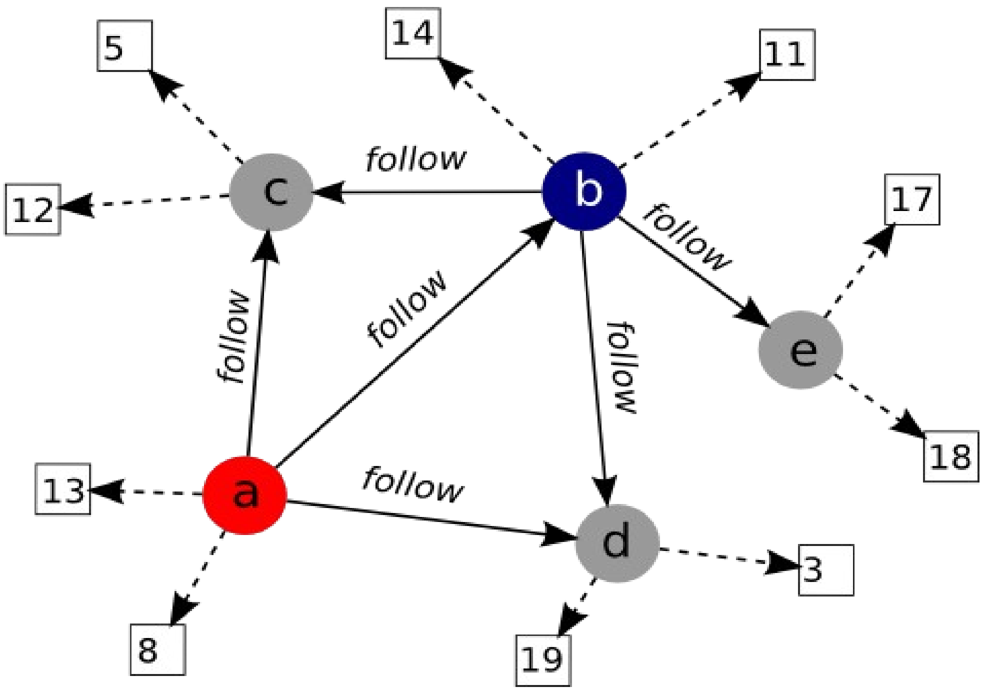
Efficient Graph Models for Retrieving the Top-k News Feeds from Ego Networks

René Pickhardt, Thomas Gottron, Jonas Kunze, Ansgar Scherp Steffen Staab



How to retrieve more than 10'000 temporal ordered news feeds per second in social networks with millions of users like Facebook and Twitter by using graph data bases (like neo4j) and Graphity

Joint collaboration with

Steffen
Staab

Ansgar
Scherp

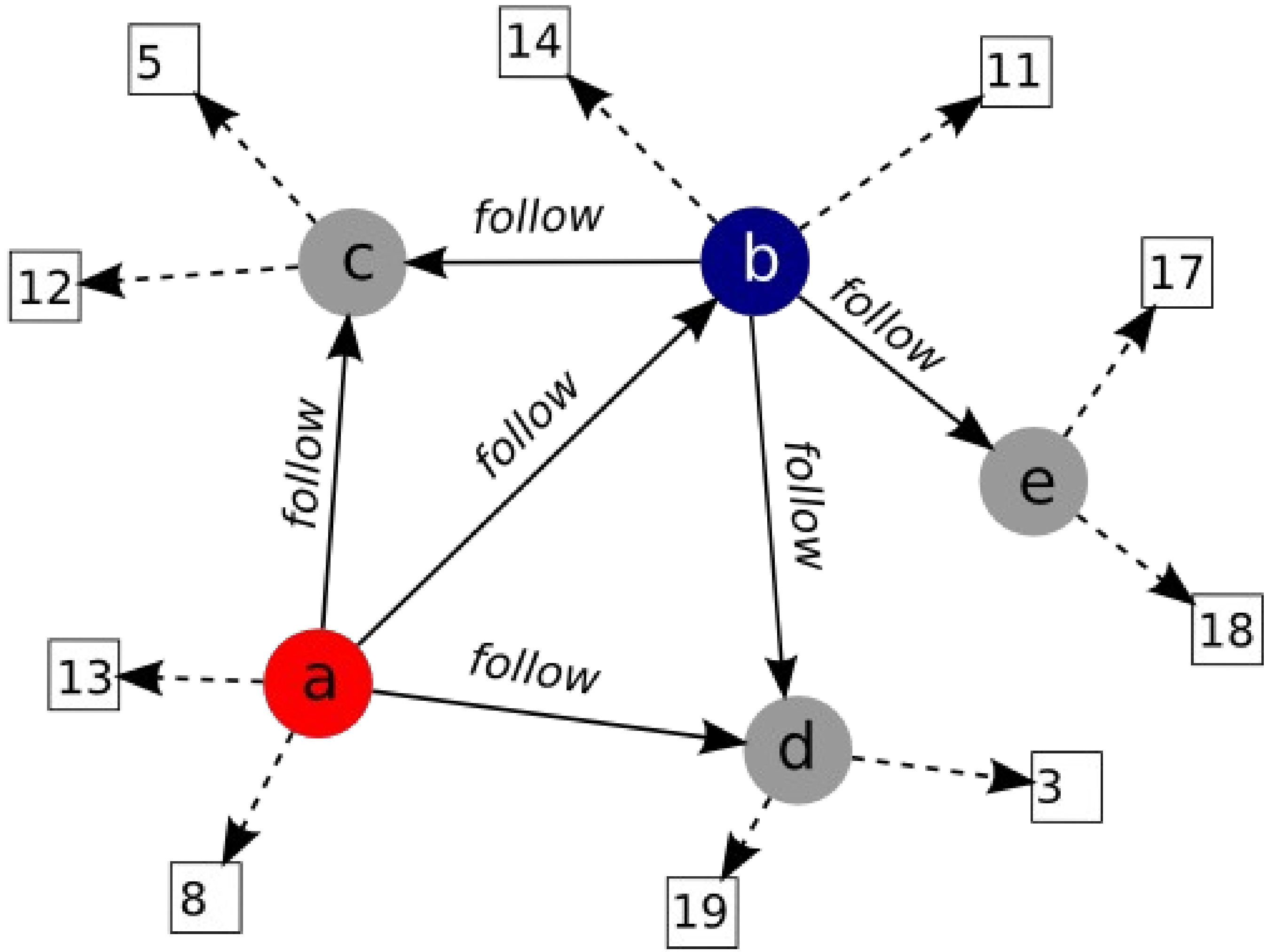
Jonas
Kunze (from metalcon.de)

Thomas
Gottron

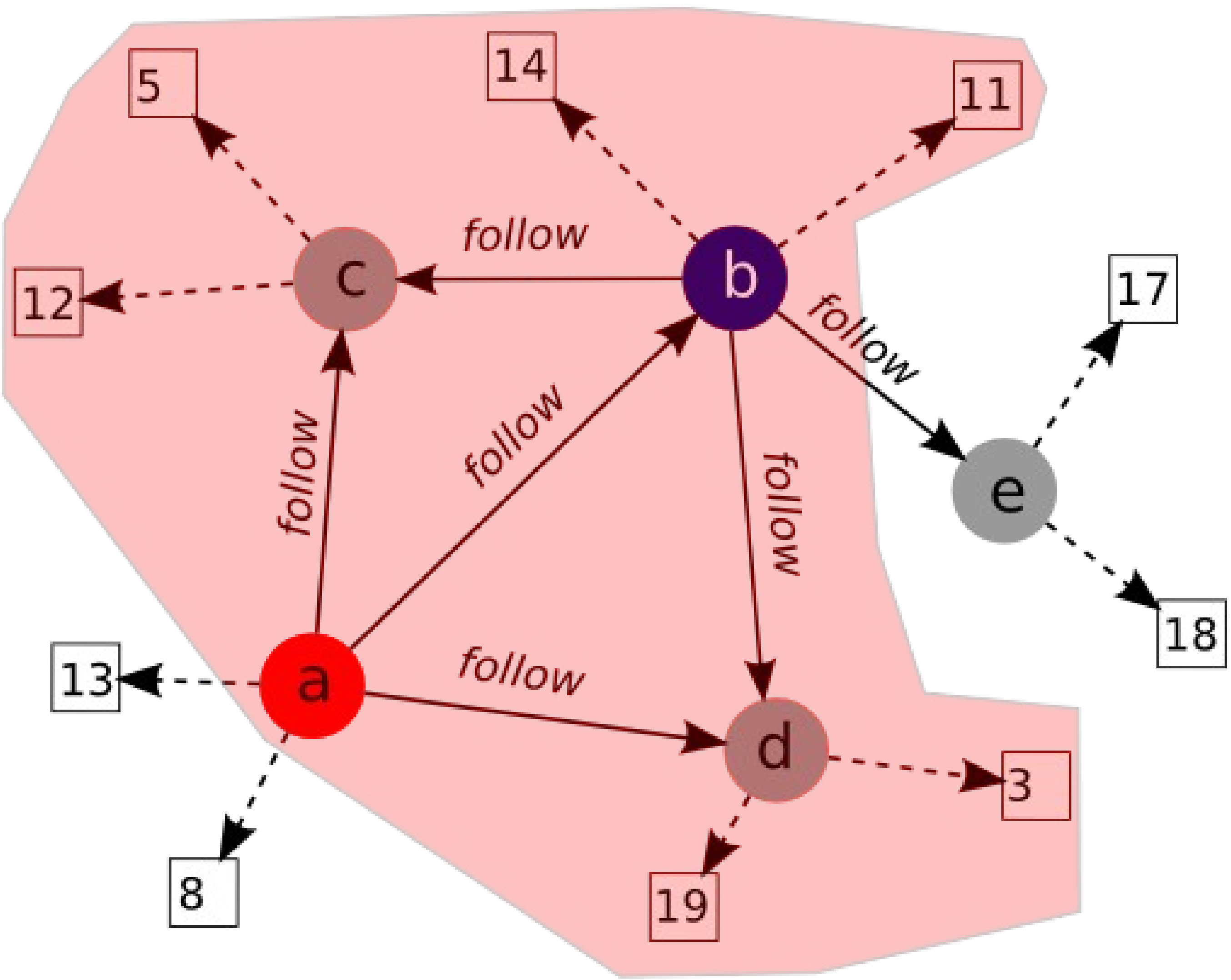
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

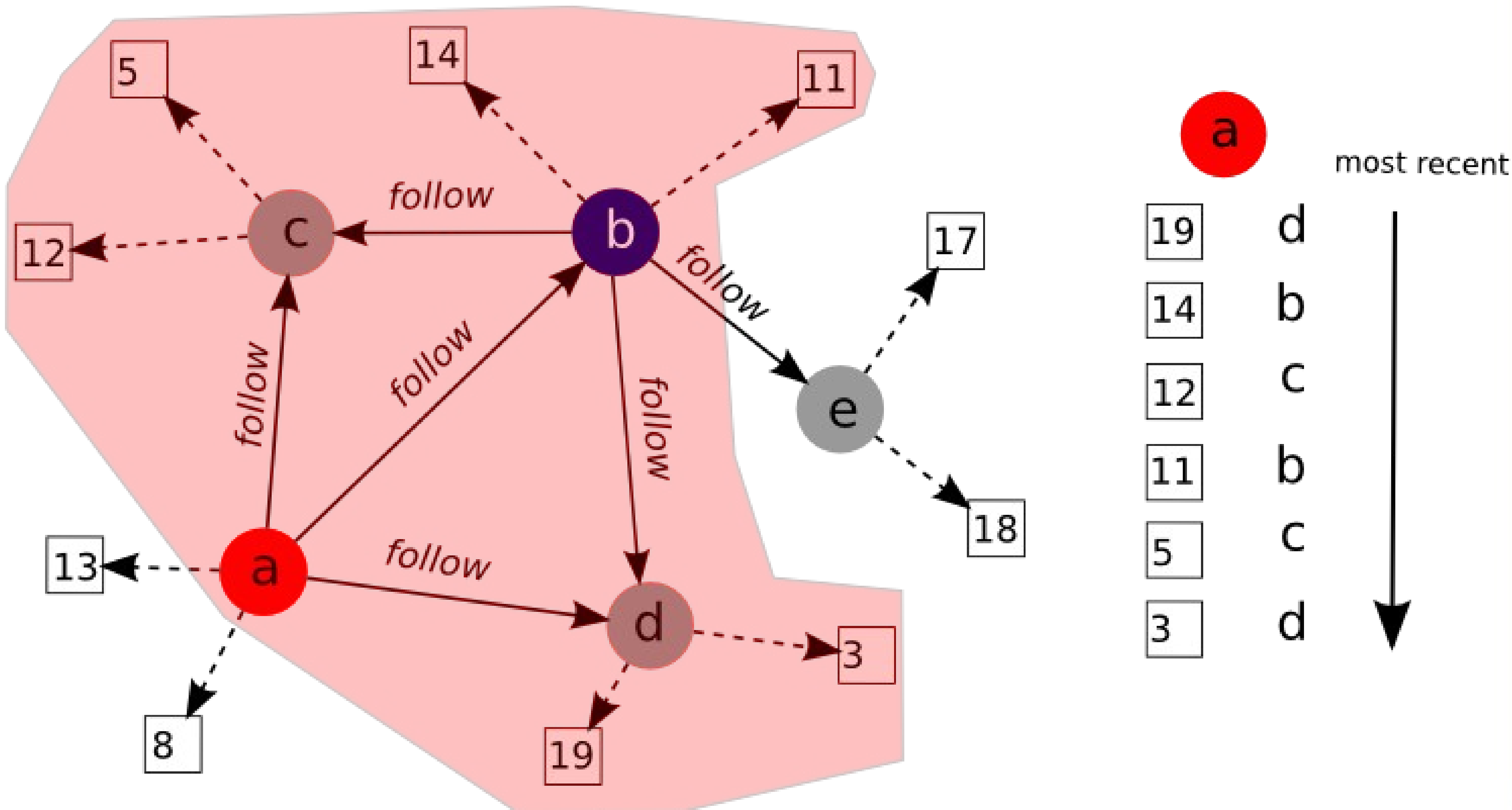
A "typical" social network graph



Retrieving Node A's news stream



Retrieving Node A's news stream



Some Challenges

Social networks like Twitter and Facebook have several thousand requested news feeds per second

News feeds change fast: Several hundred newly created content items per second. (600 tweets / sec in 2010)

News feeds are different for every user

Realtime (retrieval should be as low as micro seconds)

Friendship graph changes over time

Overall: This is a very **dynamic problem** with a lot of **chaotic & unpredictable behaviour**

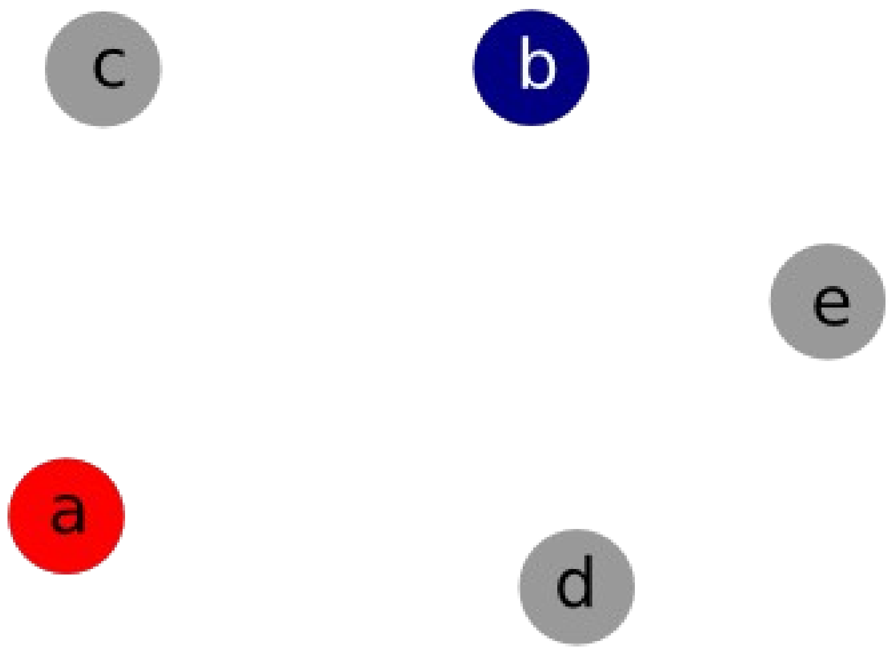
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

