Larnaca, Cyprus, October 23, 2014

Consistency in Distributed Systems, and Automatic Network Configuration with Dynamic Churn Prediction

Thomas Bocek, Sebastian Golaszewski, Maxat Pernebayev, Andri Lareida, Burkhard Stiller

Department of Informatics IFI, Communication Systems Group CSG, University of Zürich UZH

bocek@ifi.uzh.ch

Universität Zürich[UZH]

Introduction

Consistency in DHTs

Churn Prediction with DHTs

Why DHT? (1)

- DHT in P2P
  - Very popular use-case: file-sharing, BitTorrent (KAD)
  - DHT/P2P is **much more** than file-sharing
- DHT in IoT
  - A DHT-Based Discovery Service for the Internet of Things, Federica Paganelli and David Parlanti
  - Implementing Secure P2P-ONS, Benjamin Fabian,
  - A distributed control plane for the Internet of Things based on a Distributed Hash Table, Jaime Jiménez Bolonio, Manuel Urueña, Gonzalo Camarillo

Why DHT? (2)

- DHT in IoT
  - Interconnecting smart objects through an overlay networking Architecture, Anastasios Zafeiropoulos, Athanassios Liakopoulos and Panagiotis Gouvas, Harjula, E., Leppänen, T., Ojala, T. and Ylianttila, M.
  - "ADHT: Agent-based DHT Architecture for Constrained Devices," In: IEEE Global Communications Conference (GLOBECOM 2014), December 8-12, Austin, TX, 2014. [Accepted]

→ DHTs is a general purpose mechanism suitable for decentralized, dynamic networks.

Consistency in DHT?

- DHTs have weak consistency semantics!
  - Update can destroy data → put / get
    - Node A: put(X), node C, D stores it
    - Node B: put(Y), node C, D stores it → X is lost!
- P2P/file sharing: can live with it
  - Node A/B are often same
- Other applications may not handle it well
  - E.g., Hive2Hive / PeerBox (distributed Dropbox), user profile from 2 different machines
- Solution: introduce versions in DHTs
  - It's a bit "git" like

Versioned DHT (vDHT)!

- vDHT Goals
  - Offer better consistency semantics
  - Keep put/get operations – don't add complicated API
- "Traditional DHT API"
  - `PutState put(locationKey,data[ttl]);`
  - `Data get(locationKey);`
- vDHT API
  - `PutState put(locationKey, versionKey, data[basedOnKey(s), ttl, prepareFlag]);`
  - `putConfirm(locationKey, versionKey, ttl);`
  - `Data get(locationKey, versionKey);`
  - `List<Data> getLatest(locationKey);`

vDHT Mechanism

- Minimal internal changes
  - HashTable → SortedTable
  - put/get RPC enhanced
  - Table for unconfirmed data / tombstone for deleted data

