A Peer-to-Peer Notification System for Distributed Online Social Networks

Michele Amoretti, Lorenzo Gandolfi, Michele Tomaiuolo
Department of Engineering and Architecture, University of Parma

Contacts:
michele.amoretti@unipr.it
michele.tomaiuolo@unipr.it

Parma, 28/6/2019

UNIVERSITÀ DI PARMA

Summary

- Introduction
- Distributed Social Architectures
- Proposed Algorithms
- Simulations
- Conclusion and Future Work

UNIVERSITÀ DI PARMA

Distributed Social Architectures

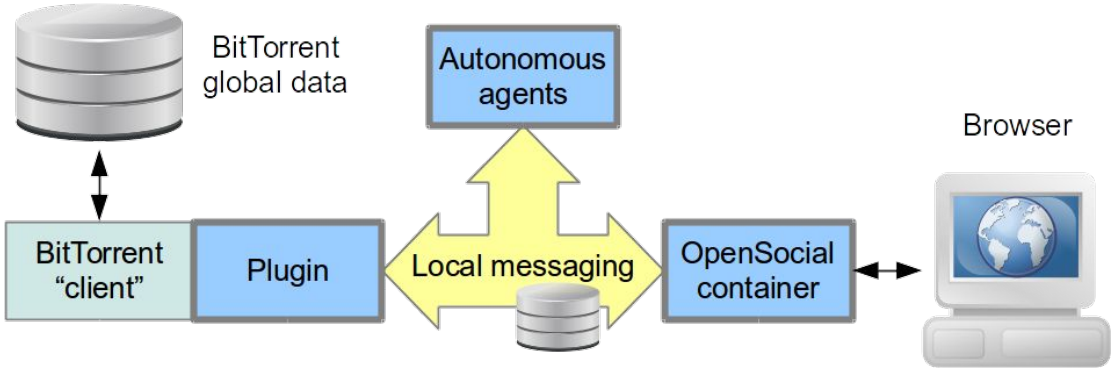
- Popular social networking platforms
  - Centralized: algorithms (recommendation etc.) have full access to data
  - Web based: easy access, across devices
- But... there are some “*buts*”
  - Privacy leaks and mass surveillance
  - Social silos, walled gardens
  - Arbitrary censorship
  - Availability and reachability during crisis
  - High costs, data exploitation, ads

Distributed platforms

- Federated systems
  - Diaspora, StatusNet
  - Users can suffer attacks to popular pods
- Custom peer-to-peer systems
  - Freenet, PeerSoN, LotusNet, Safebook, Persona, Life Social
  - In some cases, a single node can still host various users
- Lack of...
  - Systems based on popular P2P protocols
  - Open source, workable implementations

Design of Blogracy

- Modular design, orthogonal solutions for different aspects
  - Web application for social data
  - P2P file sharing
  - Async messaging
- Available on Github
- http://blogracy.net/



