for scalability, availability

- Early systems (Gnutella, KaaZa) relied on query flooding
    - Many peers, loosely aware of each other

- More structured systems (CAN, Chord, Pastry) employ DHTs
    - O(log*n*) query routing performance

Complications

- *Churn* introduces problems
  - Nodes arrive/depart unpredictably, data becomes unavailable
  - **Solution:** *Replicate* data across many nodes to ensure availability

- New Problem
  - Which peers have most recent version of data?

- By P2P definition, no centralized authority
  - How to define “most recent”?

Formally, the problem can be defined as follows. Given a key $k \in K$, let R_k be the set of replicas such that for each $r \in R_k$, the pair (k, r) is stored at one of the peers of the DHT. Our goal is to return efficiently an $r \in R_k$ which is current, *i.e.* reflects the latest update.

How to find the “freshest” data...

Data Currency in Replicated DHTs

Reza Akbarinia Esther Pacitti Patrick Valduriez

… among many duplicate peers...

Presented by John Harris
CS 755, Fall 2013

… without any central authority.

Proposed Solution

UMS + KTS

Update Management Service

Improves data availability through replication using set of pairwise independent hash functions ***H***

Each DHT can have unique ***H***.

Notice: size of set ***H*** determines degree of replication (and therefore data availability).

ADHT provides efficient means of determining

$$presp(k,h) \rightarrow \overset{\downarrow}{resp(k,h)} \rightarrow \underset{\uparrow}{nresp(k,h)}$$

Will be used during key/timestamp ‘handoff’