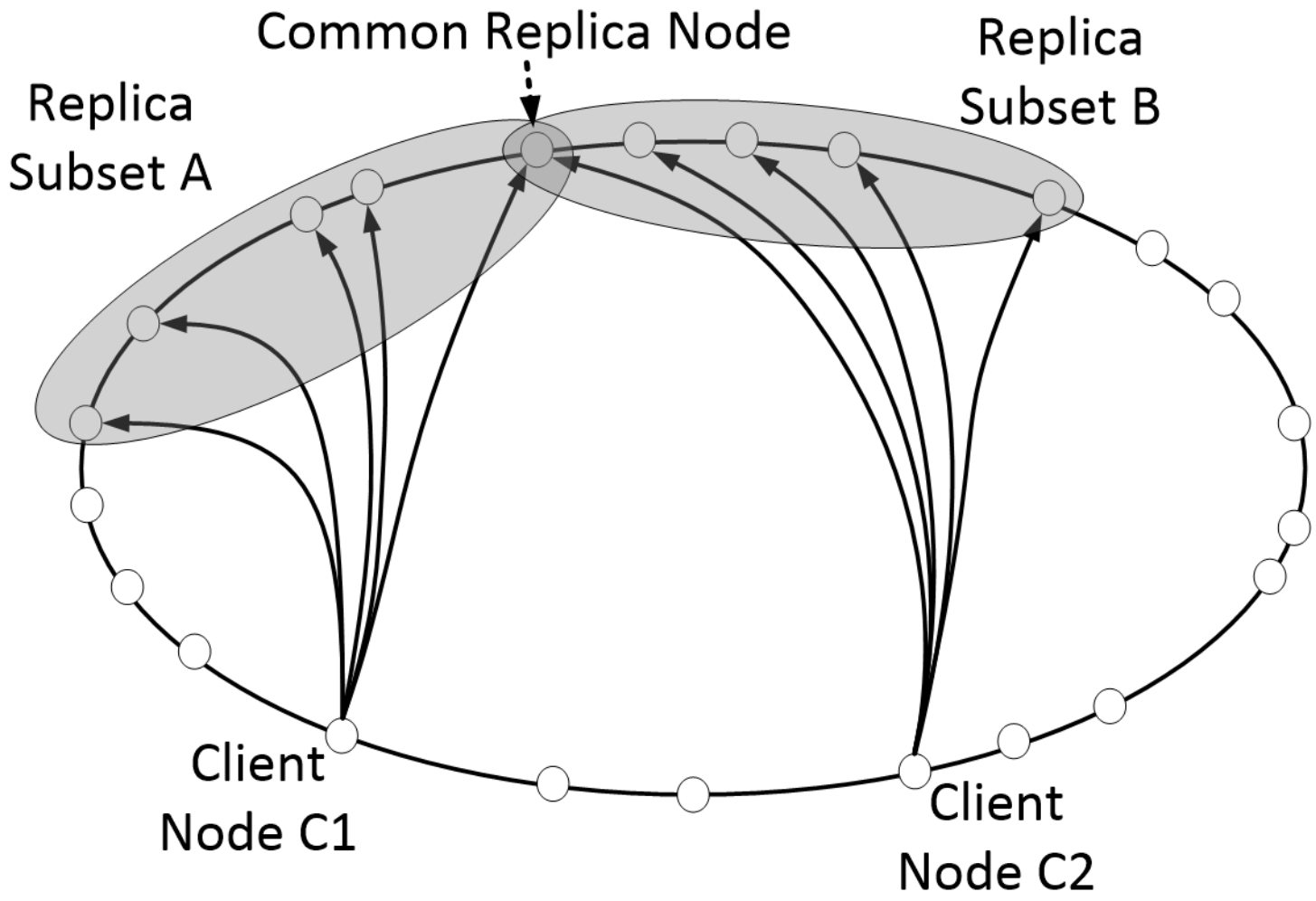
- 3 strategies implemented in TomP2P
  - Versioned put/get - increase version
  - Optimistic put - assume no conflict, if conflict, remove version
  - Pessimistic put - set prepare flag, if no conflict, commit