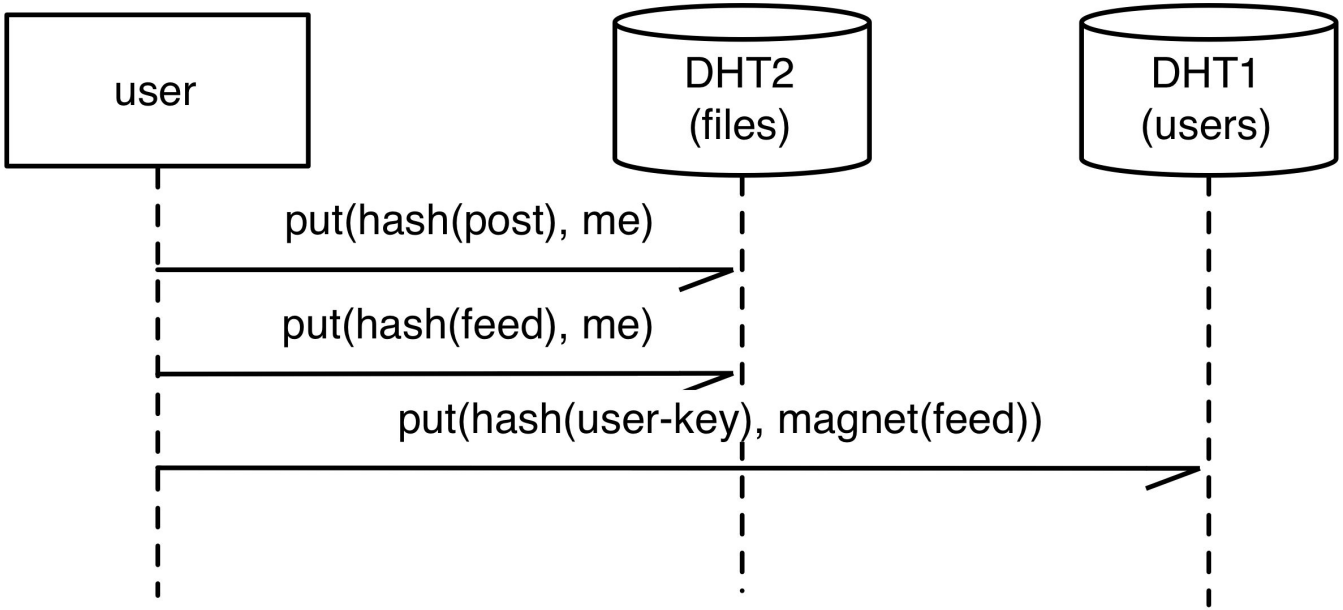
Basic formats and protocols

- OpenSocial
  - W3C standard for social activities, based on JSON
  - Recent social activities saved in a local file
  - File includes *profile information*, and a list of *followees*
- BitTorrent, file sharing
  - P2P, fully distributed
- Kademlia, Distributed Hash Table
  - All nodes maintain information about file availability
  - Without servers/trackers