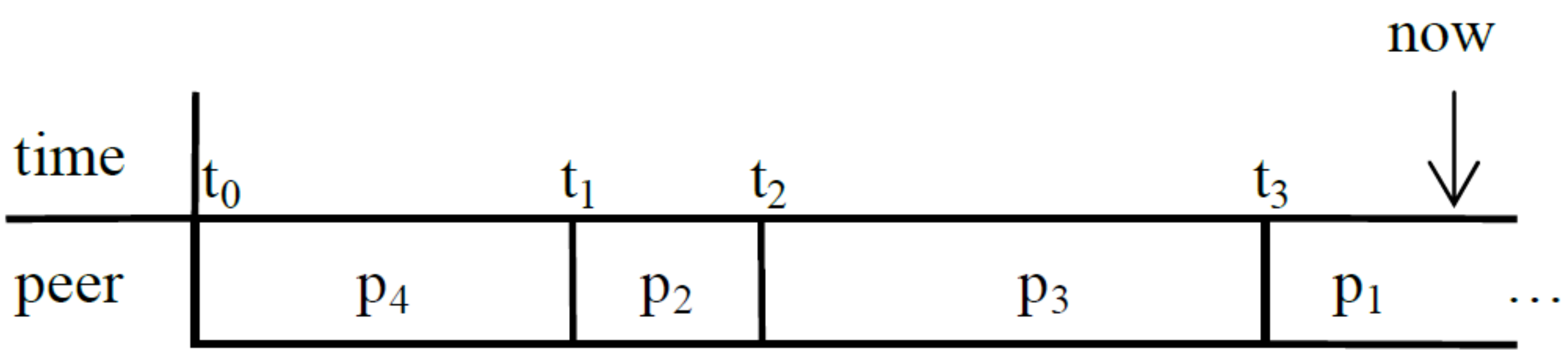
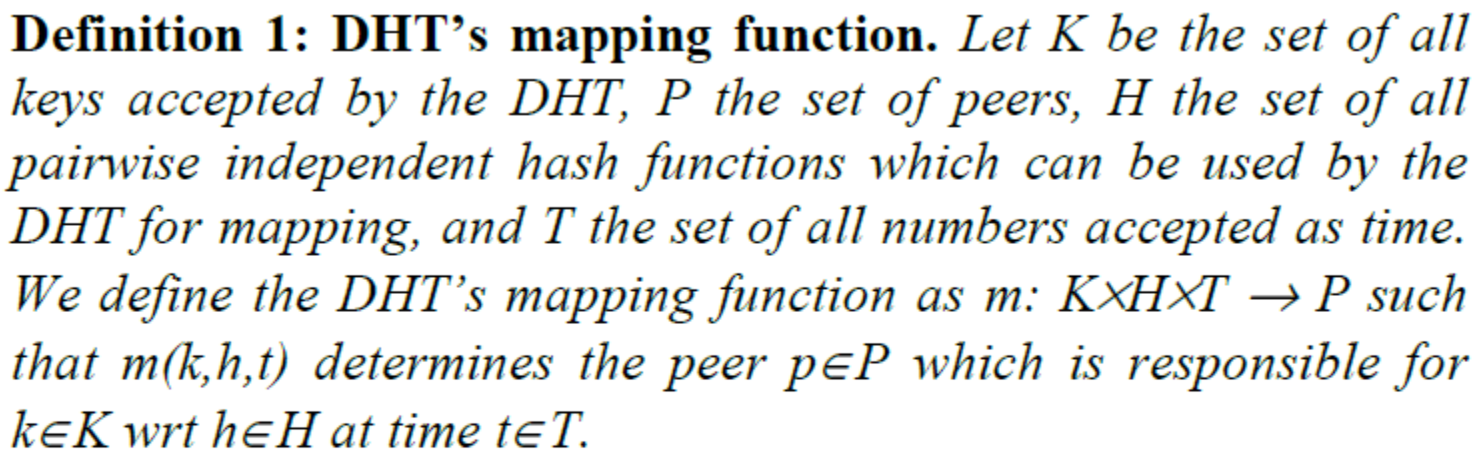
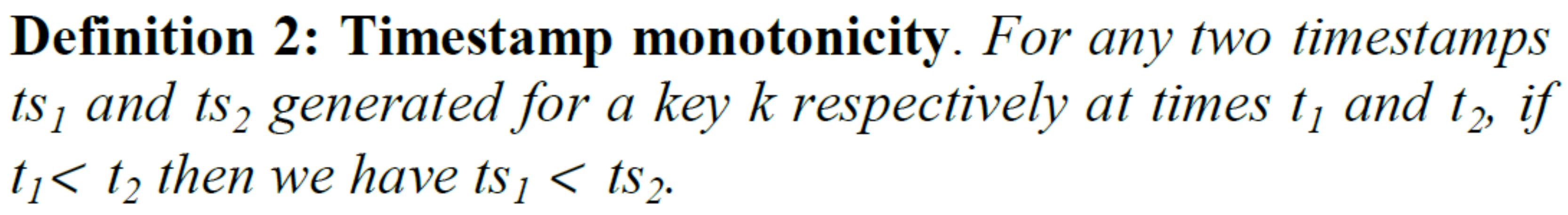


Figure 1. Example of peers’ responsibilities



Definition 1: DHT's mapping function. *Let K be the set of all keys accepted by the DHT, P the set of peers, H the set of all pairwise independent hash functions which can be used by the DHT for mapping, and T the set of all numbers accepted as time. We define the DHT's mapping function as m: $K \times H \times T \rightarrow P$ such that m(k,h,t) determines the peer $p \in P$ which is responsible for $k \in K$ wrt $h \in H$ at time $t \in T$.*

I.e. each (key, data) pair receives logical timestamp and is distributed using each hash function ***h*** in ***H*** to appropriate set of peer nodes



Definition 2: Timestamp monotonicity. *For any two timestamps ts_1 and ts_2 generated for a key k respectively at times t_1 and t_2, if $t_1 < t_2$ then we have $ts_1 < ts_2$.*

But how is this achieved?

The Crux of the Paper

Key-Based Timestamping

Distributing responsibility for generating timestamps mirrors distributing responsibility for storing data.