vDHT Pessimistic put

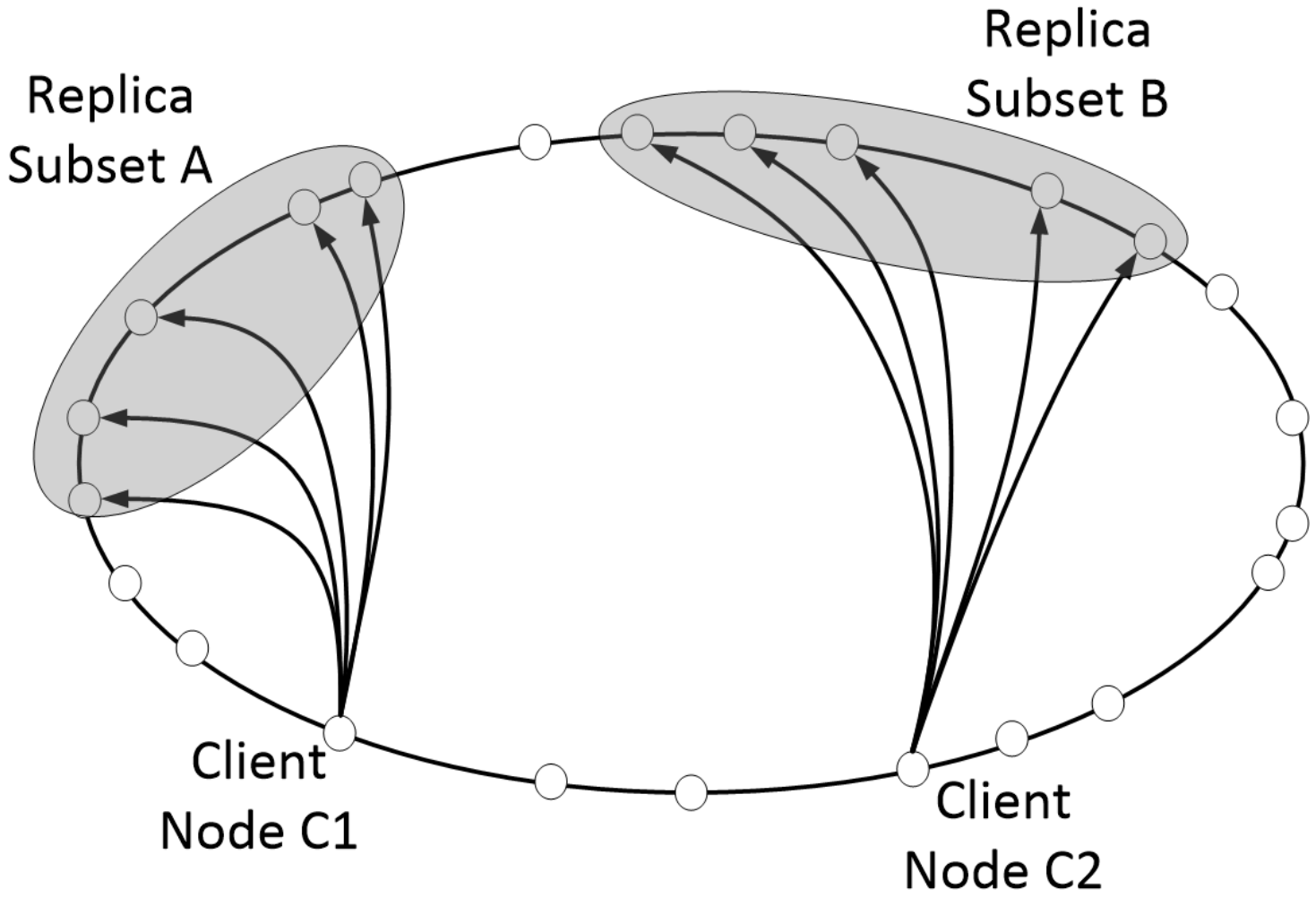
- Highest consistency semantics
  - 1. put with prepare flag, set short TTL
  - 2. put returns R results, indicating if another version exists
  - 3a. if no other version, put with commit flag, regular TTL
  - 3b. If other version, remove prepared data, get latest version, continue with step 1.) → exponential backoff

- Strong consistency in “light” churn.
- Light/heavy churn?
  - Light churn if at least one common replica peer has latest version