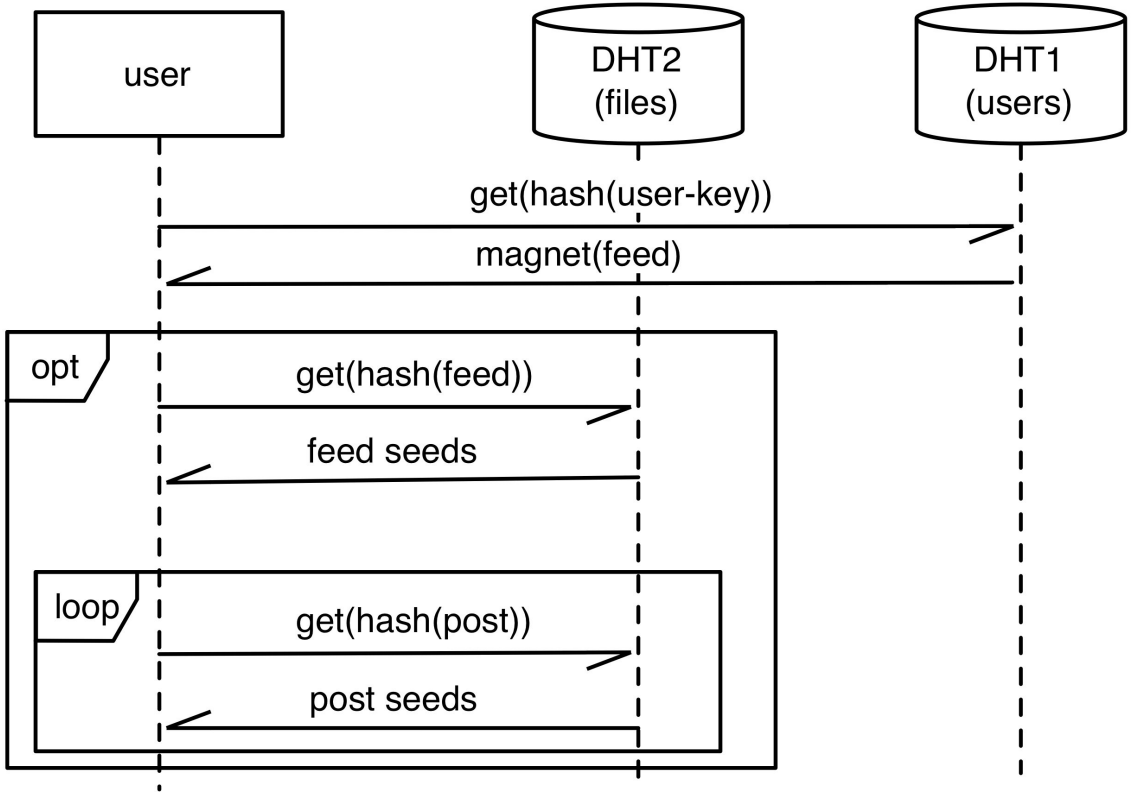
Identity and security

- Key based identity
  - Each user is identified by his/her public key
- Each user has an associated file
  - Social db: profile, followees and activities
  - DHT for mapping: *user_id → file_hash*
- Signed activities for integrity
  - JSON Web signature
- End to end encryption for confidentiality
  - Attribute-Based Encryption

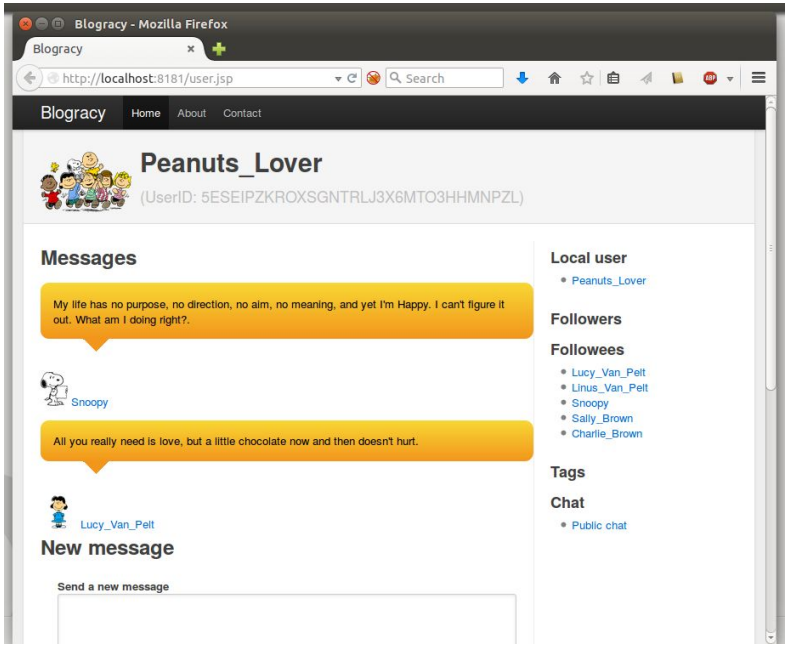
Resource sharing



Updating the followees' feeds



Implementation over BitTorrent



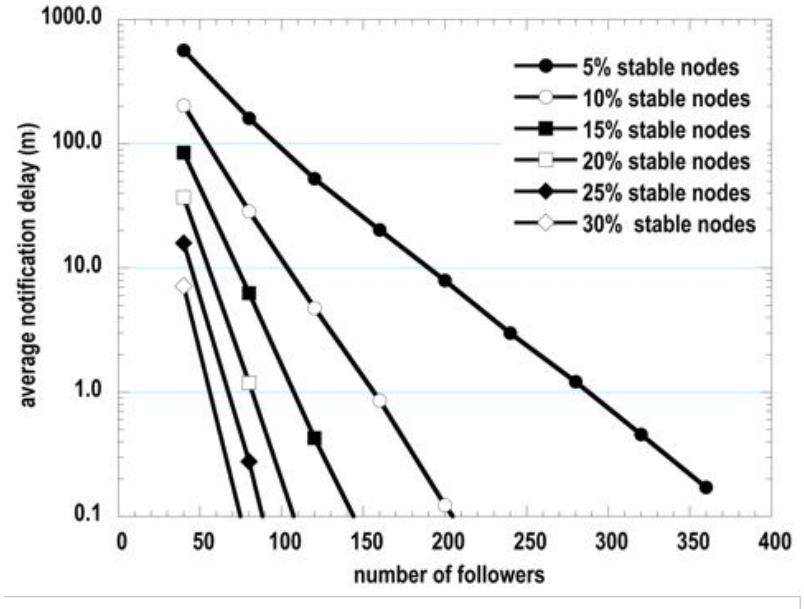
- Plugin for Vuze (fka Azureus)
  a. Java-based BitTorrent client
  b. Object-oriented, extensible
- Web application
  a. Embedded Jetty web server
- Communications through JMS
  a. ActiveMQ, WebSockets
  b. Instant messages & updates exchanged with the browser
- Publish/subscribe channels based on Vuze DHT

