We address the problem of data reconciliation in peer-to-peer (P2P) collaborative applications. We propose P2P-LTR (Logging and Timestamping for Reconciliation) which provides P2P logging and timestamping services for P2P reconciliation over a distributed hash table (DHT). While updating at collaborating peers, updates are timestamped and stored in a highly available P2P log. During reconciliation, these updates are retrieved in total order to enforce eventual consistency.

**Abstract**

We address the problem of data reconciliation in peer-to-peer (P2P) collaborative applications. We propose P2P-LTR (Logging and Timestamping for Reconciliation) which provides P2P logging and timestamping services for P2P reconciliation over a distributed hash table (DHT). While updating at collaborating peers, updates are timestamped and stored in a highly available P2P log. During reconciliation, these updates are retrieved in total order to enforce eventual consistency.

**Motivations**

- Killer application: XWiki P2P
  - A Wiki application based on P2P technology
  - scalable to millions of users
- Multi-Master replication
- Challenge: ensuring data consistency
  - Ensure convergence of copies
  - Support asynchronous updates, possibly disconnected

**P2P-LTR Overview**

- P2P-LTR, an extension of KTS* service
  - Distributed continues timestamp generation
  - Publishing operations (patches) in DHT network
- Use of a DHT
  - Efficient data location through put and get primitives
  - Scalability
  - Fault Tolerance through the DHT stabilization layer
  - Dealing with dynamicity (mobility)
  - Simplifying the construction of large-scale distributed applications

**P2P-LTR Model**

Each XWiki document is identified by a key

- **Master-key**
  - succ: Crash recovery
  - gen_Ts: generates Ts and publishes patch

- **Log peer**
  - stores timestamped patches for a key

- **XWiki peer**
  - application

- **DHT**
  - (OpenChord)

**Reconciliation using P2P-LTR**

Briefly, P2P-LTR performs the following main procedures:
1. Produce a tentative patch (e.g. after each update in a XWiki page)
2. Generate continuous timestamps
3. Validate the timestamp of the tentative patch (by considering other updaters) and retrieve stored patches if necessary
4. Replicate the new patch at Log peers

**Prototype**

- DHT management
- XWiki document management
- Patch management
- Patch & Timestamp informations
- Network Monitoring
- LTR & DHT configurations

**Demonstration Scenarios**

- Continuous timestamp generation
- Concurrent patch publishing
- Master-key peer departures
- New Master-key peer joining

**Acknowledgements**

This work is partially funded by French ANR XWiki Concerto project and European STREP Grid4All project.