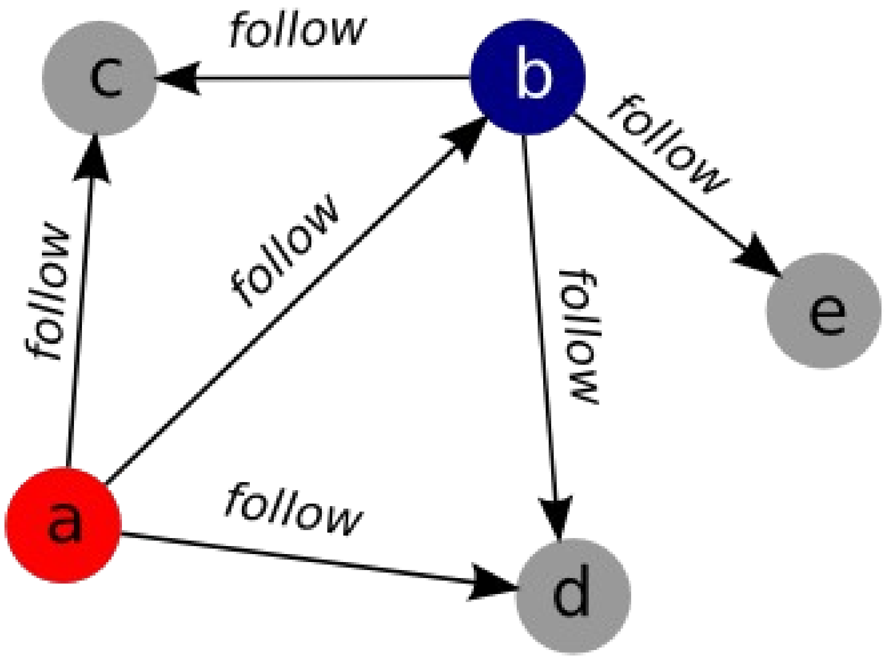
First we have some Users in a social Network

User

ID
a
b
c
d
e



They follow other users



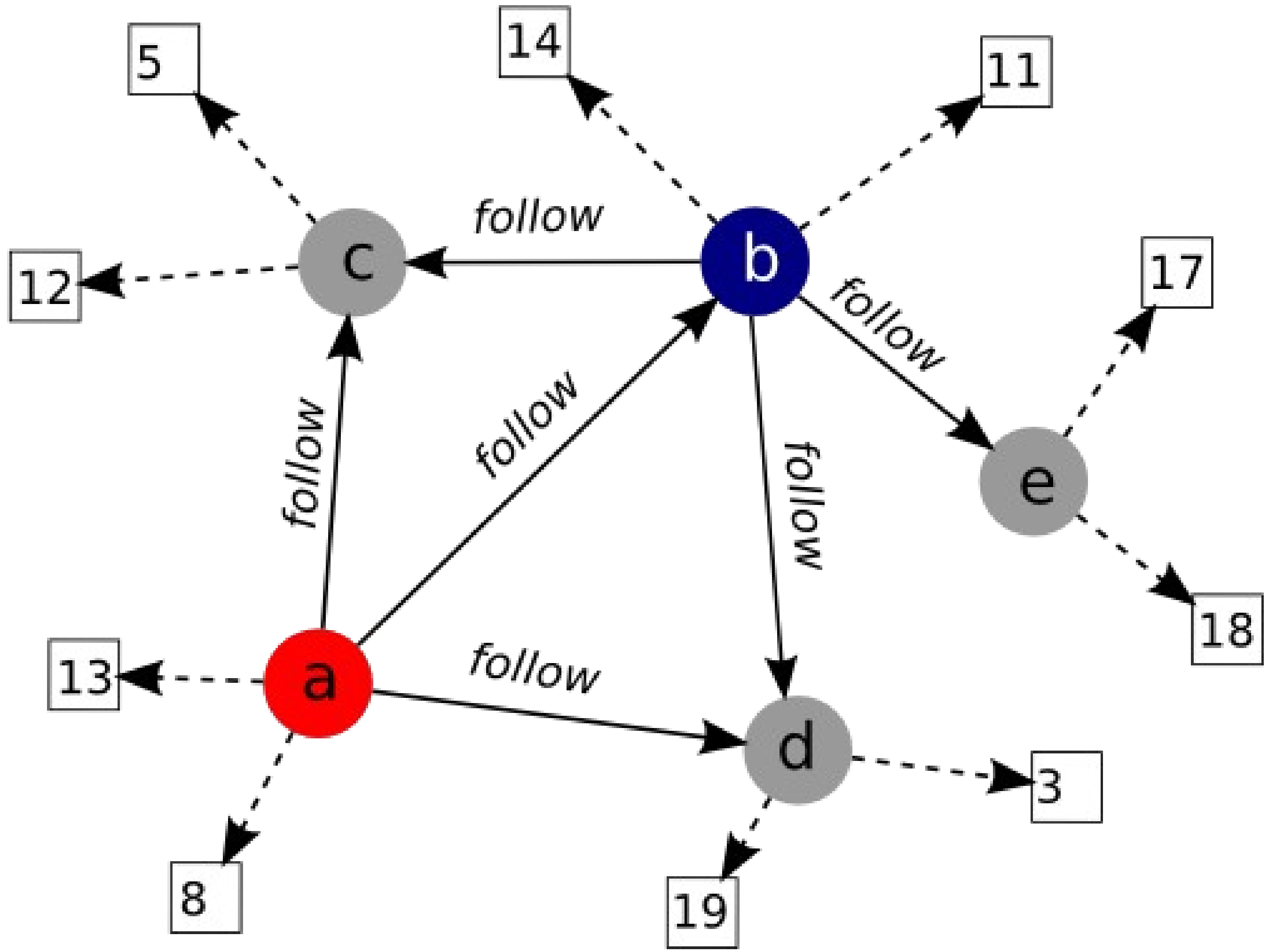
User

ID
a
b
c
d
e

Follower

from	to
a	c
a	b
a	d
b	c
b	d
b	e

Everyone produces status updates and content



User

ID
a
b
c
d
e

Follower

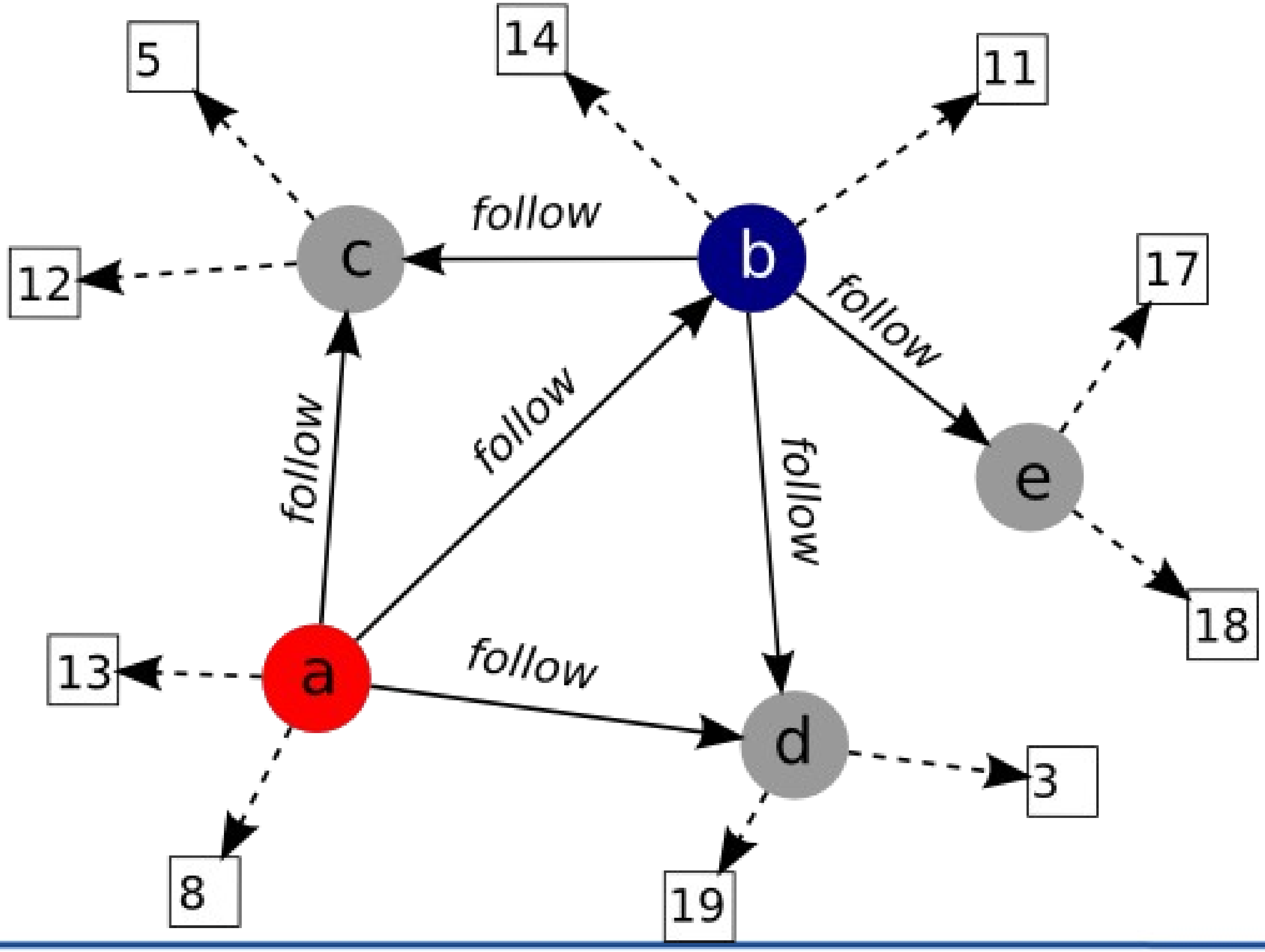
from	to
a	c
a	b
a	d
b	c
b	d
b	e

ContentItems

User	time	Content
d	19	Lorem ipsum
e	18	dolor sit amet,
e	17	consectetur
b	14	adipisici elit, sed
a	13	eiusmod tempor
c	12	incidunt ut labore
b	11	et dolore magna
a	8	aliqua. Ut enim
c	5	ad minim veniam
d	3	quis nostrud

Our Query joins over huge Follower Matrix

SELECT ci.User, ci.time, ci.Content
FROM **ContentItems** ci
JOIN **Follower** f on ci.User=f.to
JOIN **User** u on u.ID = f.from
WHERE u.ID like "a" *ORDER BY* ci.time *DESC*



User

ID
a
b
c
d
e

Follower

from	to
a	c
a	b
a	d
b	c
b	d
b	e

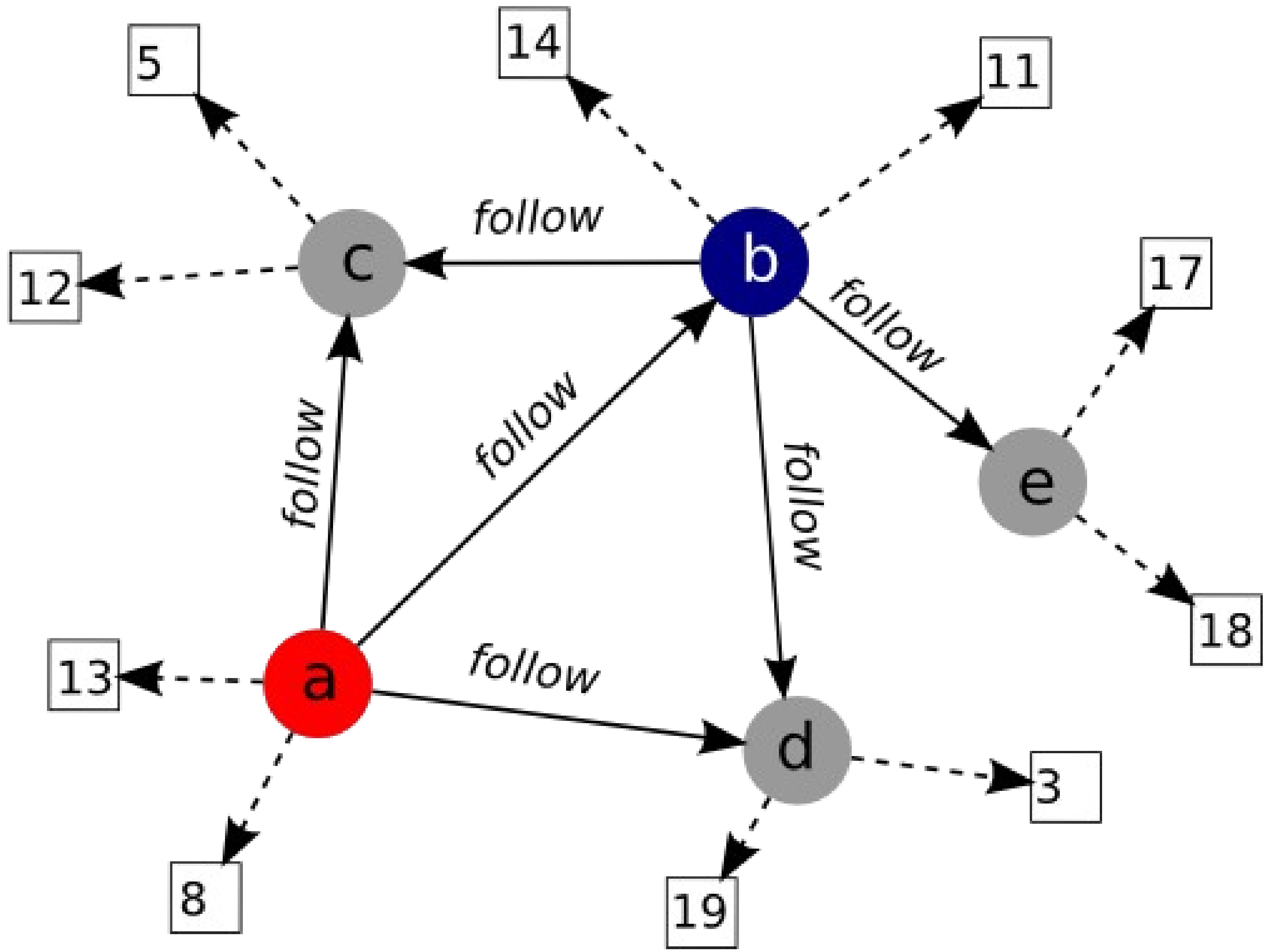
ContentItem

User	time	Content
d	19	Lorem ipsum
e	18	dolor sit amet,
e	17	consectetur
b	14	adipisici elit, sed
a	13	eiusmod tempor
c	12	incidunt ut labore
b	11	et dolore magna
a	8	aliqua. Ut enim
c	5	ad minim veniam
d	3	quis nostrud

Index

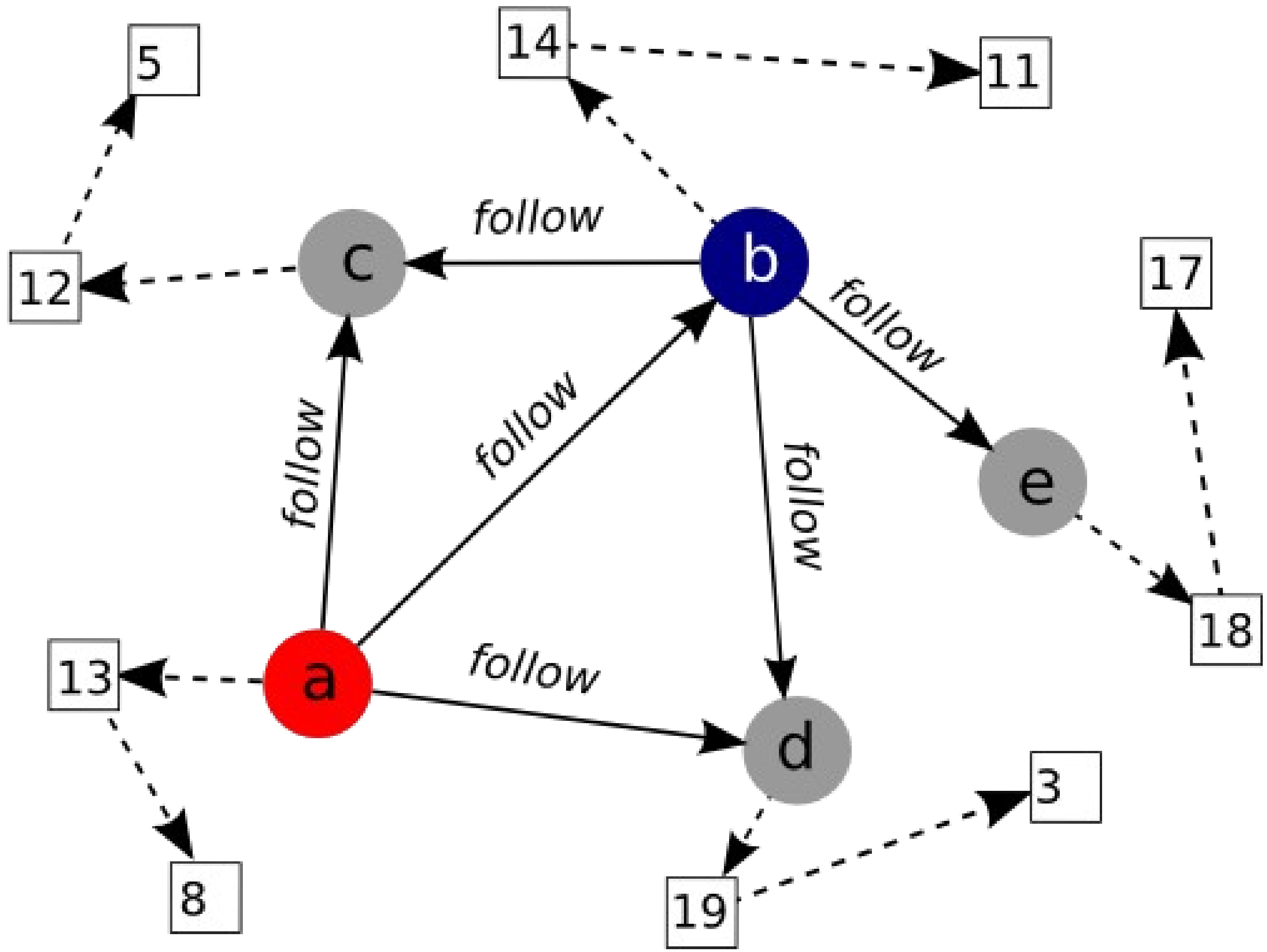
- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