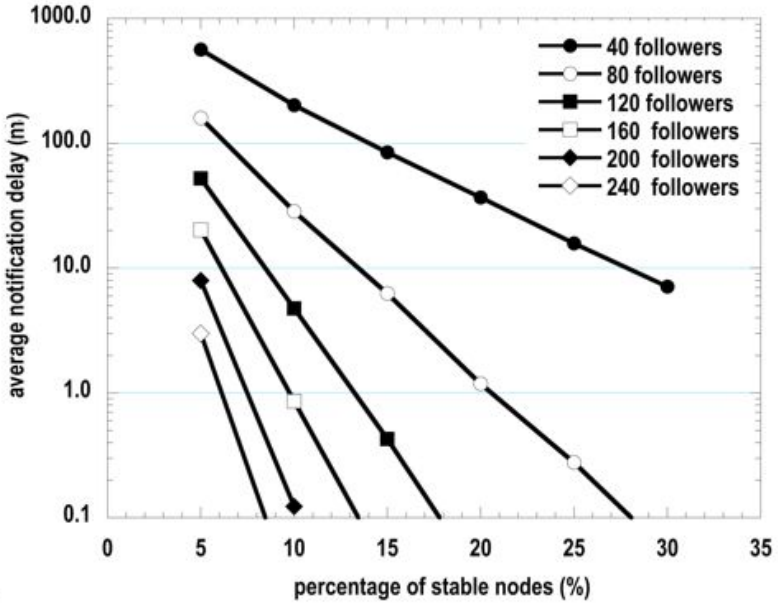
Node churn and polling

- In P2P, some users connect and others disconnect, continuously
- But in BitTorrent, downloading a resource corresponds to sharing it
  - If a user has enough followers, they contribute to make his/her messages available
  - Also when the original message source is offline
- Ok with 120 followers, 10% stable nodes → 5 min delay
  - Stable nodes connected for 8 hours a day
  - Others connected occasionally (3 times a day, 15 min. total)
- Otherwise, shared trusted always-on nodes for new users

Simulations of node churn

- Delays grow exponentially when stable nodes or followers decrease





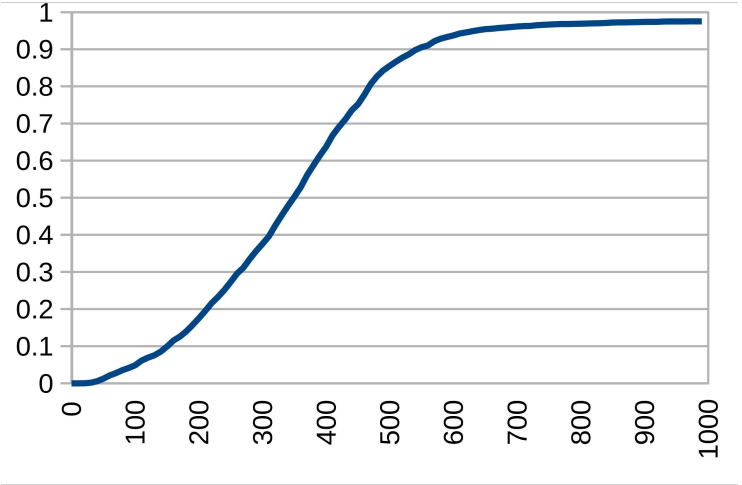
Node churn and polling

- Polling: unnecessary delays and traffic
  - Pushing: not always appropriate; node churn in P2P
  - Need to strike a balance between polling/pushing mechanisms
- Blogracy: pushing mechanism based on DHT
  - Publish/subscribe on multiple channels
  - Potentially, a channel for each user
  - Users interested in a channel, share a torrent
  - The shared hash is obtained by the name of the channel
  - Messages are sent directly to all online subscribers

