Rollerchain: a DHT for Efficient Replication
IEEE NCA’13

João Paiva, João Leitão, Luís Rodrigues

Instituto Superior Técnico / Inesc-ID, Lisboa, Portugal

August 22, 2013
Outline

Introduction

Our approach

Evaluation

Conclusions
Distributed Hash Tables are structured overlays where nodes organize into a predefined topology that supports routing.

DHTs allow for scalable key-value storage.
Motivation

- In dynamic environments, replication is paramount to maintaining data.

- However, predefined topologies are expensive to maintain in dynamic environments (churn).
- DHTs do not handle churn as well as unstructured networks.
Motivation

▶ In dynamic environments, replication is paramount to maintaining data.

▶ However, predefined topologies are expensive to maintain in dynamic environments (churn).
▶ DHTs do not handle churn as well as unstructured networks.
Main Approaches to DHT replication

1. Neighbour Replication

2. Multi-Publication
Neighbour Replication

Each node replicates its data on its $R$ closest neighbours

- Good control on replication degree
- Simple to locate replicas

- Expensive replication: data is moved to respect topological constraints
- Not resilient under churn: each node acts on its own
- Poor load balancing: no active mechanisms to balance load
Neighbour Replication

Each node replicates its data on its $R$ closest neighbours

- Good control on replication degree
- Simple to locate replicas
- Expensive replication: data is moved to respect topological constraints
- Not resilient under churn: each node acts on its own
- Poor load balancing: no active mechanisms to balance load
Neighbour Replication: operation
Neighbour Replication: operation
Neighbour Replication: operation
Neighbour Replication: operation
Neighbour Replication: operation
Multi-Publication

Each object is attributed $R$ different identifiers to be stored by $R$ different nodes.

- Better load balancing
- Reduced correlated failures
- Expensive overlay maintenance: each object has a different set of replicas
- Expensive replication: data is moved to respect topological constraints
- Not resilient under churn: each node acts on its own
Each object is attributed $R$ different identifiers to be stored by $R$ different nodes.

- Better load balancing
- Reduced correlated failures
- Expensive overlay maintenance: each object has a different set of replicas
- Expensive replication: data is moved to respect topological constraints
- Not resilient under churn: each node acts on its own
Current DHTs

Based on structured networks

Characterized by:

- Nodes with fixed positions in the overlay
- Static replication degree
- Poor performance under churn
Main challenges

Challenges:

1. Increase churn resilience
2. Minimize replication costs
3. Improve load balancing
Outline

Introduction

Our approach

Evaluation

Conclusions
Our approach: Architecture overview

- Ring-based overlay: Composed of virtual nodes
Our approach: Architecture overview

Ring-based overlay: Composed of virtual nodes
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: Dynamic topology overview
Our approach: beating the challenges

1. Increase churn resilience: unstructured networks
2. Minimize replication costs: variable replication degree
3. Improve load balancing: dynamic key distribution
Our approach: beating the challenges

1. **Increase churn resilience**: unstructured networks
2. Minimize replication costs: variable replication degree
3. Improve load balancing: dynamic key distribution
Increasing churn resilience

- Ring maintained through gossip mechanisms
Increasing churn resilience

- Gossip to keep virtual node membership up-to-date
Increasing churn resilience

- Gossip to trade connections between virtual nodes
Increasing churn resilience
Increasing churn resilience
Increasing churn resilience
Increasing churn resilience
Increasing churn resilience
Our approach: beating the challenges

1. Increase churn resilience: unstructured networks
2. **Minimize replication costs:** variable replication degree
3. Improve load balancing: dynamic key distribution
Minimizing replication costs: node failure

- Variable replication degree: No data movement on failure
Minimizing replication costs: node failure

- Variable replication degree: No data movement on failure
Minimizing replication costs: node failure

- Variable replication degree: No data movement on failure
Minimizing replication costs: node join

- Nodes can select where to join: may join recently-failed virtual nodes
Minimizing replication costs: node join

- Nodes can select where to join: may join recently-failed virtual nodes
Minimizing replication costs: node join

- Nodes can select where to join: may join recently-failed virtual nodes
Minimizing replication costs: node join

- New nodes can replace failed nodes: Blue’s data was moved only once and never discarded
Minimizing replication costs: node join

- New nodes can replace failed nodes: Blue’s data was moved only once and never discarded
Minimizing replication costs: node join

- New nodes can replace failed nodes: Blue’s data was moved only once and never discarded
Our approach: beating the challenges

1. Increase churn resilience: unstructured networks
2. Minimize replication costs: variable replication degree
3. **Improve load balancing**: dynamic key distribution
Improving replication costs: creating dynamic key distribution

- Virtual nodes store a number of keys proportional to their size. Blue’s data is split proportionally by its children.
Improving replication costs: creating dynamic key distribution

- Virtual nodes store a number of keys proportional to their size. Blue’s data is split proportionally by its children.
Improving replication costs: creating dynamic key distribution

- Virtual nodes store a number of keys proportional to their size: Blue’s data is split proportionally by its children.
Outline

Introduction

Our approach

Evaluation

Conclusions
Experimental settings

- Overlay simulation in Peersim
- 100K Nodes
- 50K Keys
- Replication degree = 7
- 5M queries
Churn resilience

Objects reachable (%)
Rollerchain
Neighbour
Multi-Pub

Churn rate
0
20
40
60
80
100

churn=1
churn=10
churn=100

churn rate
0
20
40
60
80
100

Rollerchain
Neighbour
Multi-Pub
Replication costs

<Diagram showing replication costs for different churn rates (churn=1, churn=10, churn=100) for three different replication methods: Rollerchain, Neighbour, and Multi-Pub. The x-axis represents the churn rate, and the y-axis shows the objects moved per node. The diagram illustrates the number of objects moved per node for each replication method at different churn rates.>
Load Balancing

![Bar chart showing the standard deviation of the number of queries processed for Rollerchain, Neighbour, and Multi-Pub.]
Outline

Introduction

Our approach

Evaluation

Conclusions
Conclusions

- DHT based on Virtual Nodes
- Designed with replication in mind
  - Unstructured Networks: Increase churn resilience
  - Variable replication degree: Minimize replication costs
  - Dynamic key distribution: Improve load balancing
Conclusions

- DHT based on Virtual Nodes
- Designed with replication in mind
- Unstructured Networks: Increase churn resilience
- Variable replication degree: Minimize replication costs
- Dynamic key distribution: Improve load balancing
Thank you