STOU = Star Topology (Temporal) Ordered Updates



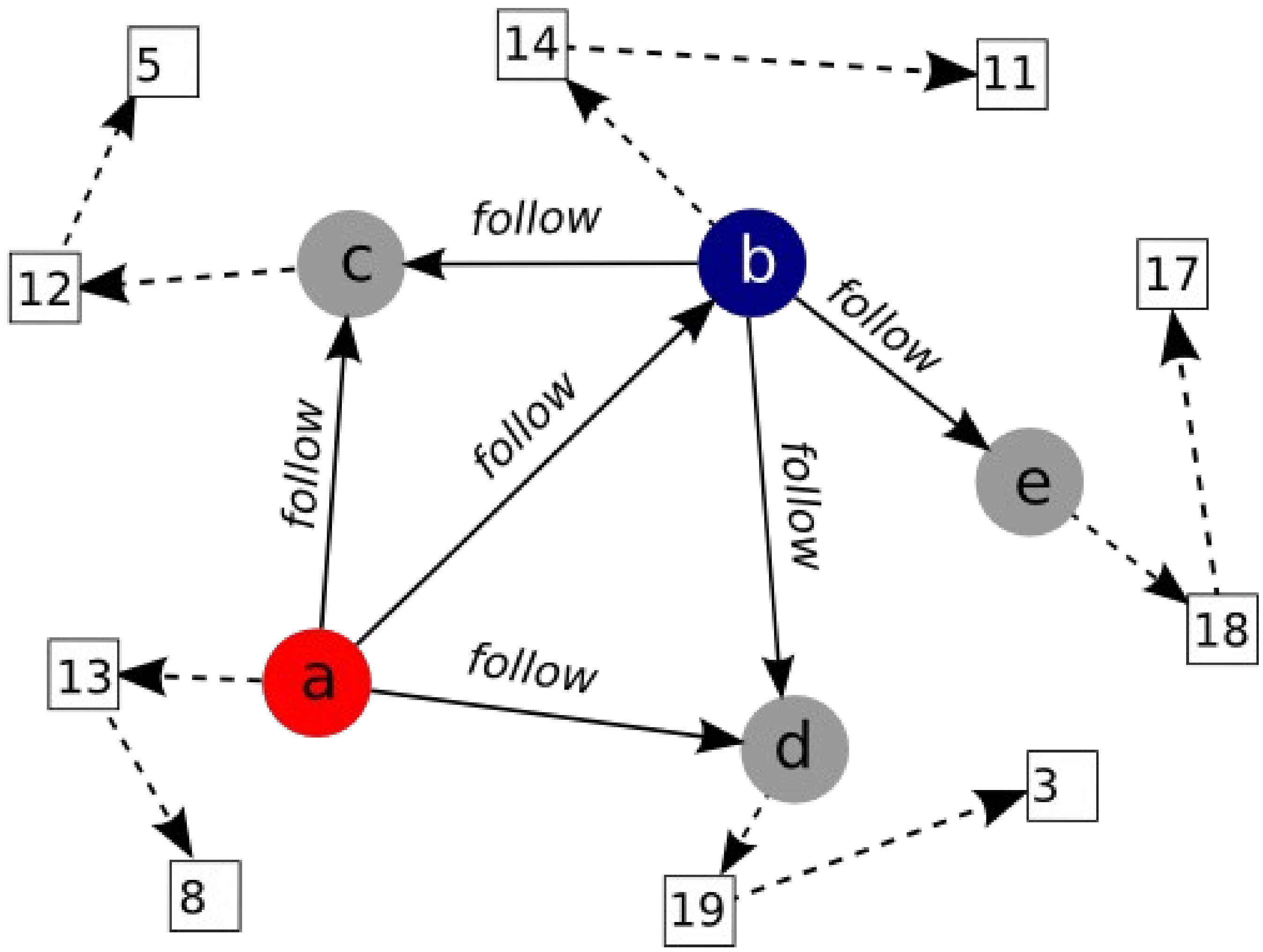
From the standard social network graph we move to (temporal) ordered lists for rectangular nodes

Introducing linked lists for status updates!



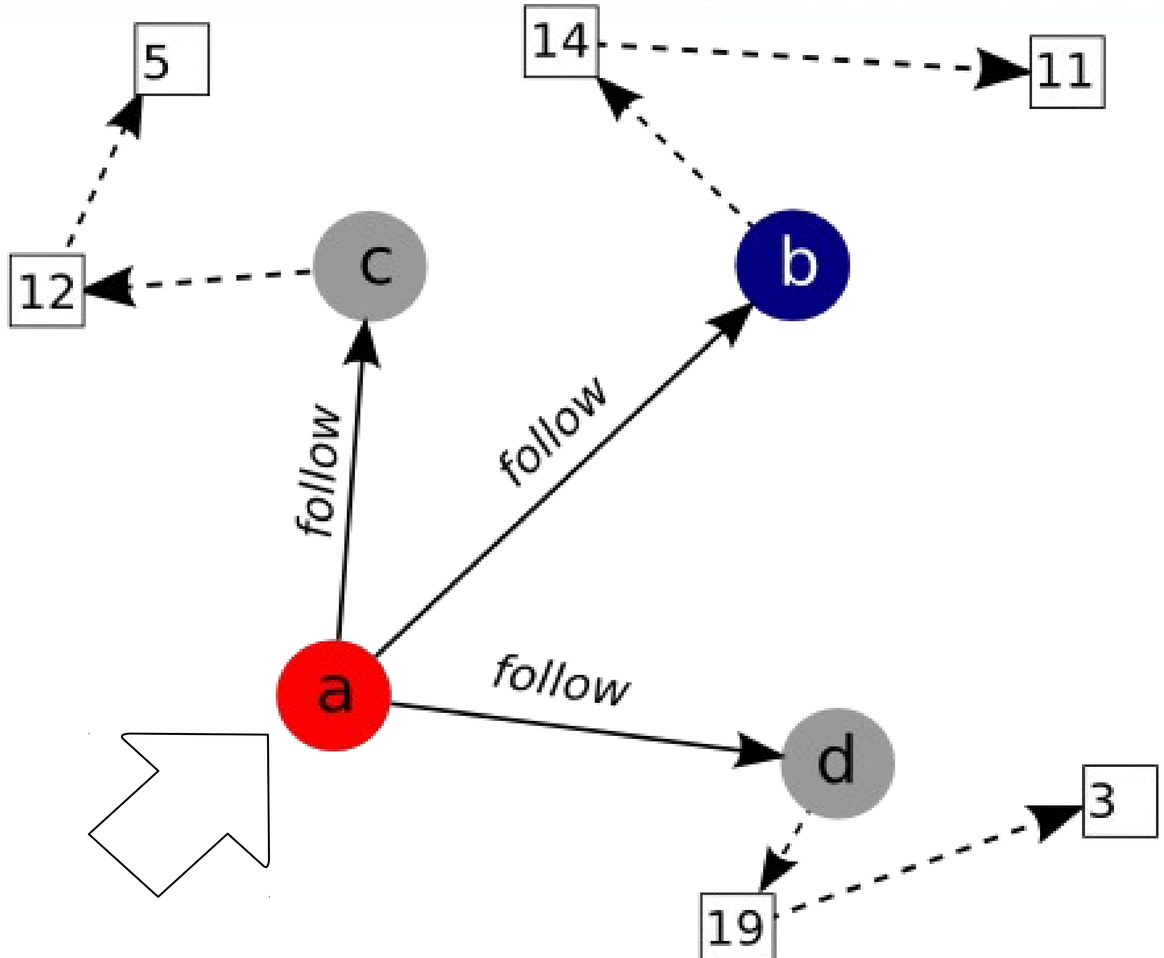
What are the pros and cons of this change?

Pros of this approach



- dynamic retrival possible (friendship graph may change)
  - very flexible data structure
- inserts and removes are very fast (all operations are O(1))

Contra: slow retrieval --> O(d log(d))

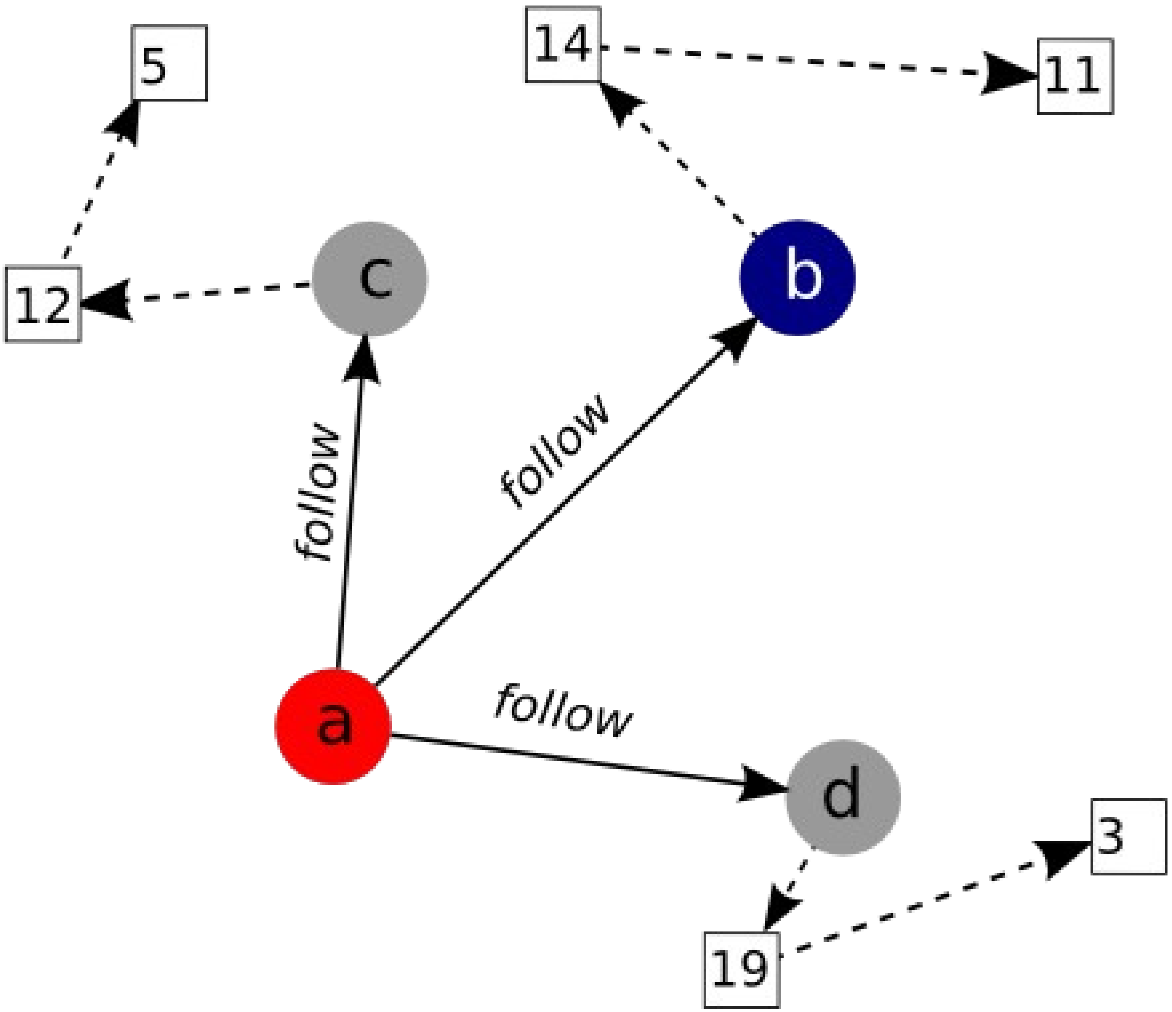


- unclear which edge to traverse first!
  - ==> entire ego network must be sorted
- Size *d of an* ego network is usually much bigger than the number of retrieved items *k*.
  - ==> Sorting seems to much effort

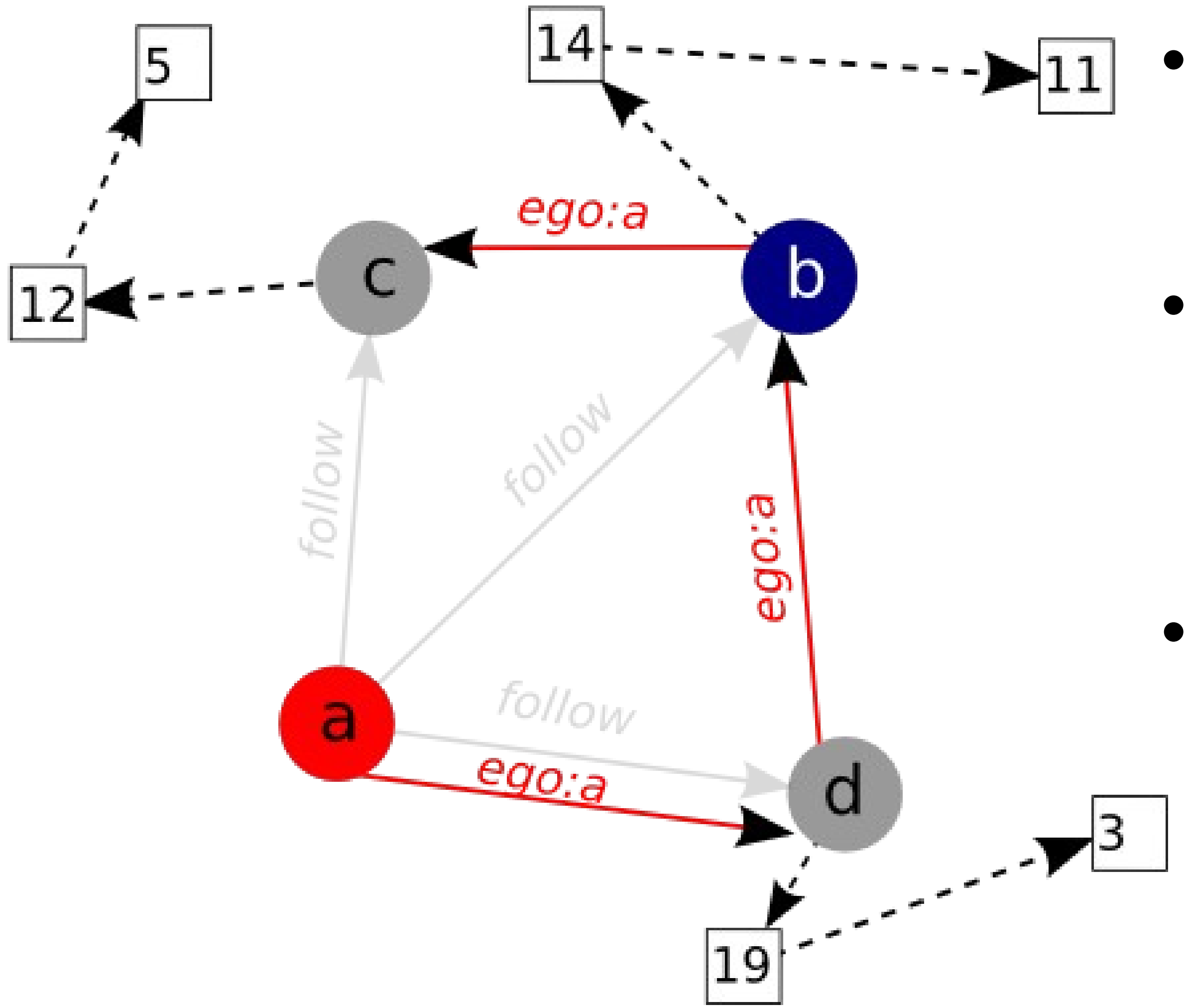
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