Light Churn Conditions

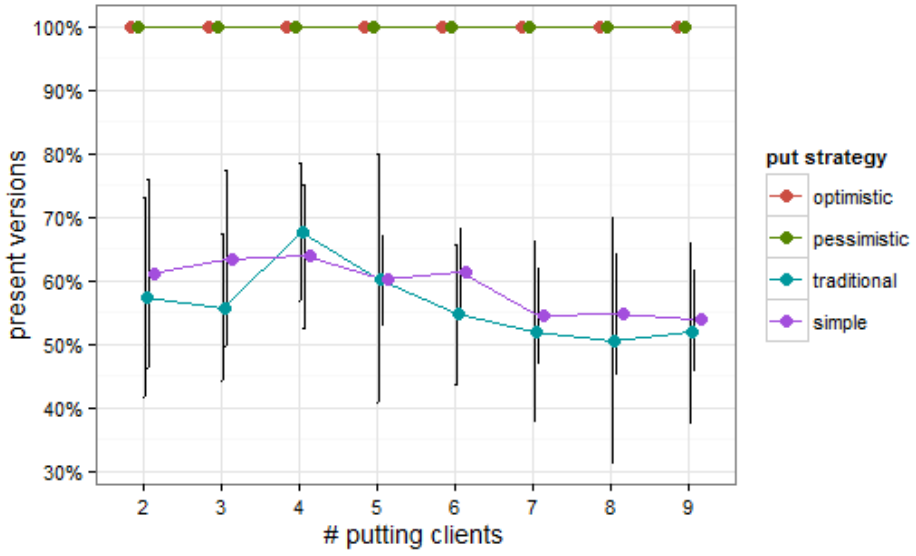


Heavy Churn Conditions

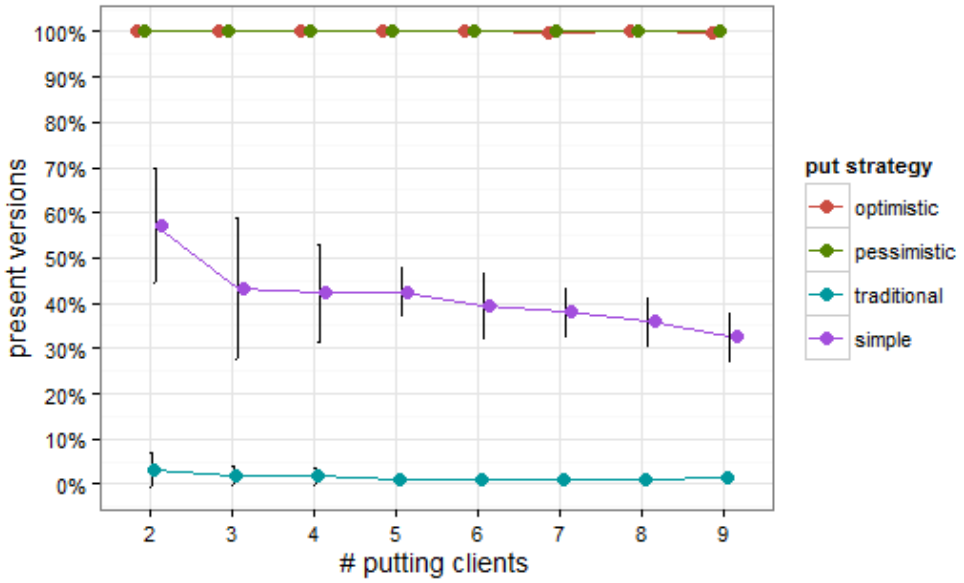


Results (1)