UNIVERSITÀ DI PARMA

Pushing update notifications

- Testing: 20 nodes over the *PlanetLab* testbed, constantly online
- All nodes participate to a common publish/subscribe channel



- *Figure*: cumulative distribution of the delay (millis) of direct notification
  - 90% of messages are received in (less than) half a minute

Spanning tree

- Push notifications are effective, for online nodes
- But a node cannot send *direct* notifications to all online followers
- The mechanism needs to scale to thousands of nodes and more
- If nodes are organized in a spanning tree (like IRC servers), work can be *distributed*

Proposed Algorithms

- Group Join (i.e., how to select the entry point)
  - *root* strategy:
    - find the node that is the root of the tree
    - send a “join” message to that node
    - may lead to a more balanced tree
  - *first* strategy:
    - find any node belonging to the tree
    - send a “join” message to that node
    - reduced workload on the root node
    - may lead to a less balanced tree

Proposed Algorithms

- Connection (i.e., how to become part of the spanning tree)

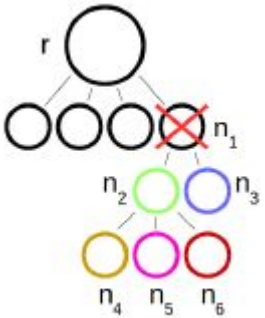
Algorithm 2 Management of a join request. *numChildren* represents the children count, *maxChildren* is the maximum degree set for the tree.

Require: $request\ from\ node\ n$
Ensure: $request\ response$
  if $numChildren < maxChildren$ **then**
    $buildConnection()$
    $numChildren = numChildren + 1$
    $return\ ConnectionAccepted$
  else
    $response = ConnectionRefused$
    $response.setAlternativeNode()$
    $return\ response$
  end if

UNIVERSITÀ DI PARMA

Proposed Algorithms

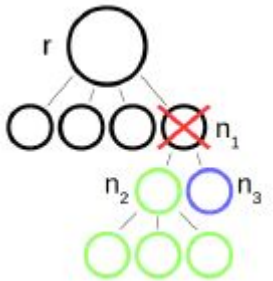
- Tree Reconstruction
  - *subtree breakout* algorithm



- n1 fails
- n2 .. n6 try to reconnect to the tree autonomously

Proposed Algorithms

- Tree Reconstruction
  - *subtree preservation* algorithm



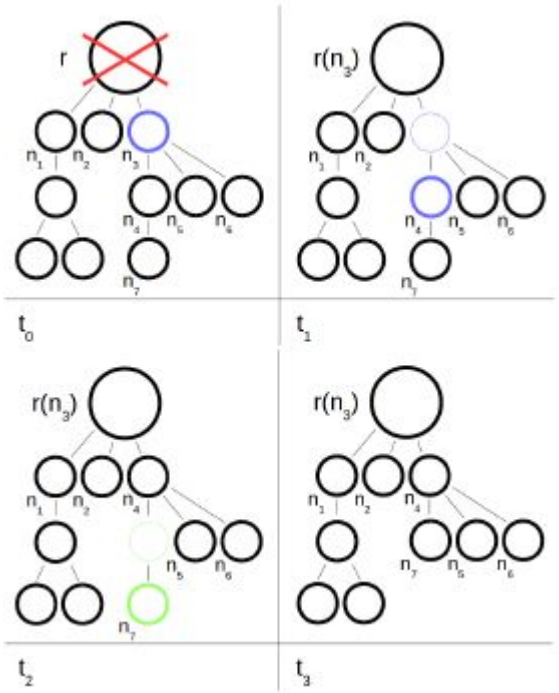
- n1 fails
- n2 preserves its subtree while looking for a new parent node
- n3 (isolated) tries to reconnect to the tree autonomously