The key concept: going from star topology to lists



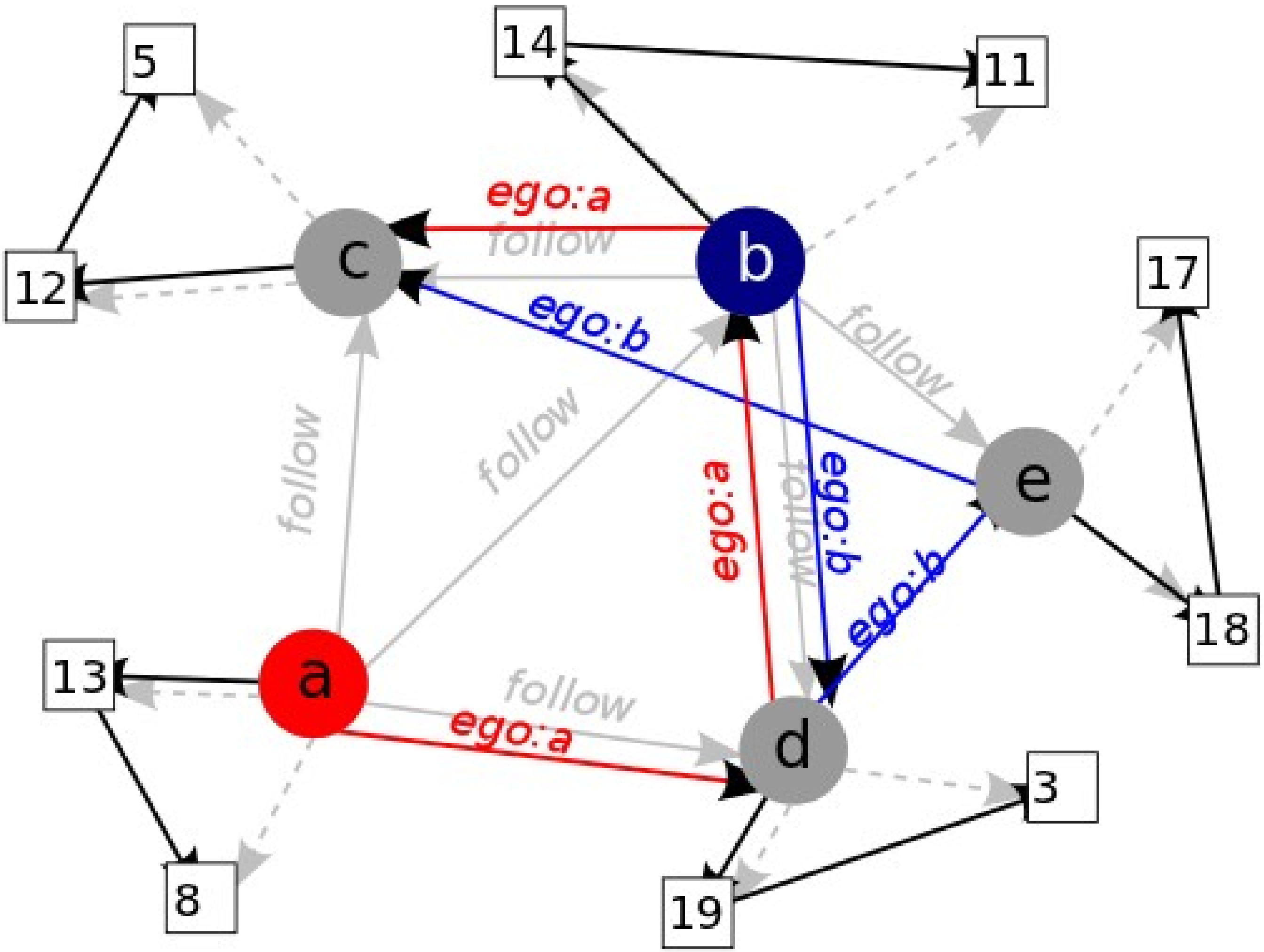
graphity index for the node "a"



Graphity rules:

- for every node (a & b) that follows others we create a linked list
- this linked list contains all the nodes that are beeing followed by this node.
- The followed nodes are sorted by the timestamp of their most recent content item

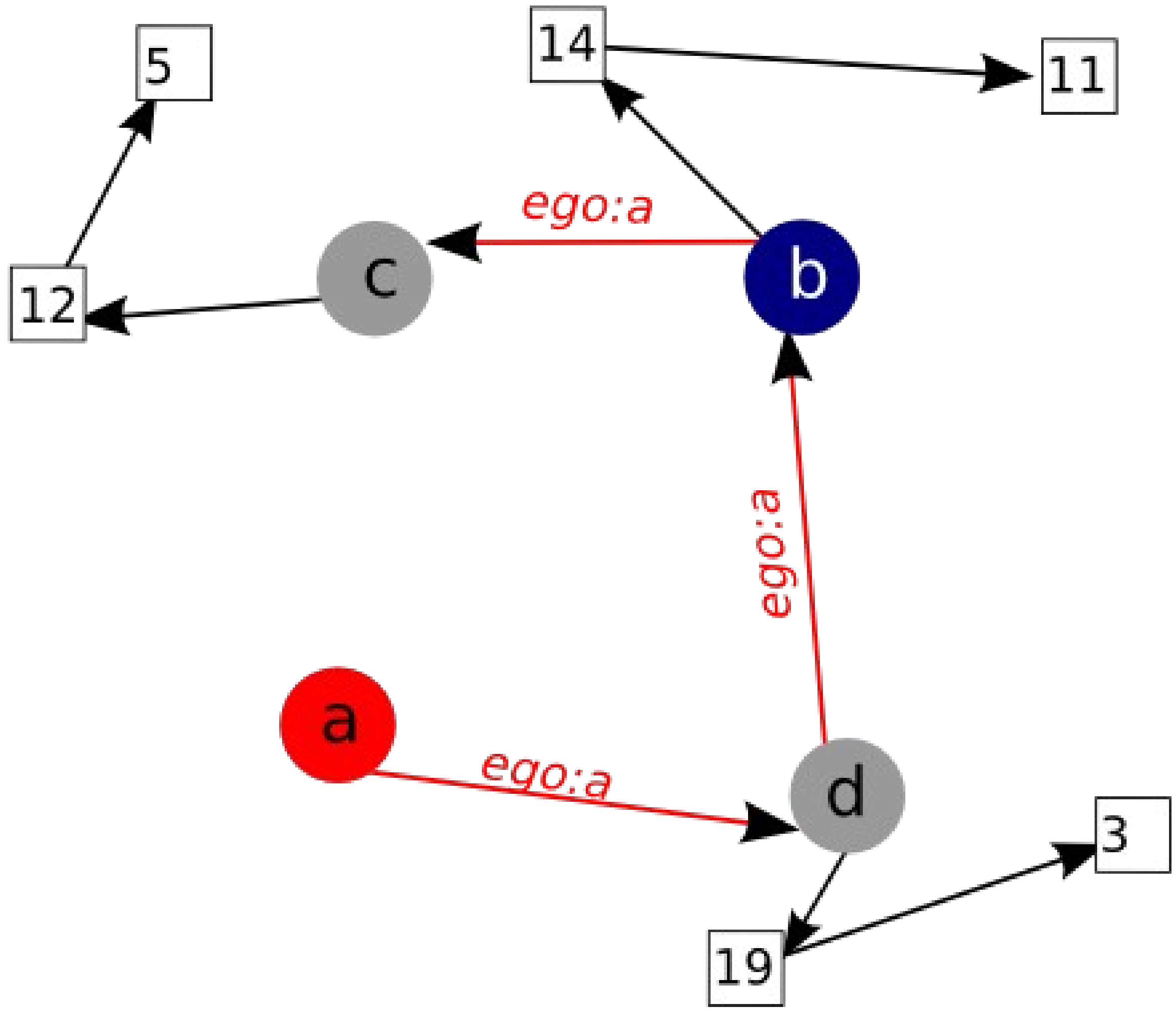
second graphity index for node "b"



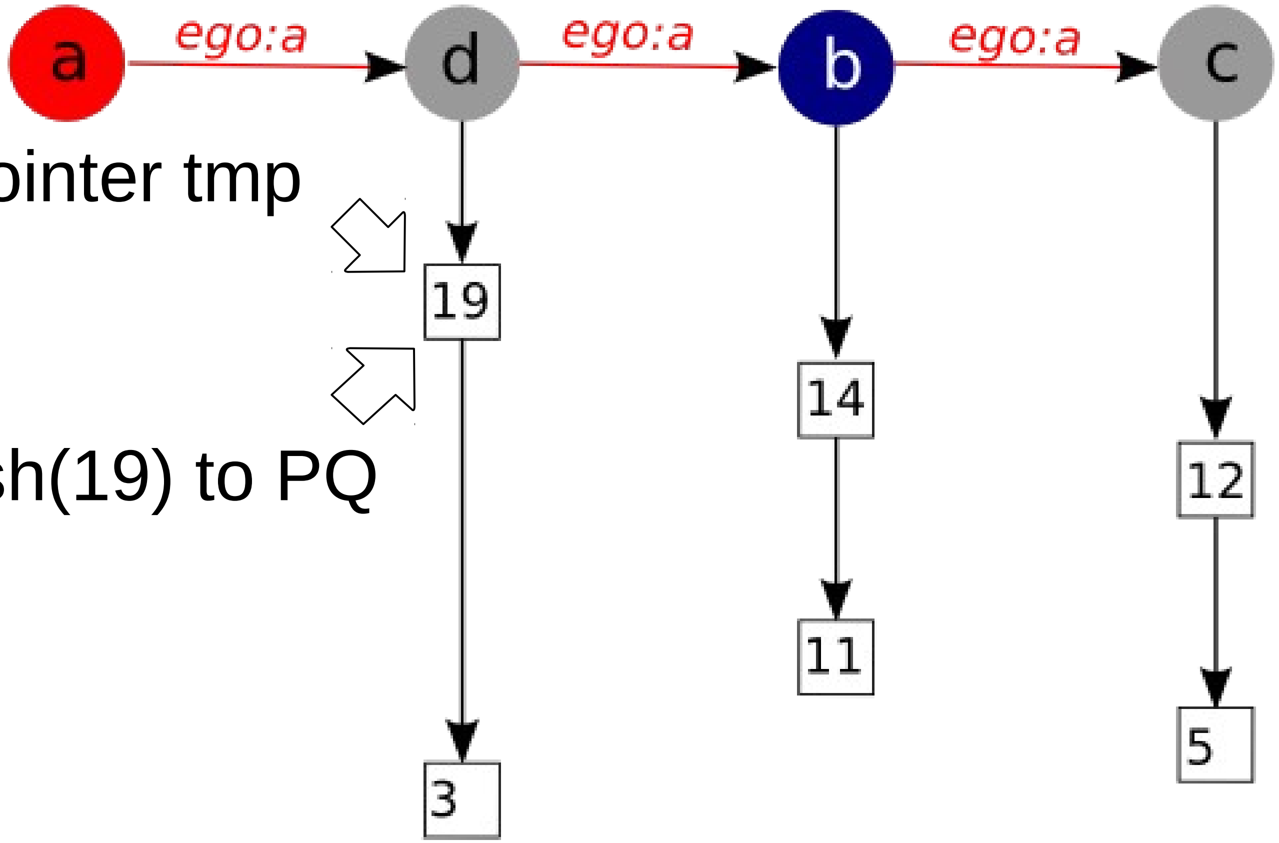
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

Rearanging the graph ...



top-k n-way Merge for retrieval in O(k log(k))



Pointer tmp

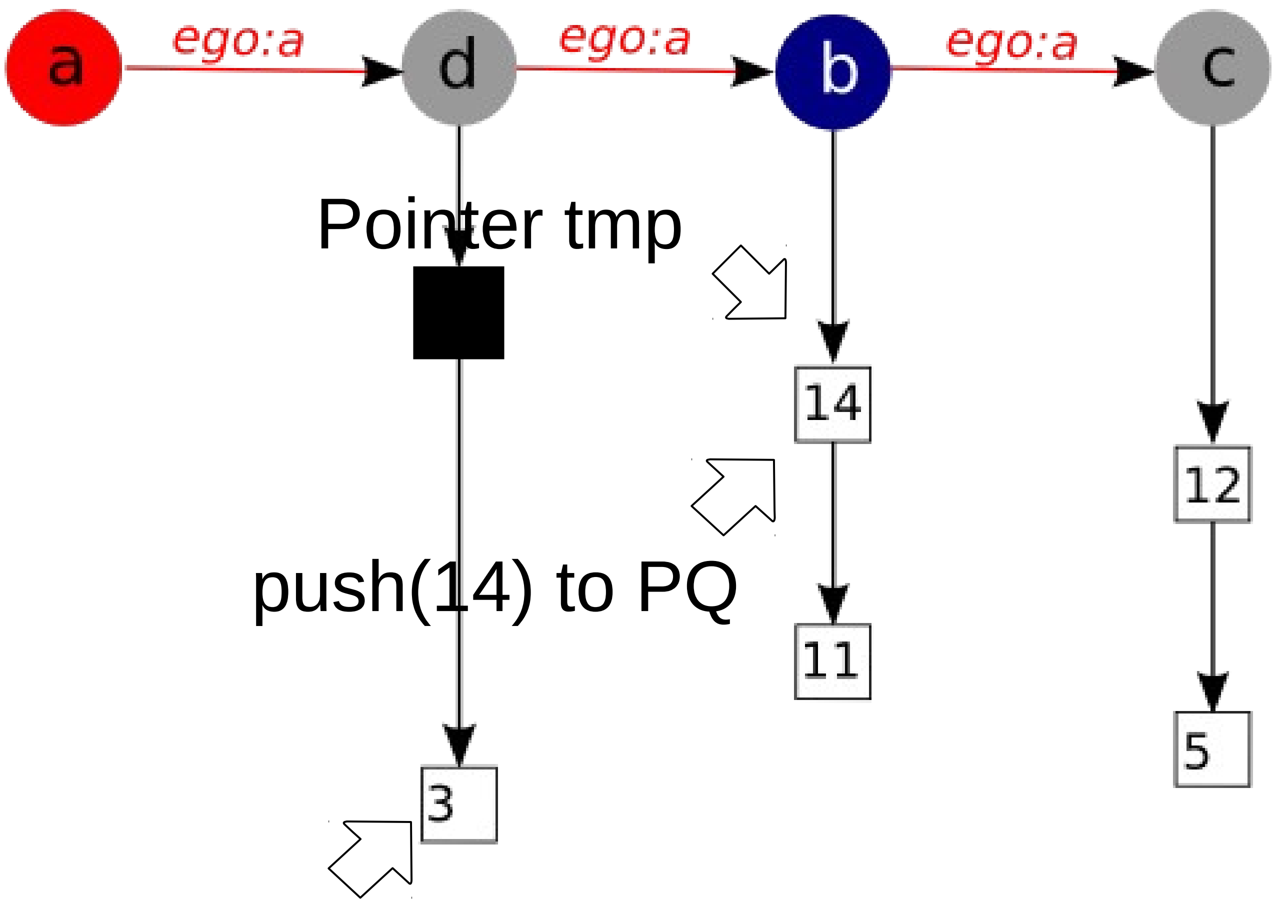
push(19) to PQ

Priority Queue:

19

Stream:

top-k n-way Merge for retrieval in O(k log(k))



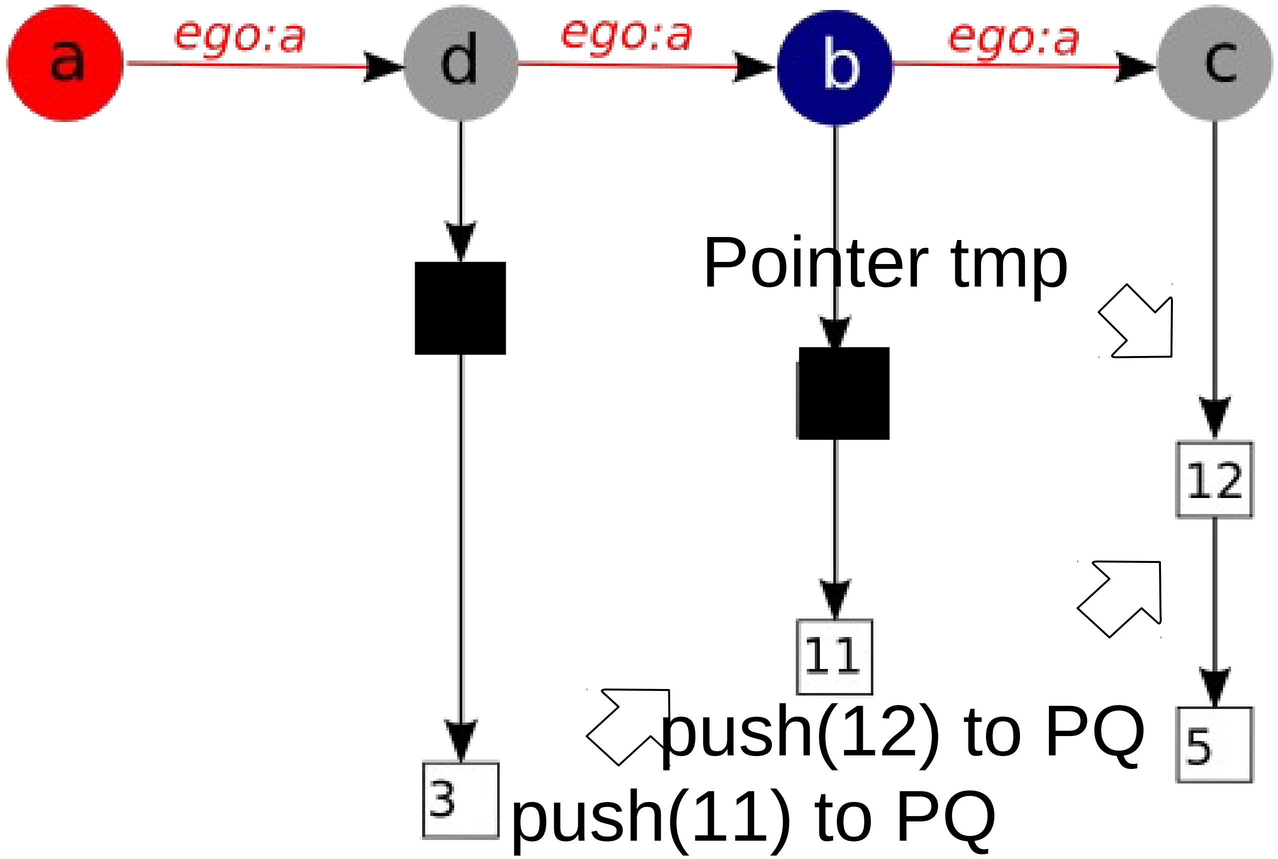
push(3) to PQ

Priority Queue:

14
3

Stream: (19,d)

top-k n-way Merge for retrieval in O(k log(k))

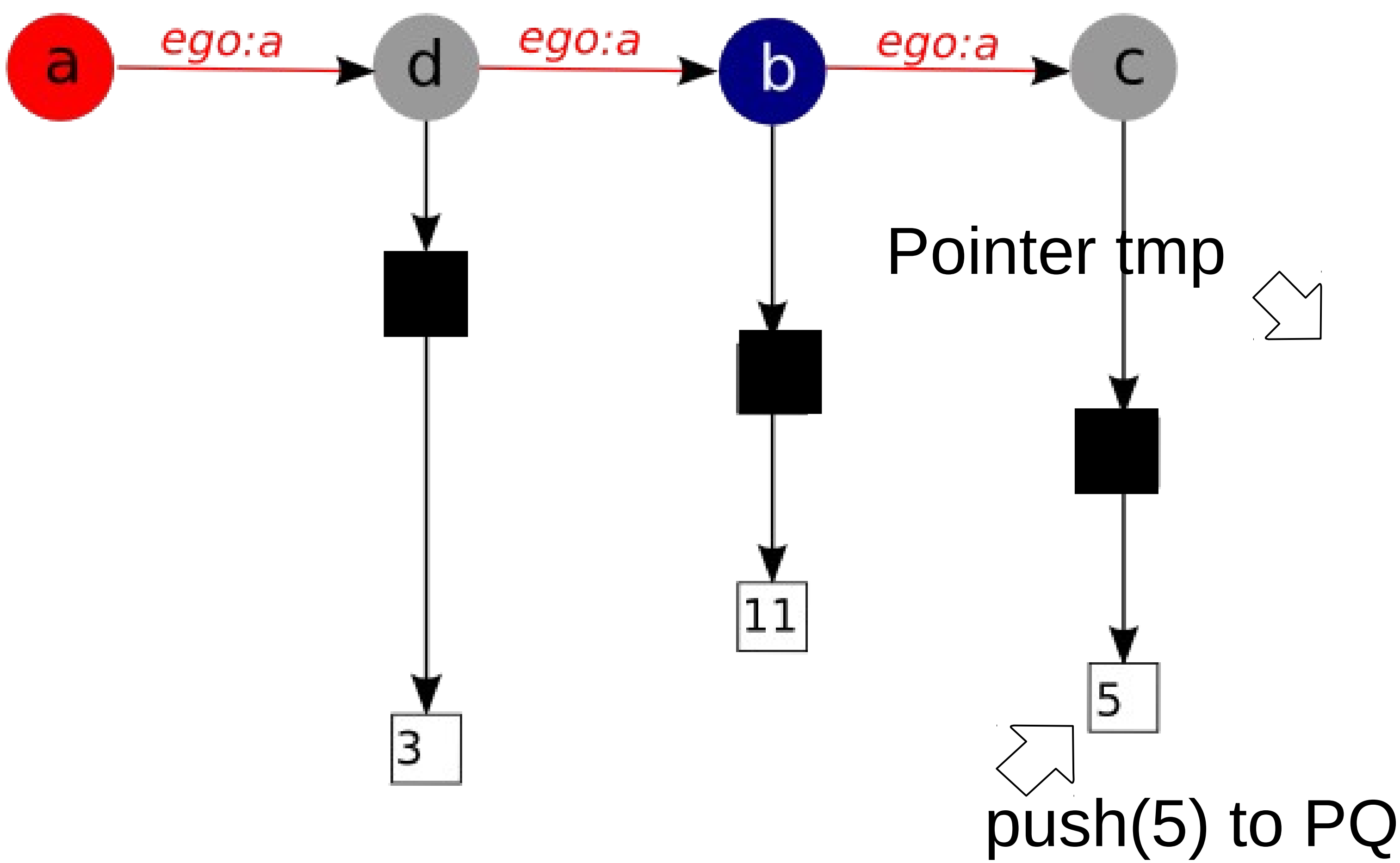


Priority Queue:

12
11
3

Stream: (19,d) (14,b)

top-k n-way Merge for retrieval in O(k log(k))

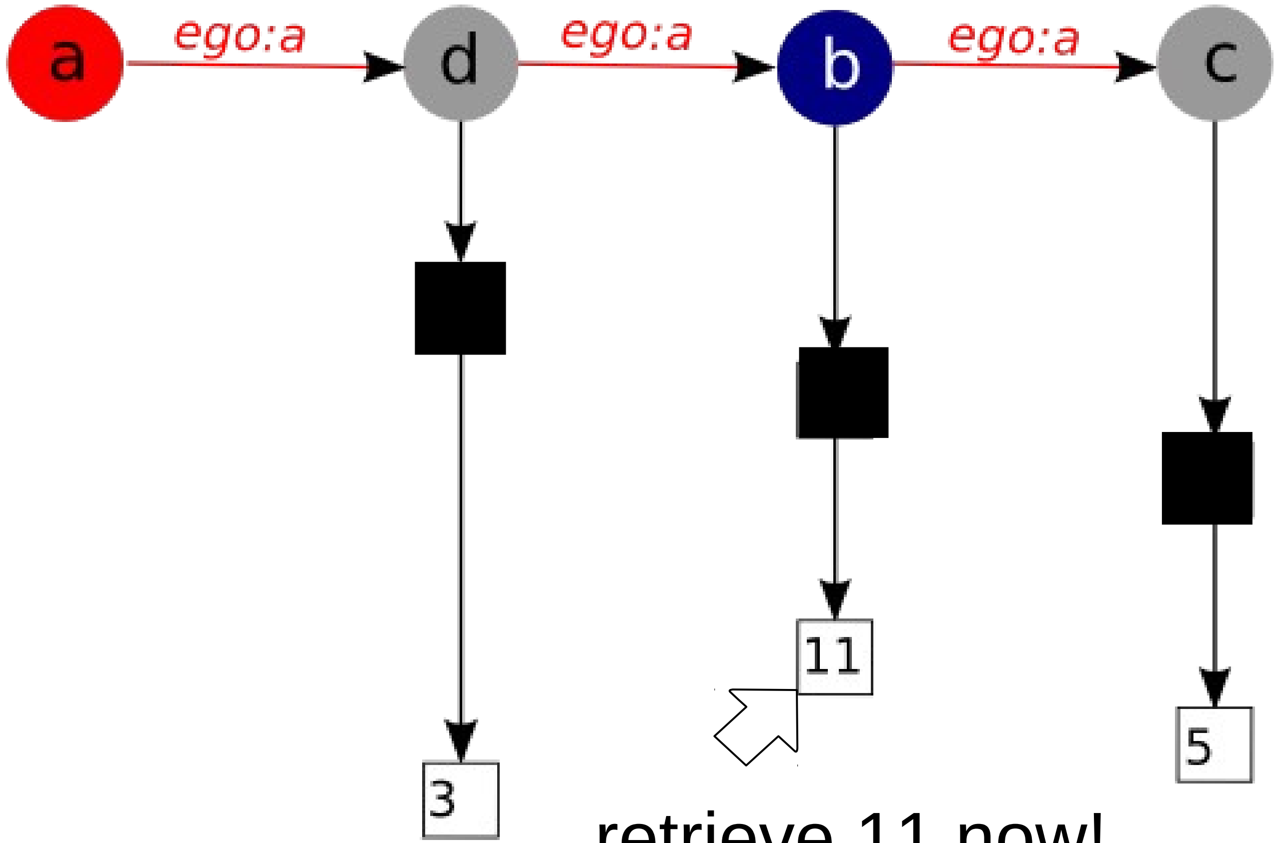


Priority Queue:

11
5
3

Stream: (19,d) ; (14,b) ; (12,c)

top-k n-way Merge for retrieval in O(k log(k))



retrieve 11 now!

Priority
Queue:

11
5
3

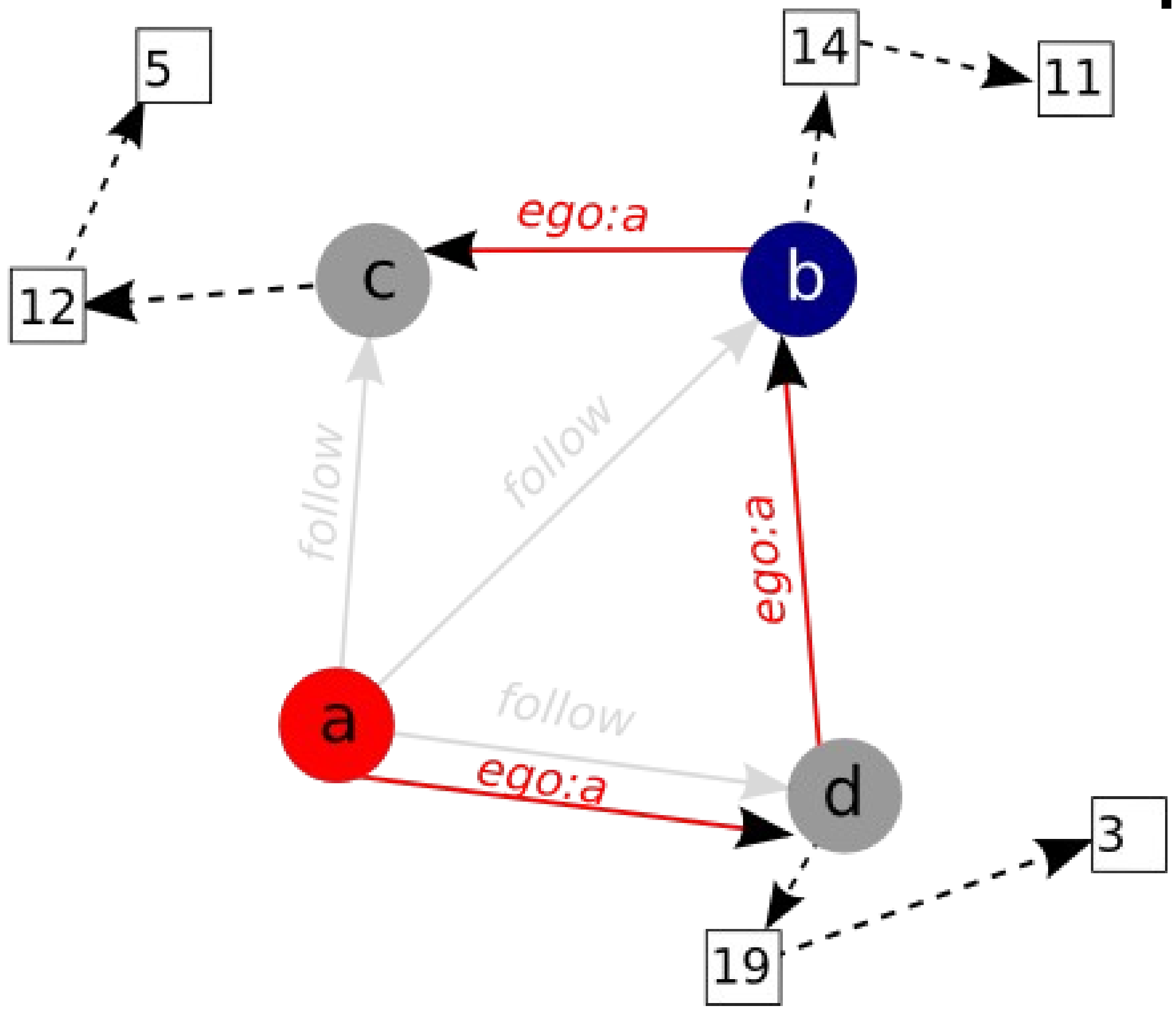
Stream: (19,d) ; (14,b) ; (12,c) ; ...

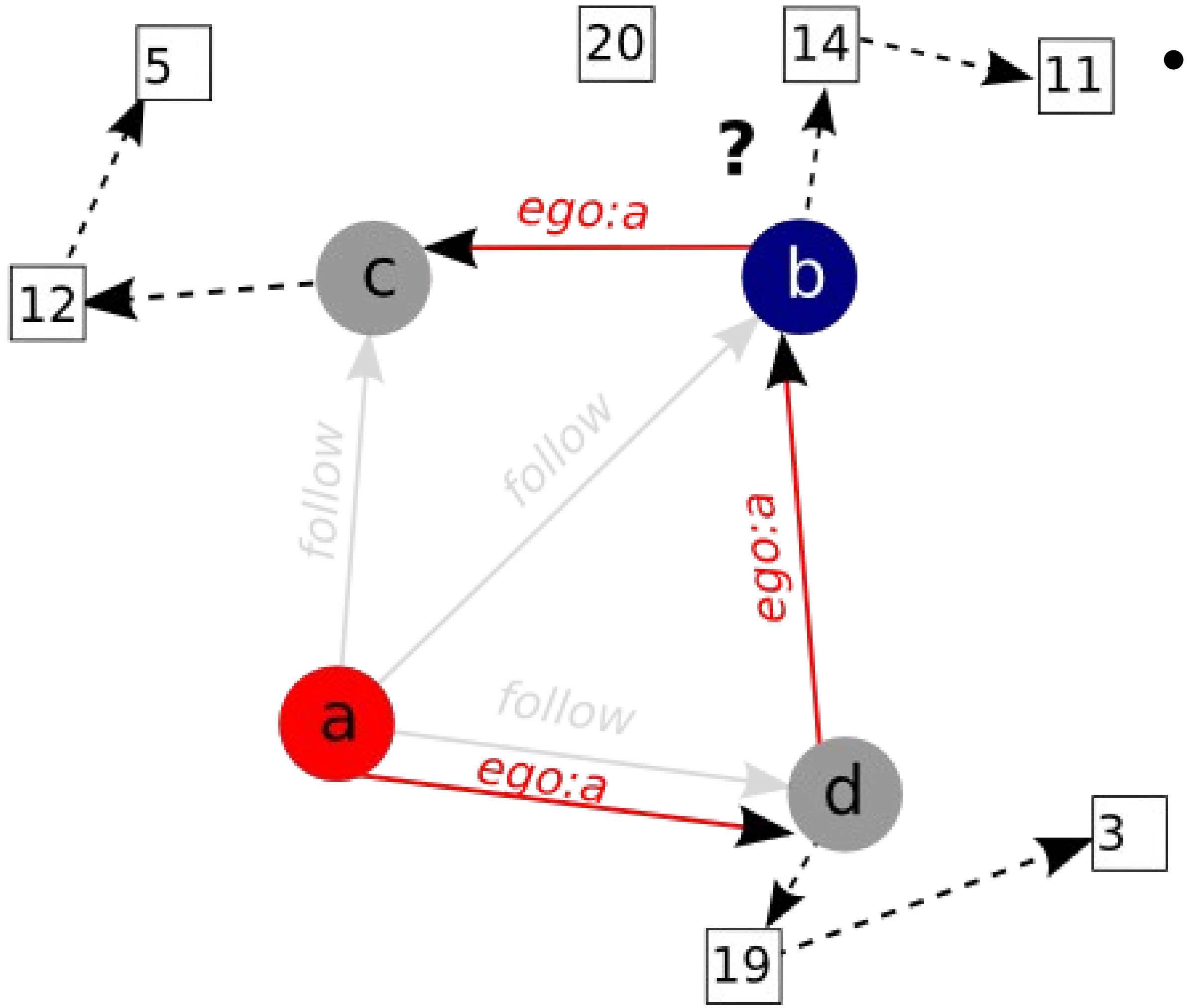
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