- No churn

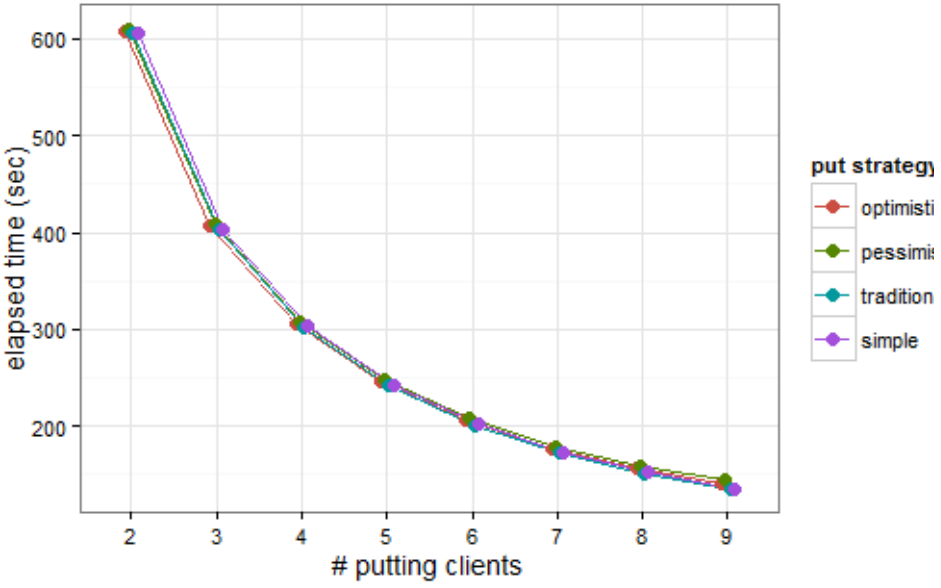


- Light churn

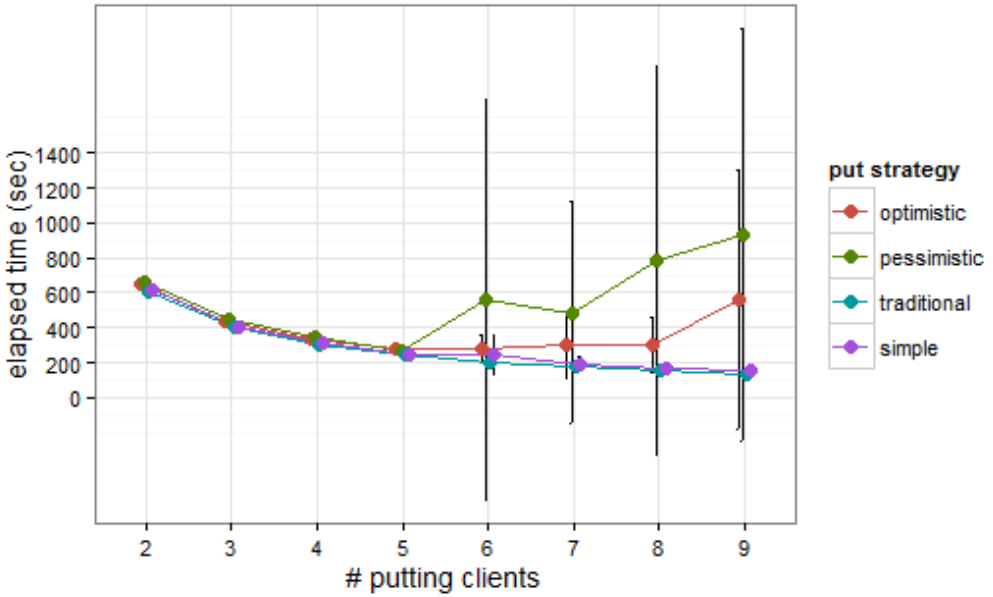


Results (2)

- Elapsed time no churn



- Elapsed time (heavy) light churn

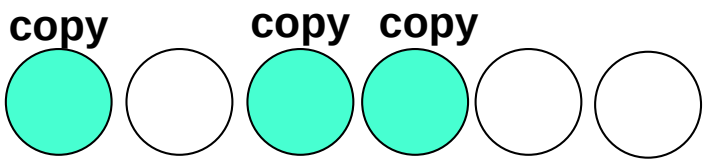


Results / Conclusions

- Consistency is not free!
  - Pessimistic takes the longest
- Works with a version history.
  - Traceability of updates; recovery of older versions; no overwrites.
- Directly integrated into DHT, no additional layer.
- No usage of timestamps, no time synchronization.
- No usage of locks, non-blocking behavior.
- Pessimistic put: light churn: consistent updates
  - Heavy churn: (manual) merging
- Number of replicas influences light/heavy churn

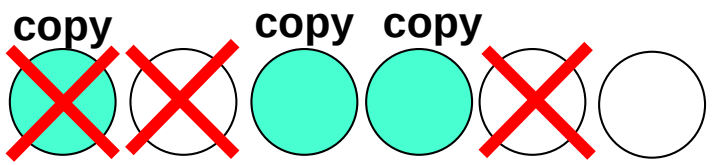
Introduction – Churn Prediction (1)

- How many replicas?
- Enough replicas are important:
  - Not to loose data in the presence of churn
  - Increases fault-tolerance and provides high data availability



Introduction – Churn Prediction (2)

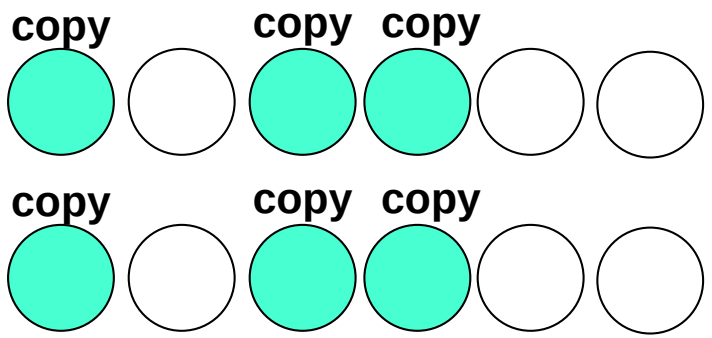
- How many replicas?
- Enough replicas are important:
  - Not to loose data in the presence of churn
  - Increases fault-tolerance and provides high data availability



- Make sure update happened under light churn
- How to determine a good replication factor?