UNIVERSITÀ DI PARMA

Proposed Algorithms

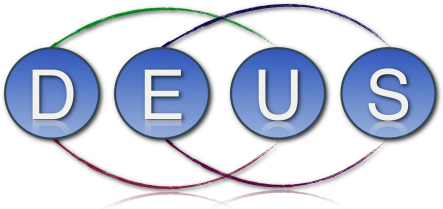
- Tree Reconstruction
  - *subtree breakout* algorithm

- recursive election according to one of
  - XOR distance
  - lifetime



UNIVERSITÀ DI PARMA

Simulations



- **DEUS: a general purpose tool for complex system simulation**
- Simple Java API for implementing **nodes, events, processes**
- Project homepage: **https://github.com/dsg-unipr/deus/**
- Main papers:

M. Amoretti, M. Agosti, F. Zanichelli, DEUS: a Discrete Event Universal Simulator, 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy

M. Amoretti, M. Picone, F. Zanichelli, G. Ferrari, Simulating Mobile and Distributed Systems with DEUS and ns-3, International Conference on High Performance Computing and Simulation 2013, Helsinki, Finland

Simulations

The tree construction algorithms are compared in terms of

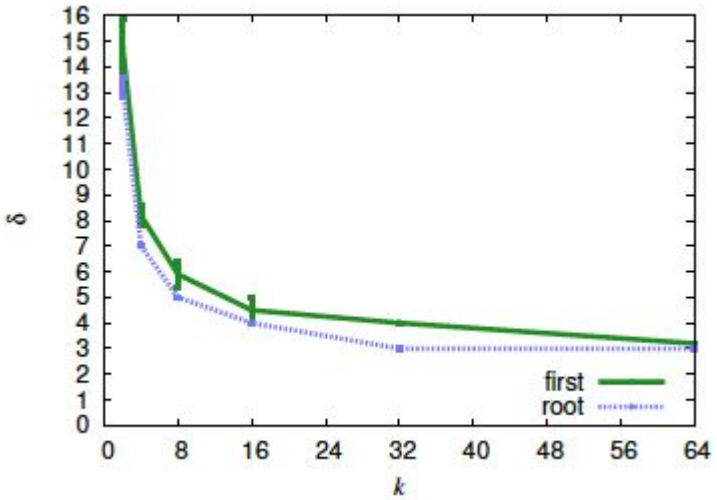
- workload distribution on network nodes
- quickness
- communication robustness

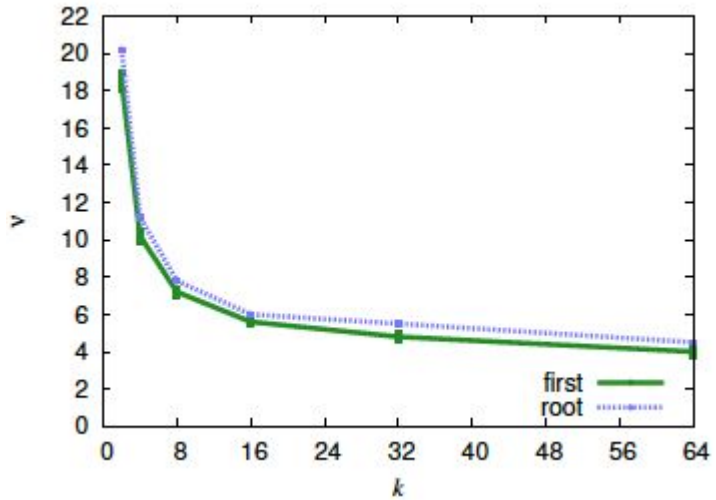
by means of the following performance indicators:

- number of control messages v
- tree depth δ
- propagation delay π

Simulations

- Group Join: δ (tree depth) and v (control messages) for different values of k (node degree)