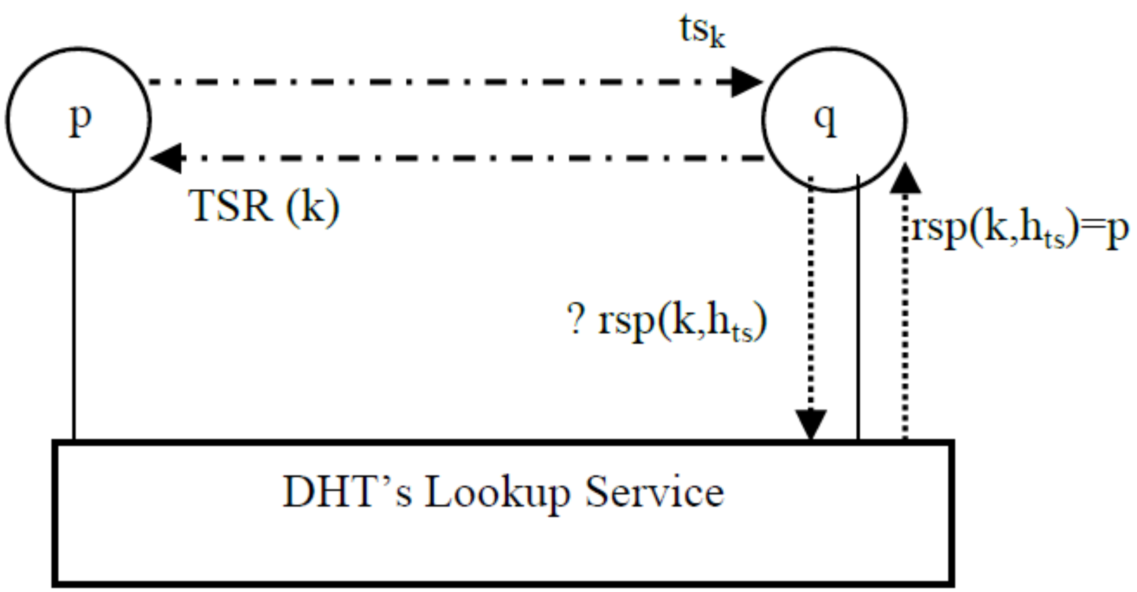


Figure 3. Example of timestamp generation

Important Assumption

“If $rsp(k,h_{ts})$ leaves or fails, the DHT detects the absence (e.g. by frequently sending “ping” messages from each peer to its neighbours). … another peer automatically becomes responsible for timestamping *k*.”

Generating Monotonicity

1) Local timestamp counter $\boldsymbol{c}_{p,k}$ for key ***k*** at peer ***p*** is incremented every timestamp request.

2) $\boldsymbol{c}_{p,k}$ is initialized to the last value of $\mathbf{c}_{q,k}$ where ***q*** is the last peer to gave generated a timestamp for ***k***

But how is #2 achieved?

*Monotonicity only applies to timestamps generated for the same key.