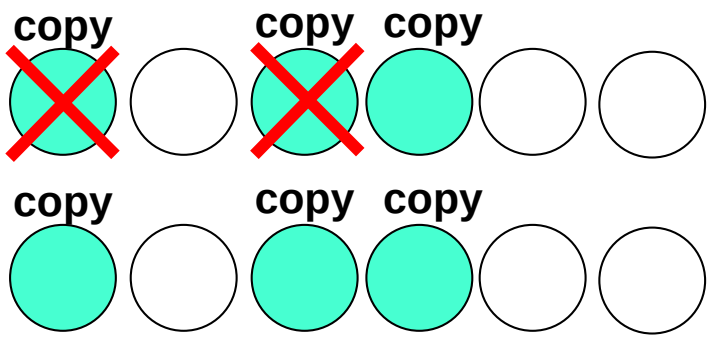
Problem Statement and Goal (1)

- Static replication factor in case the churn rate is not stable
  - if churn rate is 2, it is ok



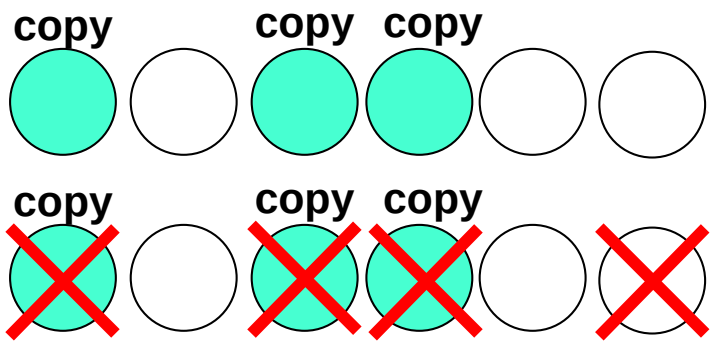
Problem Statement and Goal (2)

- Static replication factor in case the churn rate is not stable
  - if churn rate is 2, it is ok



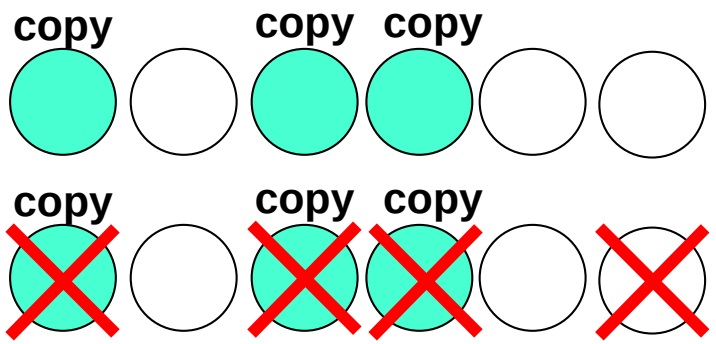
Problem Statement and Goal (3)

- Static replication factor in case the churn rate is not stable
  - if churn rate is 2, it is ok
  - if churn rate is 2-5, it is a problem



Problem Statement and Goal (4)

- Static replication factor in case the churn rate is not stable
  - if churn rate is 2, it is ok
  - if churn rate is 2-5, it is a problem



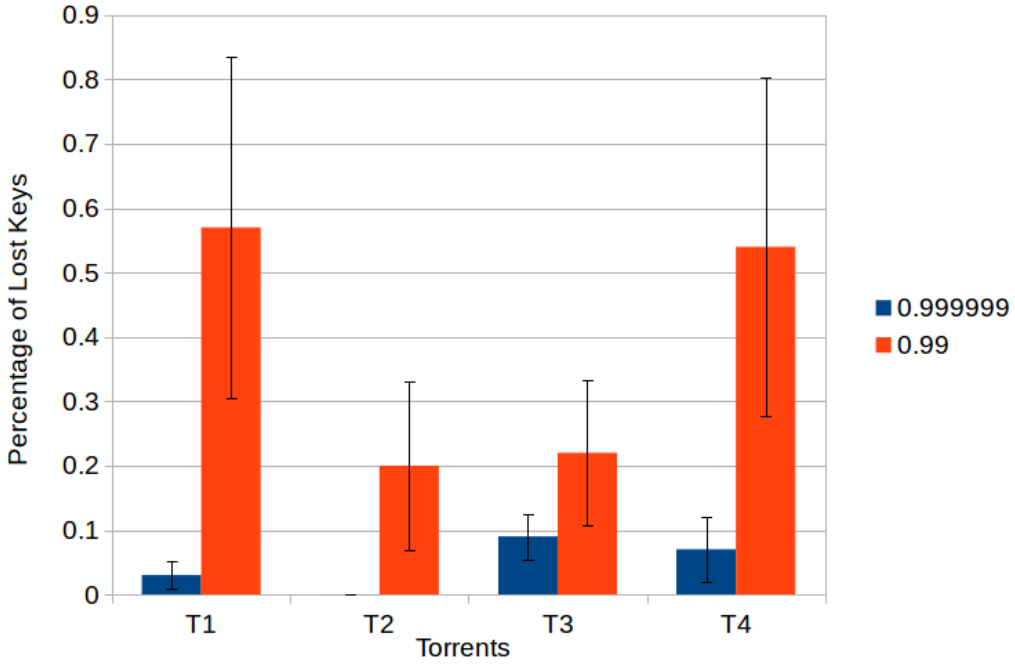
- Goal: changing the replication factor dynamically according to churn rate