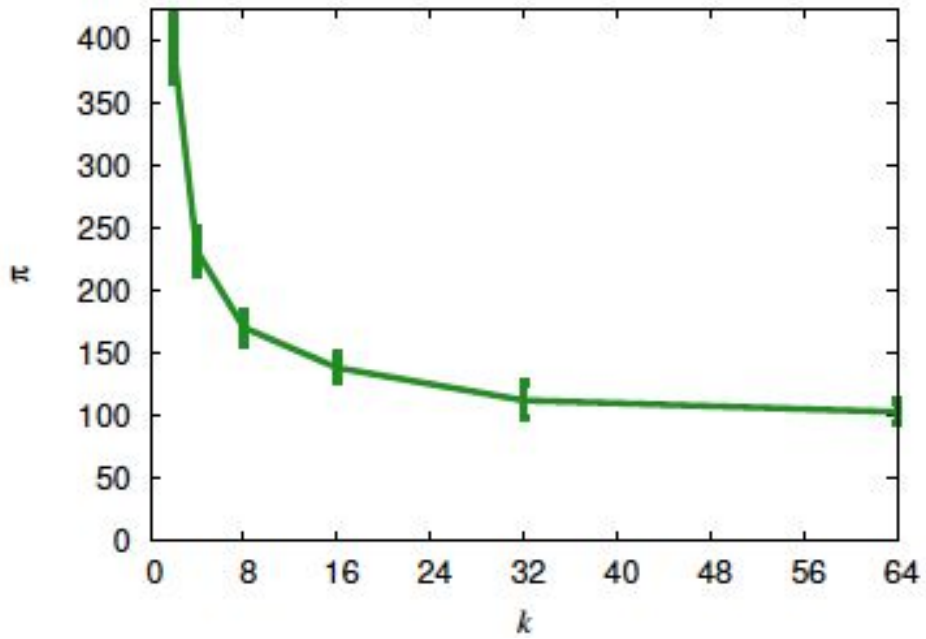


first is better than root (from now on, we assume first)

UNIVERSITÀ DI PARMA

Simulations

- Propagation delay: reduced gains for *k*>8 (*node degree*)



Simulations

- Tree reconstruction, for the *subtree breakout* strategy
- First row: δ (*tree depth*) and ν (*control messages*) before nodes fail

% of failed nodes	δ	variation	ν	variation
-	6.0	-	7.1	-
1%	6.0	0%	7.5	4.3%
5%	5.9	-1.5%	9.4	31.1%
10%	5.8	-2.8%	11.5	60.9%
20%	5.8	-2.8%	15.6	117.3%
50%	5.4	-9.0%	37.9	427.9%

Simulations

- Tree reconstruction, for the *subtree preservation* strategy
- First row: δ (*tree depth*) and ν (*control messages*) before nodes fail

% of failed nodes	δ	variation	ν	variation
-	6.0	-	7.1	-
1%	6.4	8.1%	7.3	1.8%
5%	7.5	26.2%	7.6	6.1%
10%	7.0	18.0%	8.1	13.7%
20%	8.1	36.0%	9.9	38.6%
50%	8.6	44.2%	18.9	163.7%

Simulations

- Tree reconstruction, for the *recursive election* strategy
- First row: δ (*tree depth*) and ν (*control messages*) before nodes fail

% of failed nodes	δ	variation	ν	variation
-	6.0	-	7.1	-
1%	6.0	0%	7.2	1.3%
5%	6.0	0%	7.6	5.8%
10%	6.0	0%	8.0	12.3%
20%	6.0	0%	9.3	29.5%
50%	5.9	-1.6%	14.7	105.1%

Conclusion and Future Work

- We designed and compared different strategies for creating a spanning tree over a generic P2P network of the structured type
- The adopted decentralized approach is motivated by the need for privacy
- Tree robustness is guaranteed by
  - first-based group join
  - recursive election for tree reconstruction
- We plan to implement the proposed strategies in the Blogracy platform, and to introduce adaptive strategies for parameter tuning