Creating new Content items

b creates a new content item

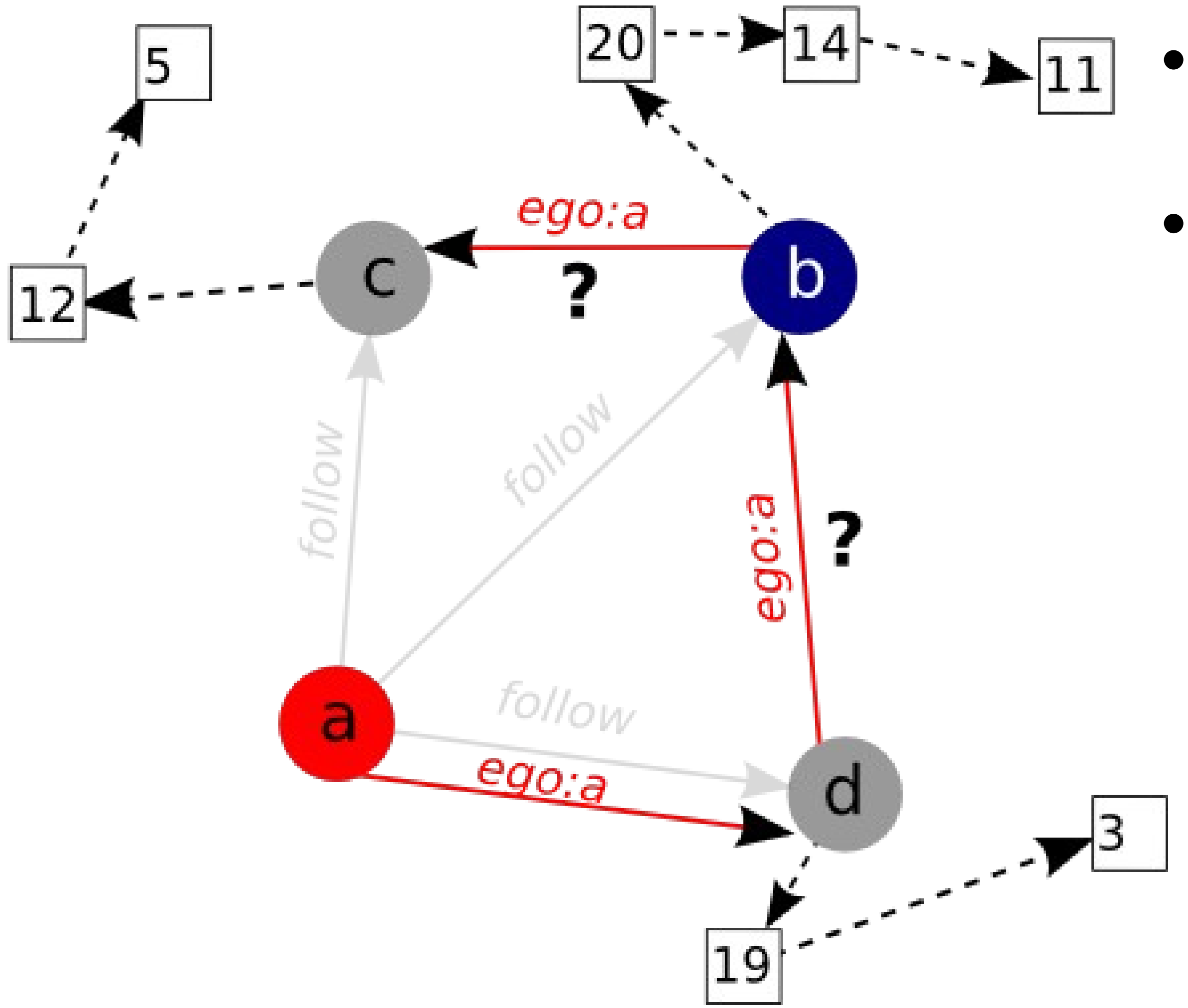




b created 20

- update linked list of b's content items

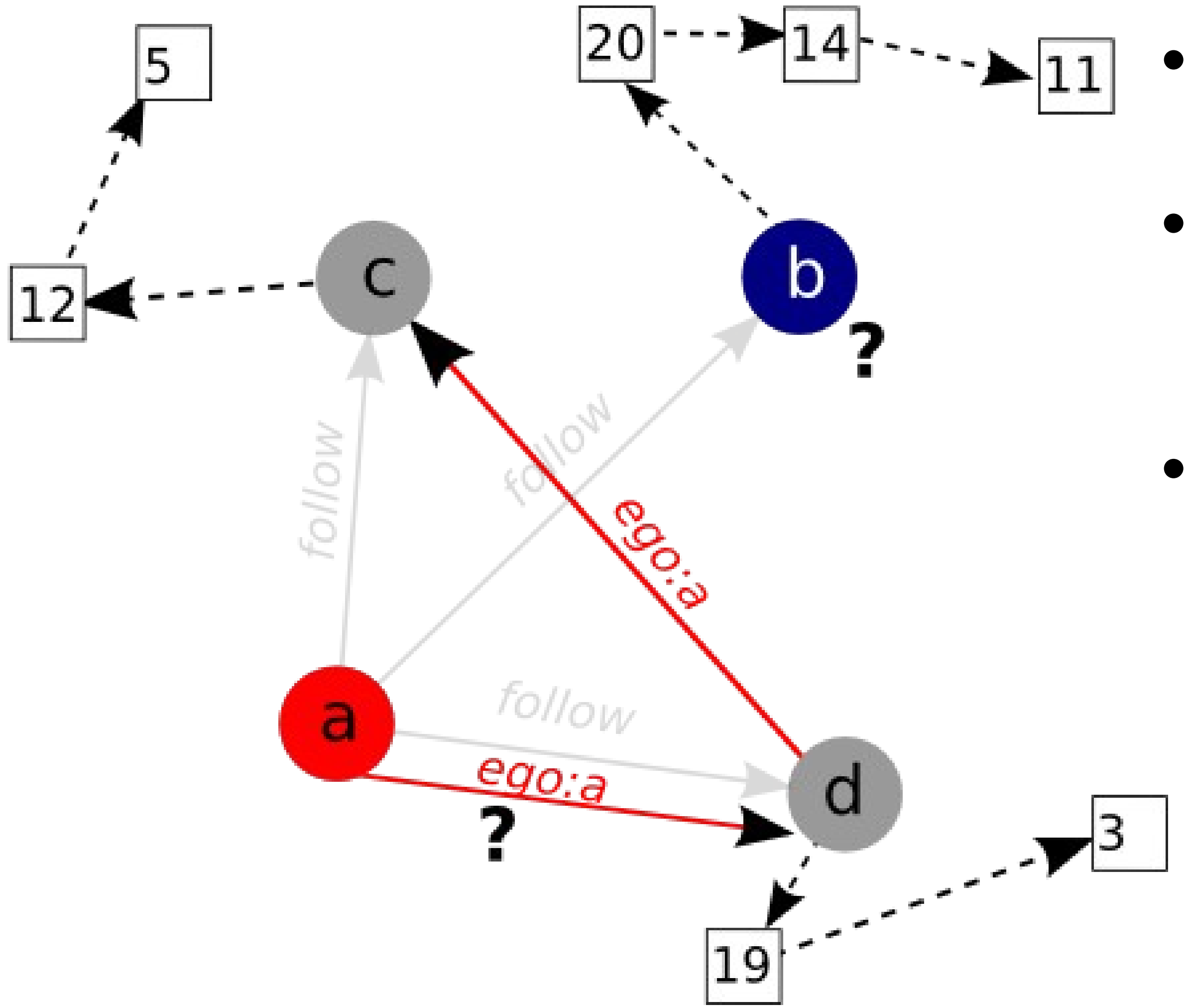
Creating new Content items



b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)

Creating new Content items

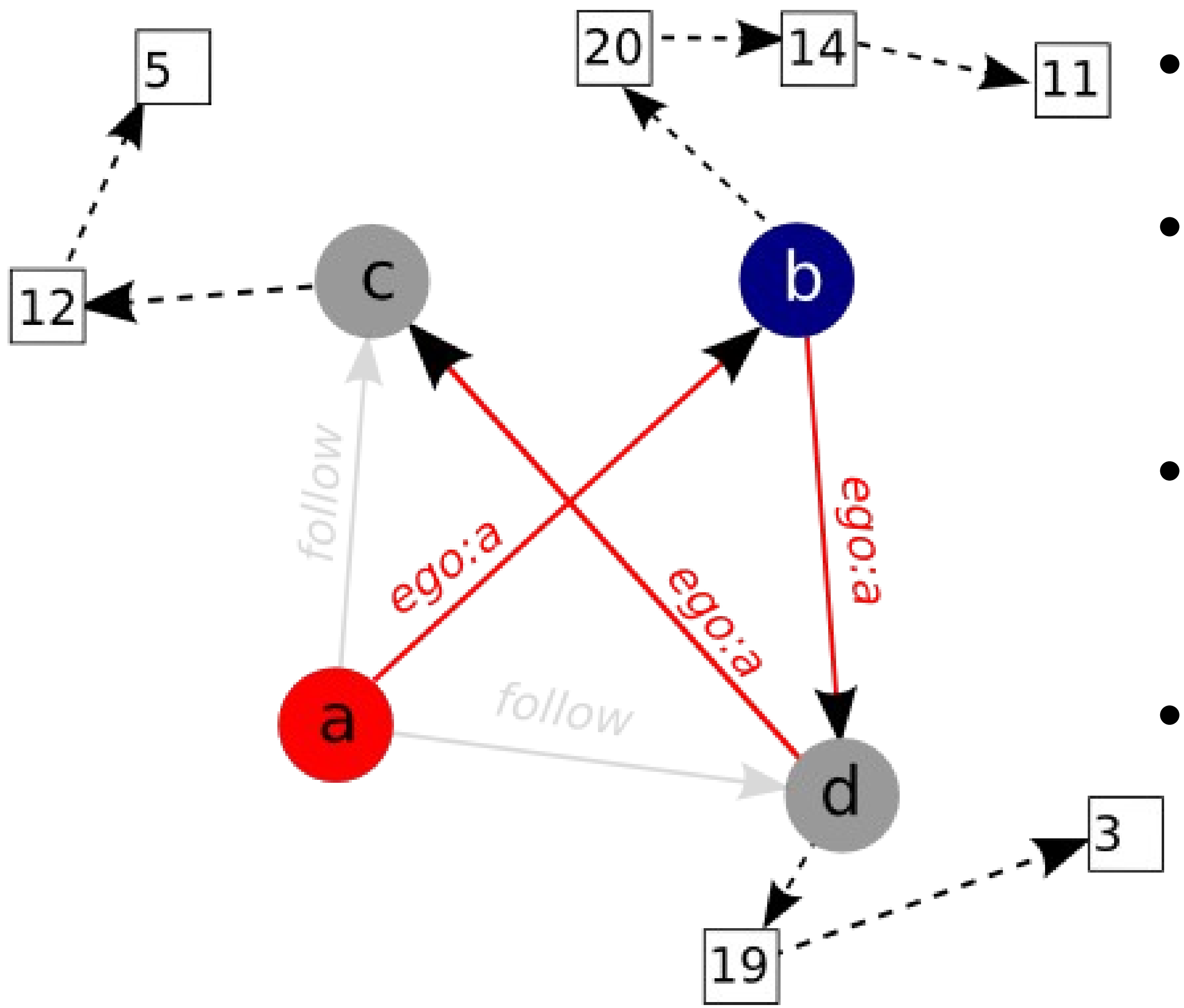


b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)
- interlink b's predecessor and successor

Rene Pickhardt
rpickhardt@uni-koblenz.de

Creating new Content items



b created 20

- update linked list of b's content items
- now look in which ego networks b is member of. (our case just a)
- interlink b's predecessor and successor
- user the follow edge from a to b and the first ego:a to insert b in the beginning of ego:a

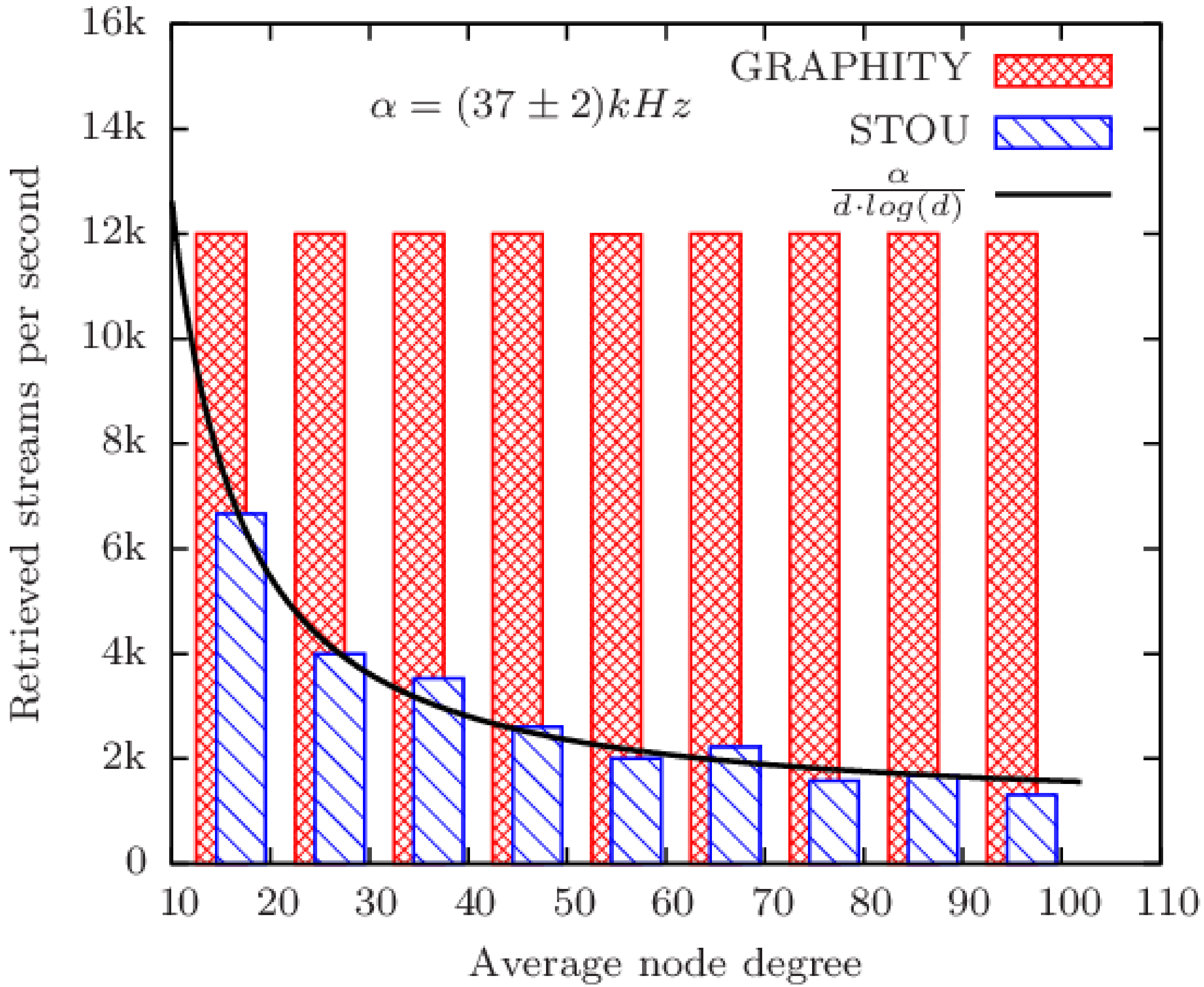
Index

- Introduction to the newsfeed problem
- Why relational Data bases won't do the job
- The construction and idea of STOU
- The construction and idea of graphity
- Example 1: retrieval of news feeds (top-k n-way merge)
- Example 2: Creating new Content Items
- Evaluation on Wikipedia data set.