Counter Initialization

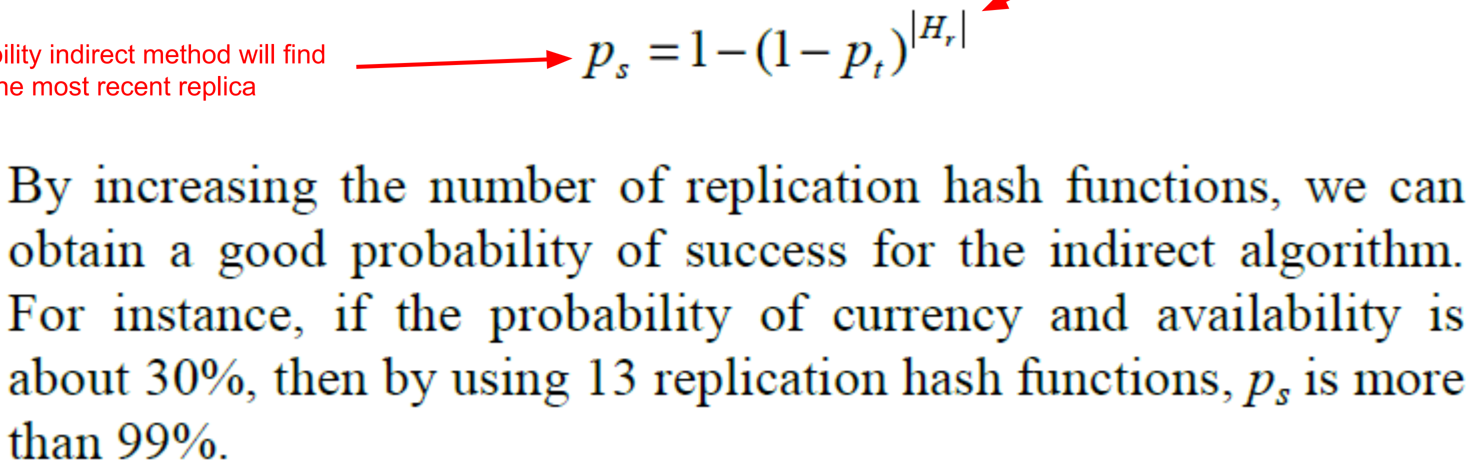
Direct: When a peer leaves gracefully, it transfers all counters to the next responsible peer. Efficient, simple.

Indirect: If old peer fails unexpectedly, newly responsible peer retrieves all replicas for *k* an initializes counter to most recent timestamp. Requires multiple lookups, not guaranteed correct.

What are the odds?

Number of replicating hash functions

Probability indirect method will find the most recent replica



$$p_s = 1 - (1 - p_t)^{|H_r|}$$

By increasing the number of replication hash functions, we can obtain a good probability of success for the indirect algorithm. For instance, if the probability of currency and availability is about 30%, then by using 13 replication hash functions, p_s is more than 99%.

Greater availability leads to greater probability of successful indirect initialization, but never 100% guaranteed

For that <1% likely error...

Recovery: Original responsible failed node returns, contacts newly responsible node, performs direct counter transfer. Current node double-checks, fixes own counters, and reinserts any erroneous *(key,data, timestamp)* records.

Periodic Inspection: If a newly responsible node never hears back from its predecessor, it periodically checks what timestamps our already in the DHT records and updates its internal counters if necessary.

Responsibility Loss (Un)Aware DHTs

- In RLA DHTs, key and timestamp responsibility is transferred at handoffs
  - Extra efficient b/c new peers tend to be neighbours