Churn Prediction

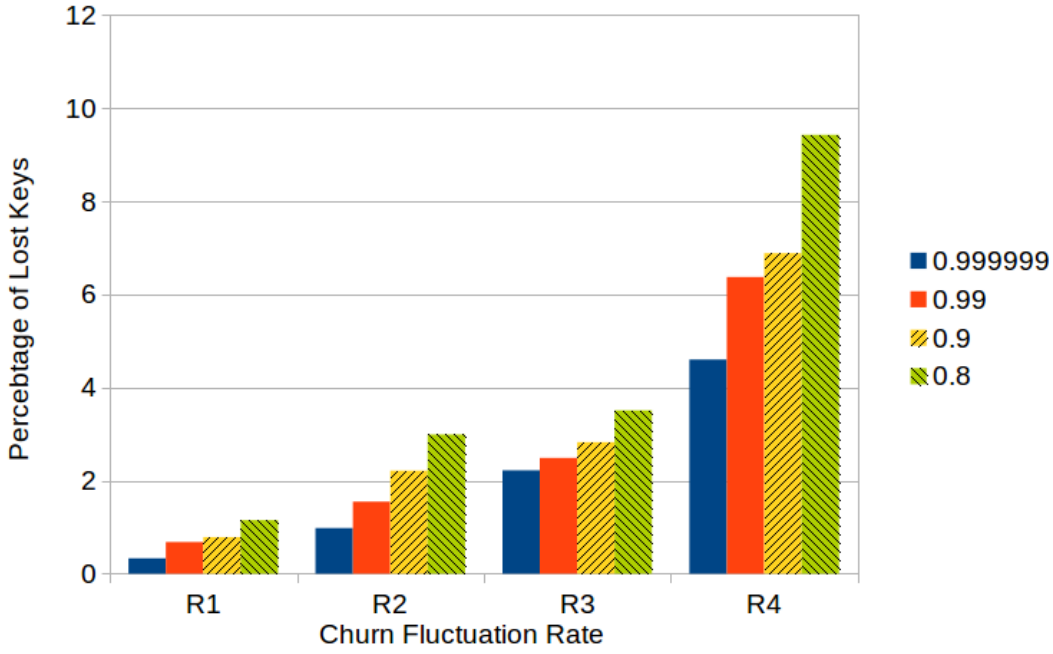
- P2P file sharing based on diurnal pattern
  - Well, what about DHTs for local clusters or IoT?
- Moving average:
  - E.g., used to define trend and make forecasts by smoothing out fluctuations (predicting stock price, forecasting GDP)
- Replication factor based on reliability (e.g., 99%) and predicted churn (EMA), for random churn
  - $\boldsymbol{r} = 0.99 = 99\%, \boldsymbol{n} = 320, \boldsymbol{k} = 40 \rightarrow$ replication factor=3
- For worst case nodes leaves (to achieve light churn), use the predicted value + x for the number of replicas
  - If prediction is correct x = 1, otherwise increase x → rf=41

Results

- Reliability Internet, data from BT



- Reliability with Synthetic data



Conclusion / Discussion

- Consistent updates in a DHT is possible with versioning
  - But only in light churn, heavy churn requires manual intervention
- To stay in light churn, correct guess of number of replicas necessary
  - Reliability is only as good as your prediction
  - EMA is used as a general purpose predictor
  - Unpredicted bursts? → increase number of replicas further

Questions?