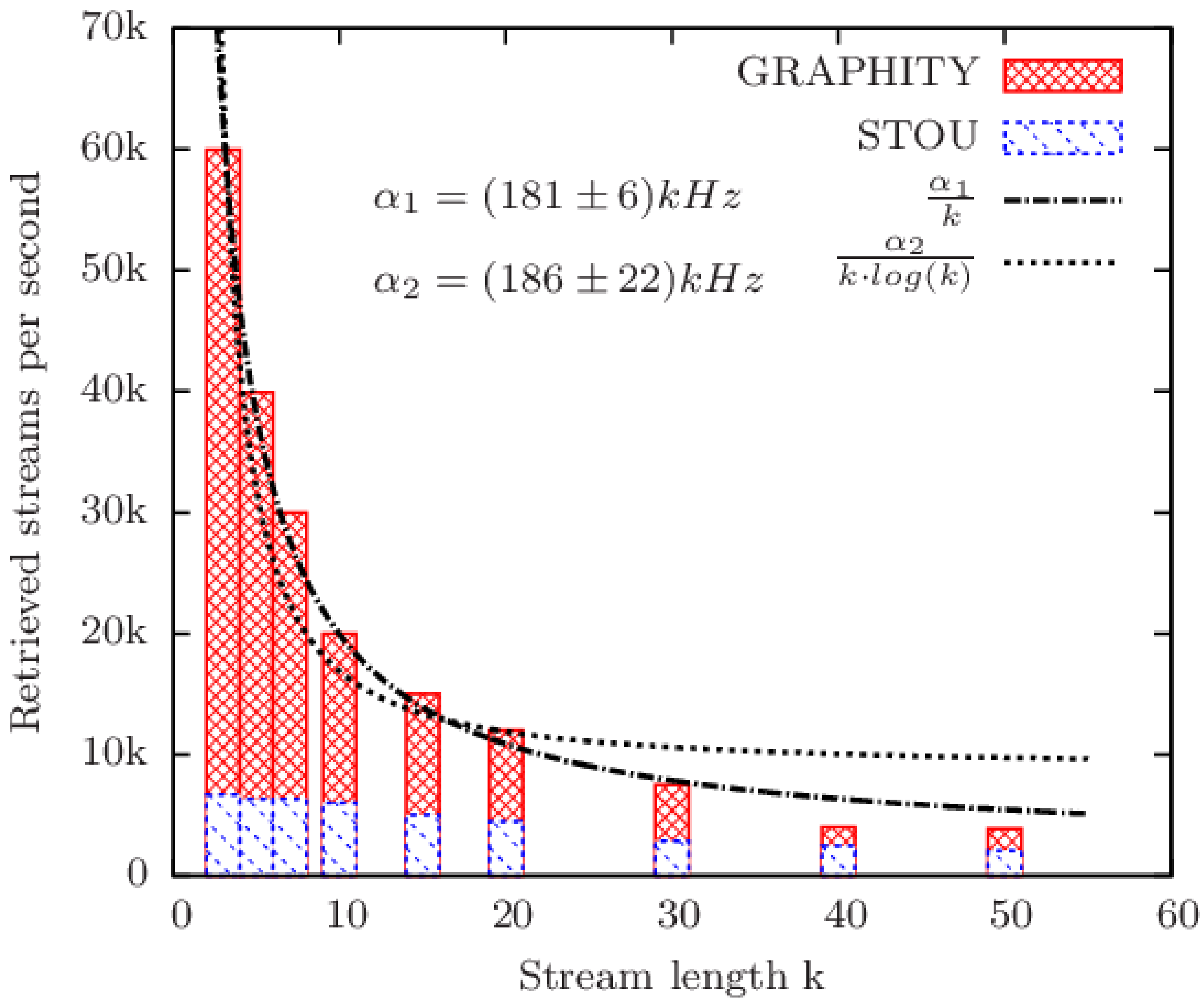
Wikipedia as a social network

- Every article ==> User
- Every link in an article ==> Follow relationship
- Every Revision of an article ==> Status update of a user
- Remark: if in a new revision the outlinks of the wikipedia article change
  - We don't take this as a status update
  - we interpret this revision as a change to the friendship graph

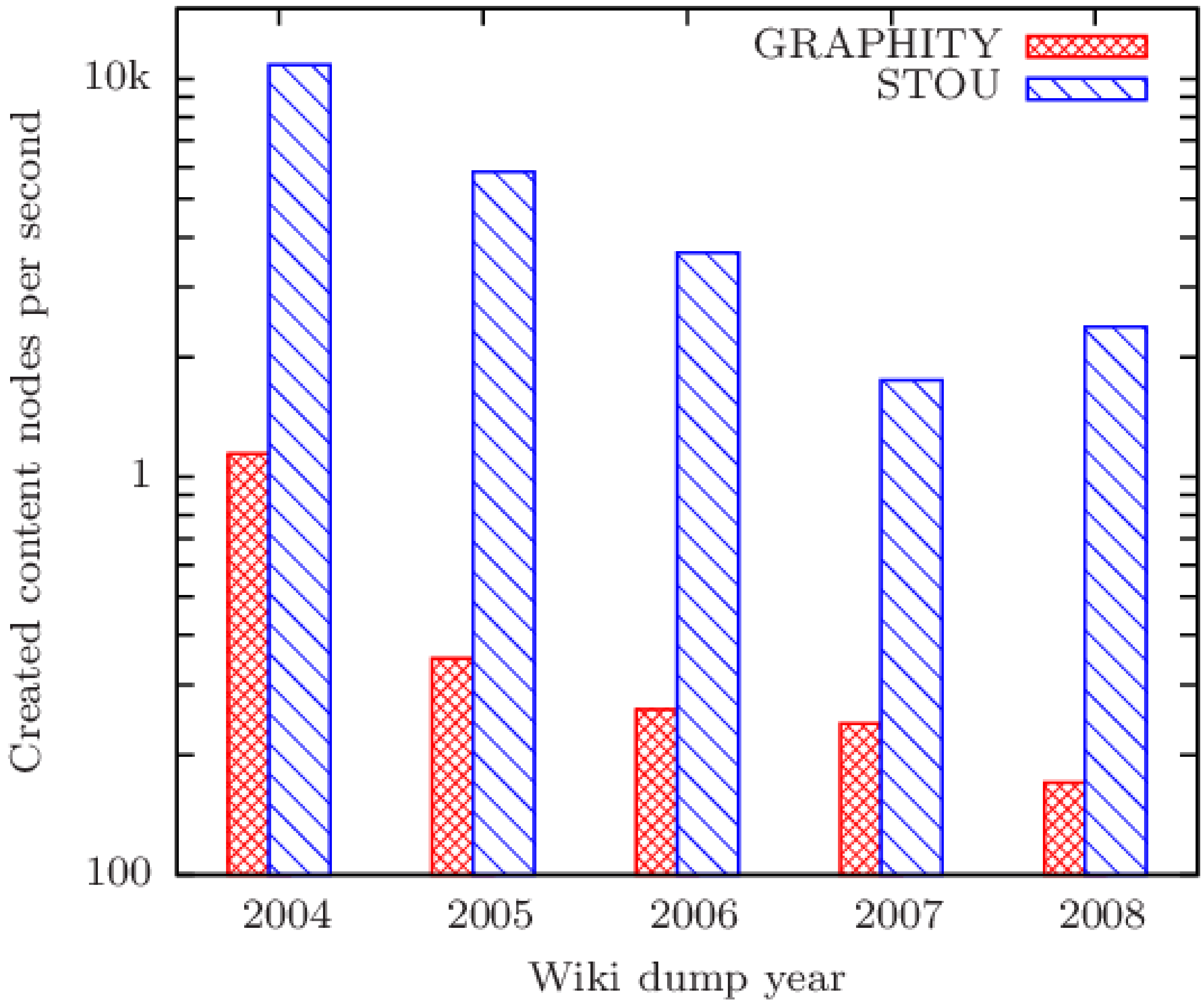
demonstrating independence of node degree



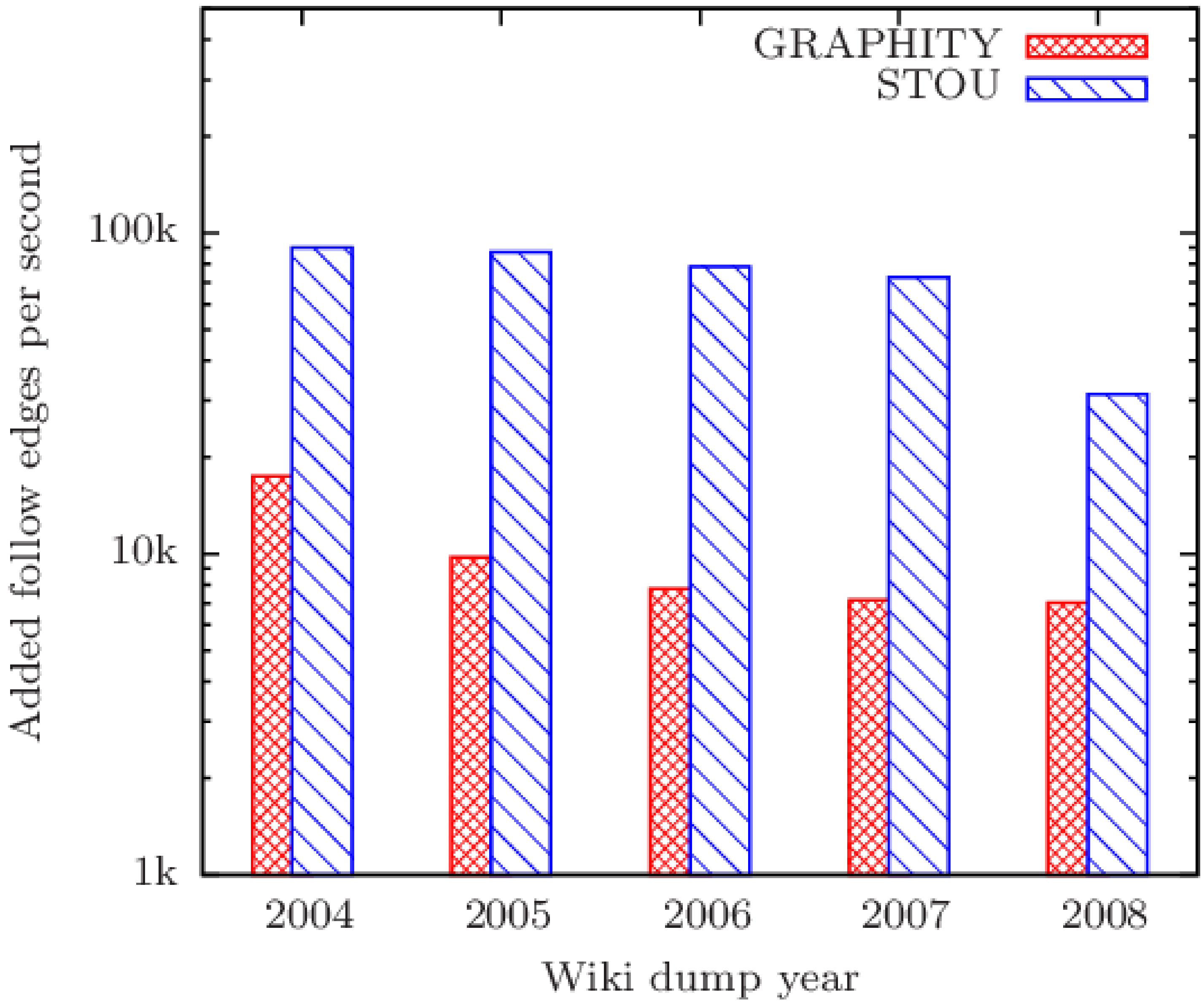
demonstrating linear dependence of k



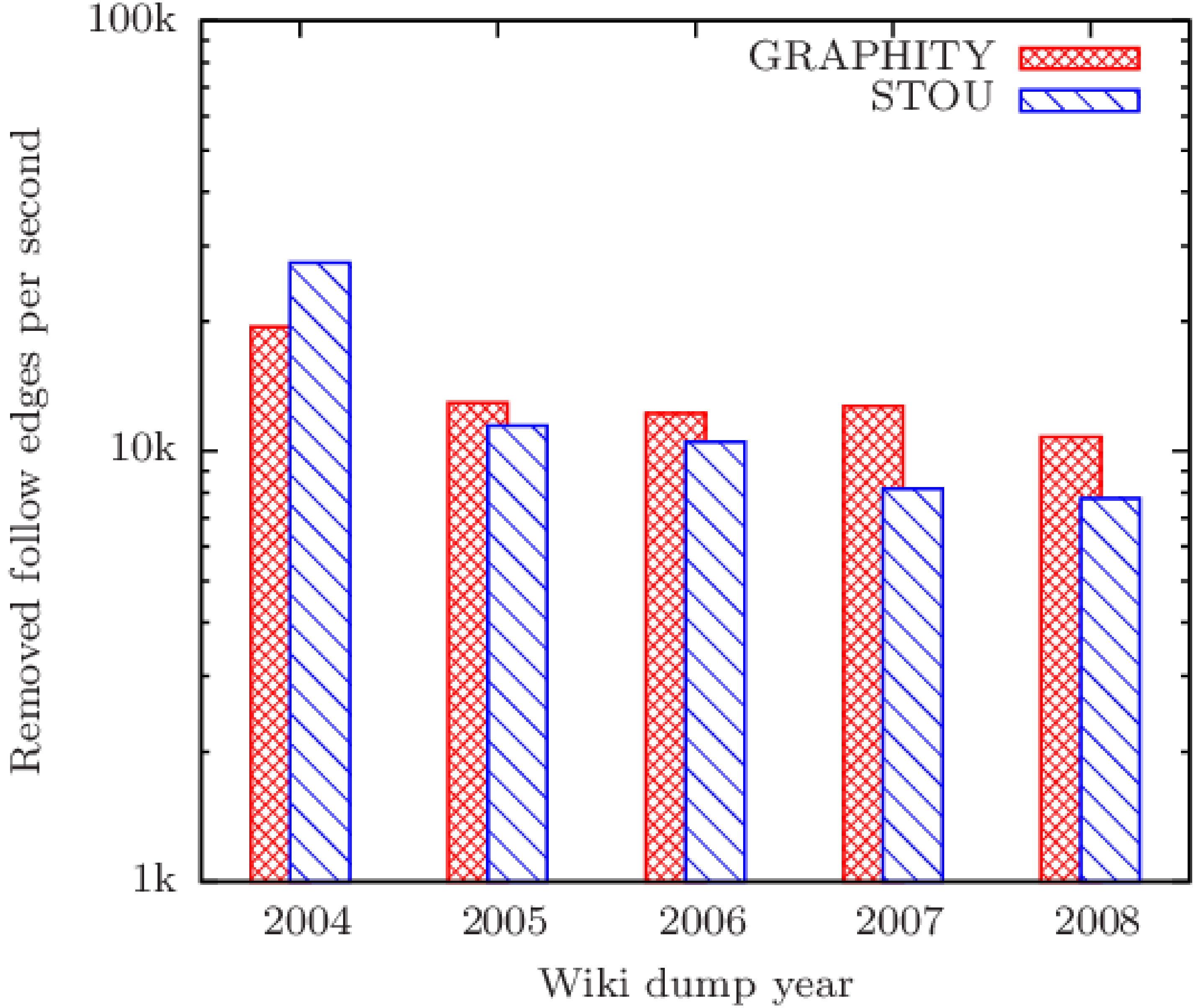
Index maintaining - inserting new content items



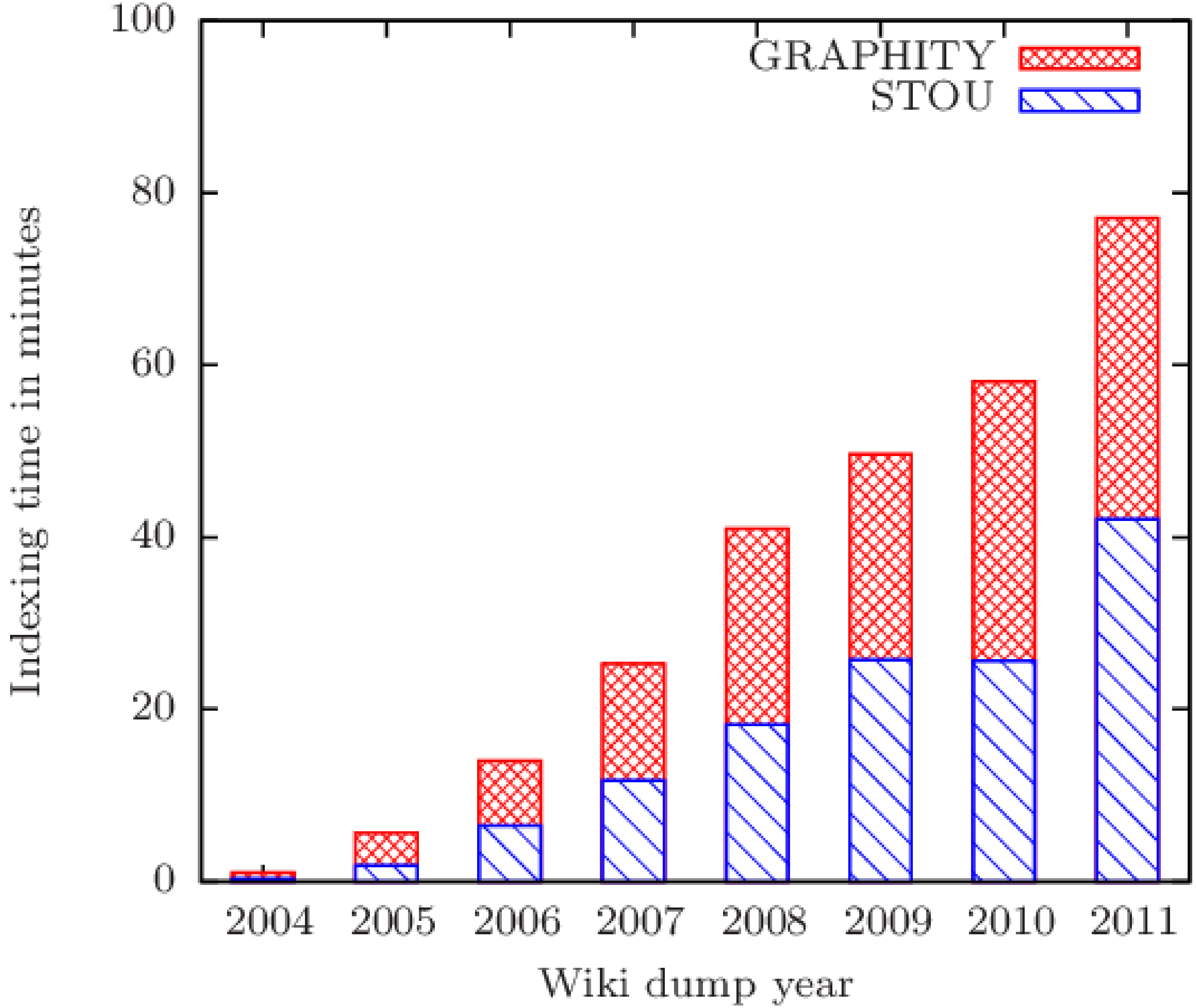
updating graphity for new friendship relations



updating graphity if friendships break



time to build the index



Conclusion

- We built two graph indices for top-k news feed retrieval
- STOU is fastest in writing operations (with moderate reading speed)
- Graphity is faster in retrieving operations
- Empirical study on a graphs with up to 2 mio. Users, 32 mio. follow relationships and 50 mio. content items shows that graphity even performs better than the theoretical runtime
- Especially for graphity we saw
  - retrieval of social news feeds of k items in O(k log(k))
  - Almost as good as redundant content lists
  - But no redundancy in content data

So which one to take?