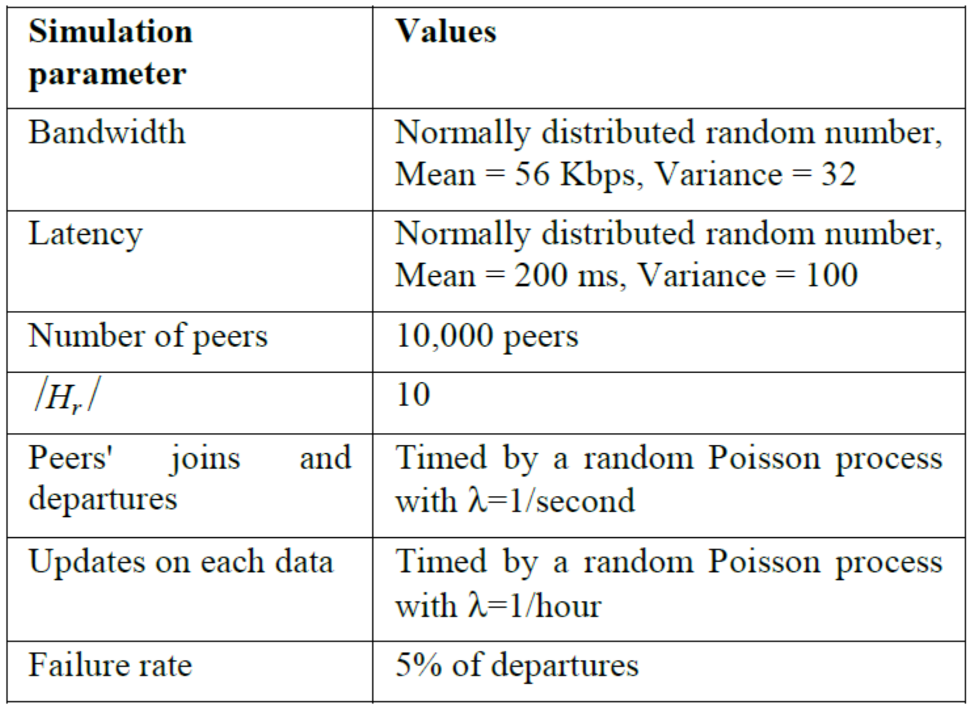
- In an RLU DHT, timestamp consistency can be achieved by forcing every peer to reacquire responsibility for a timestamp each time it generates one
  - I.e. go through the *indirect initialization* procedure every time
  - It’s expensive but compensates for otherwise “silent” handoffs

Performance Evaluation

Simulation Conditions

- Implemented using modified Chord DHT
- Baseline 64-node cluster, scaled up with 10,000 node SimJava simulation
- Compared against BRICKS project

Table 1. Simulation parameters



Simulation parameter	**Values**
Bandwidth	Normally distributed random number, Mean = 56 Kbps, Variance = 32
Latency	Normally distributed random number, Mean = 200 ms, Variance = 100
Number of peers	10,000 peers
$/H_r/$	10
Peers' joins and departures	Timed by a random Poisson process with λ=1/second
Updates on each data	Timed by a random Poisson process with λ=1/hour
Failure rate	5% of departures

Data Charts

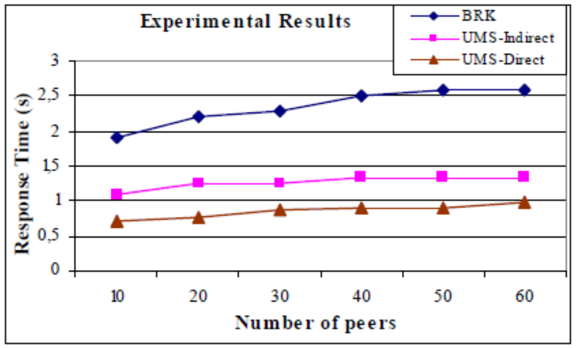


Figure 6. Response time vs. number of peers

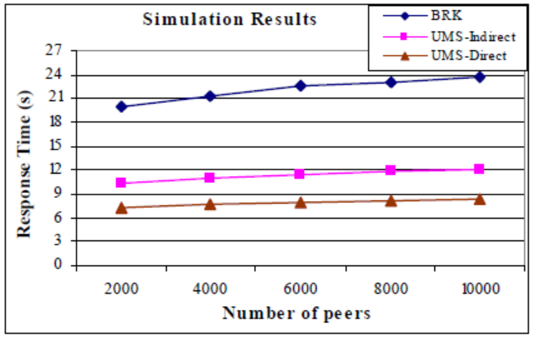


Figure 7. Response time vs. number of peers

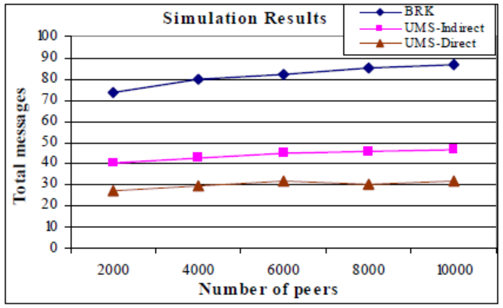


Figure 8. Communication cost vs. number of peers

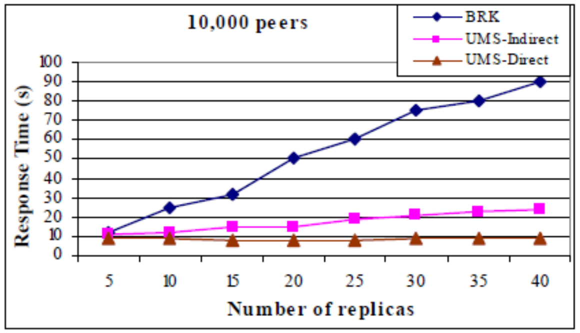


Figure 9. Response time vs. number of replicas

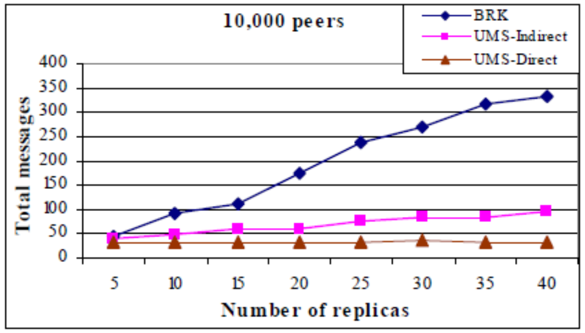


Figure 10. Communication cost vs. number of replicas

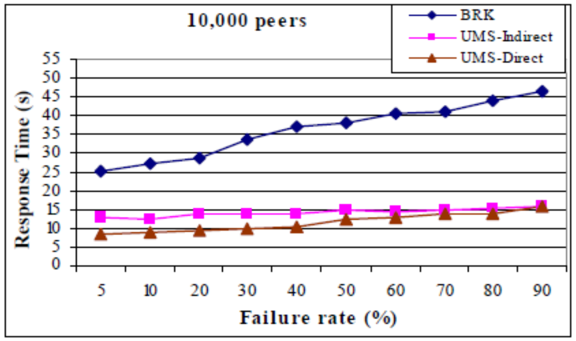


Figure 11. Response time vs. failure rate

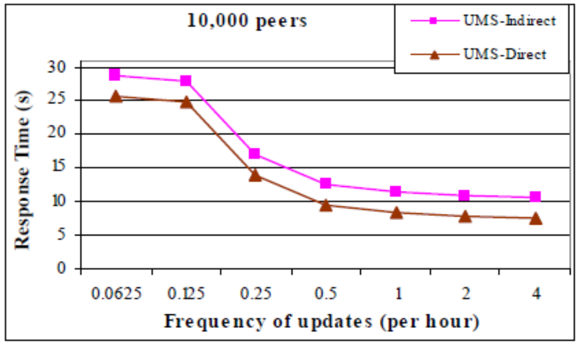


Figure 12. Response time vs. frequency of updates

Related Work (circa 2007)

- PGrid: concurrent updates -> inconsistency
- Freenet: absent peers are never updated
- CFS, Past, OceanStore: immutable data only
- BRICKS: non-unique version numbers -> conflicts

Related Work (circa 2010)

“Continuous Timestamping for Efficient Replication Management in DHTs”

- Same authors extends timestamp *monotonicity* property with *continuous* (no gaps) property
- Improves efficiency and fault tolerance using “replica holder groups”
-

Questions/Discussion

Discussion/Questions

- What is a potential weakness of the timestamp “recovery” and “periodic inspection” algorithms proposed in the paper?

- Consider: If there’s only ever one peer responsible for timestamping *k* at time *t,* this paper achieves “dynamic centralized authority”

- Can you think of a scenario where *churn* would be a **good** thing?