42

Thank you for your attention

More information + Slides on:
http://www.rene-pickhardt.de/graphity

Thanks to

- Mattias Persson and Peter Neubauer from neotechnology.com
- the neo4j community on the neo4j mailinglist for helpful advices
- Knut Schumach for coming up with the name GRAPHITY
- Matthias Thimm & Leon Kastler for helpful discussions

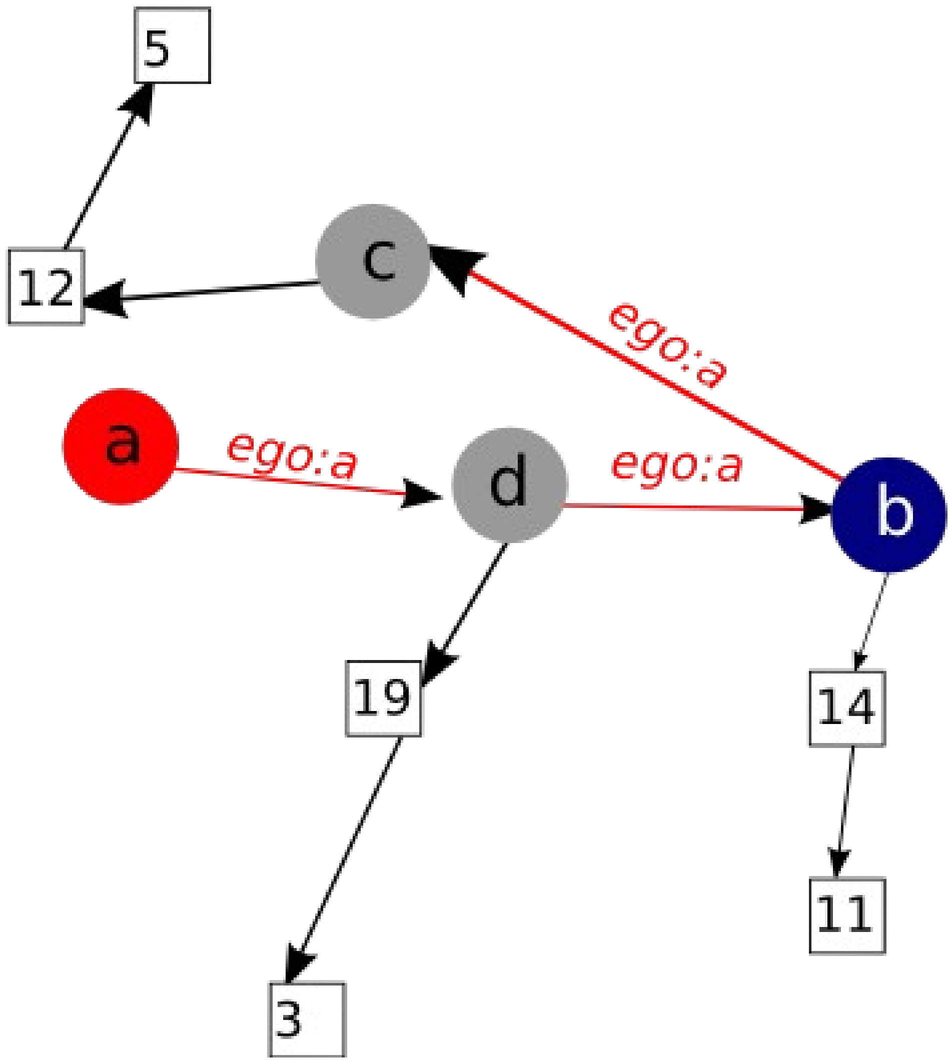
This project is founded by the EU Projects Social Sensor and ROBUST.

Sorce code & data sets on:
http://www.rene-pickhardt.de/graphity-source-code/

Backup slides

- Backup slides

Rearanging the graph...



Updates need to be done in the following situation

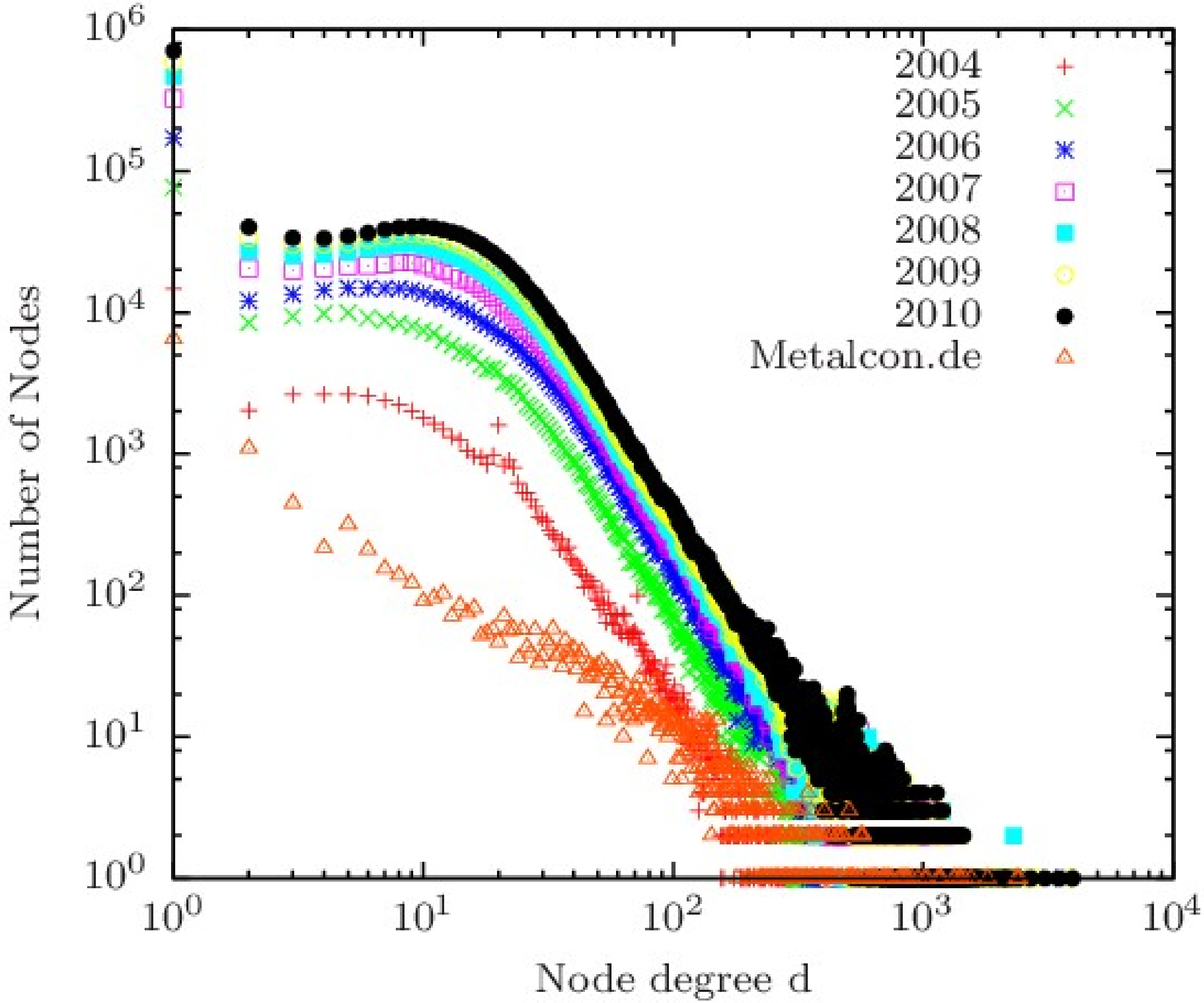
Updates need to be done in the following situations

- **new created content item** (O(d))
  - index of every follower needs to be updated
- **new created follow relation** (O(d))
  - index of follower needs to be updated
- **friendship relation breaks** (O(d))
  - index of the former follower needs to be updated
- **most recent content item of a user is deleted** ($O(d^2)$)
  - index of every follower nees to be updated

Future work

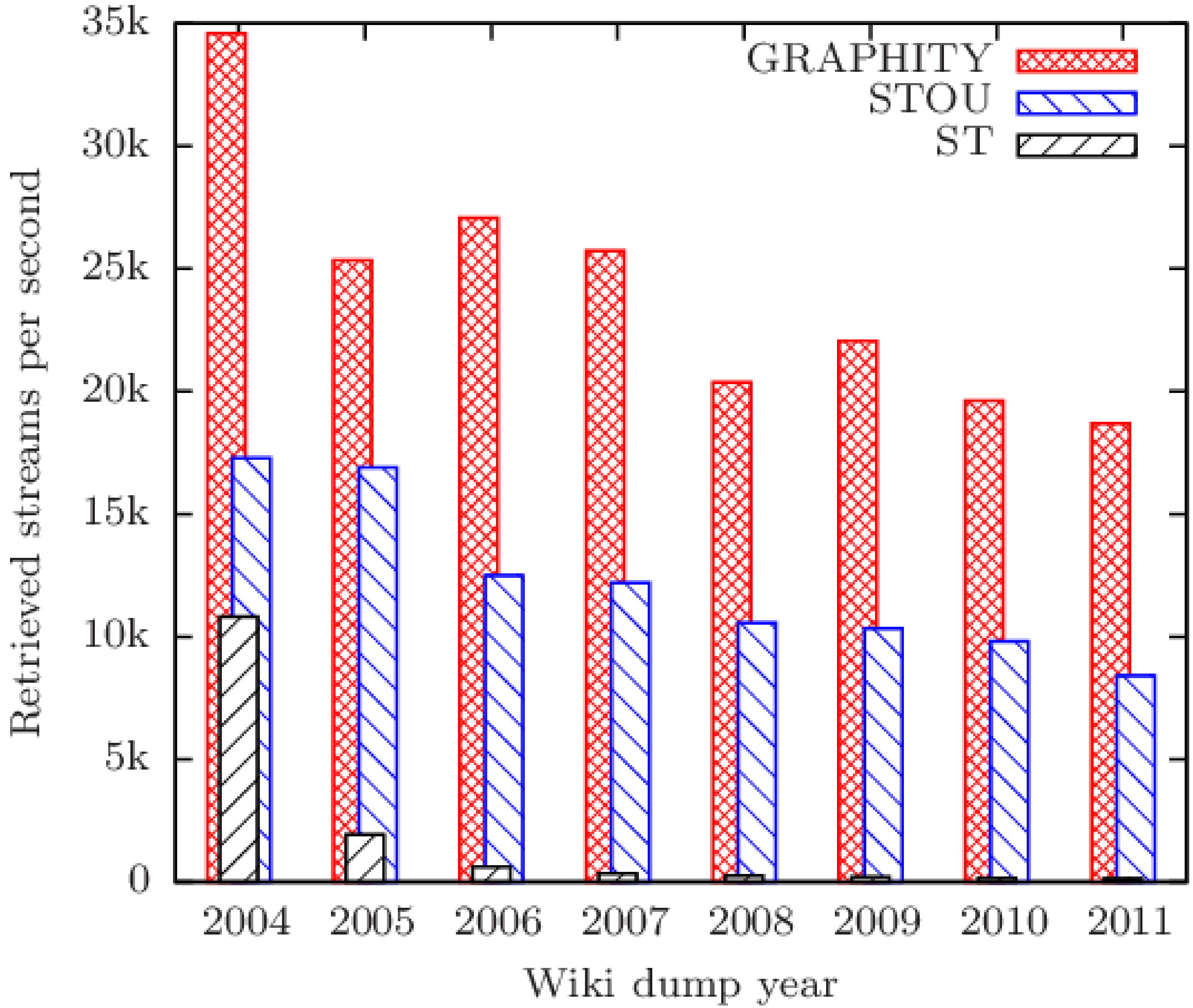
- Generalize / built theory on top-k joins
- Distributed system
- Partially do graphity index
  - (ever update only yields updating a constant number of graphity indices)
- Tie strength (filtering / ranking)

Evaluation Characteristics of data sets

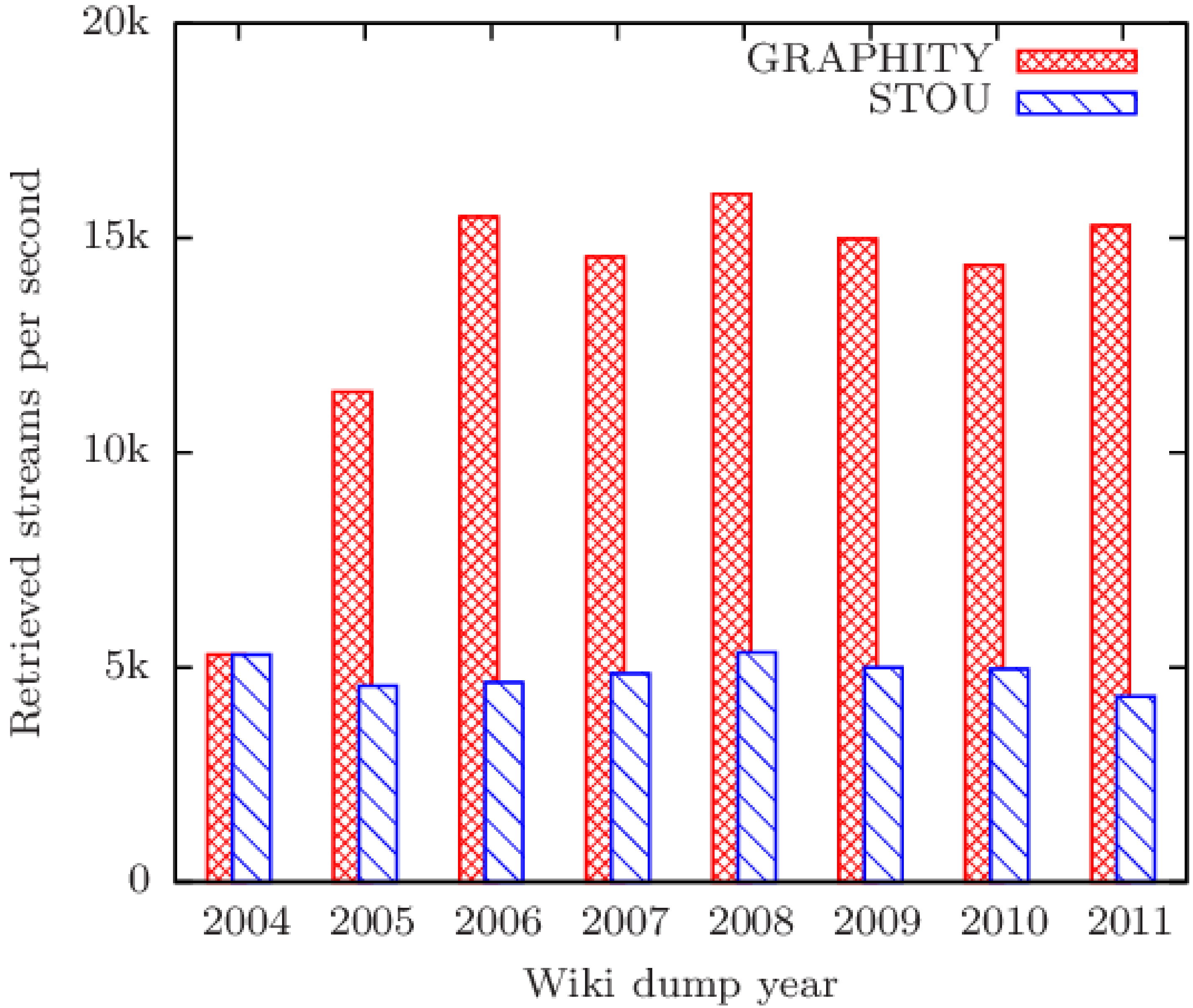


Rene Pickhardt
rpickhardt@uni-koblenz.de

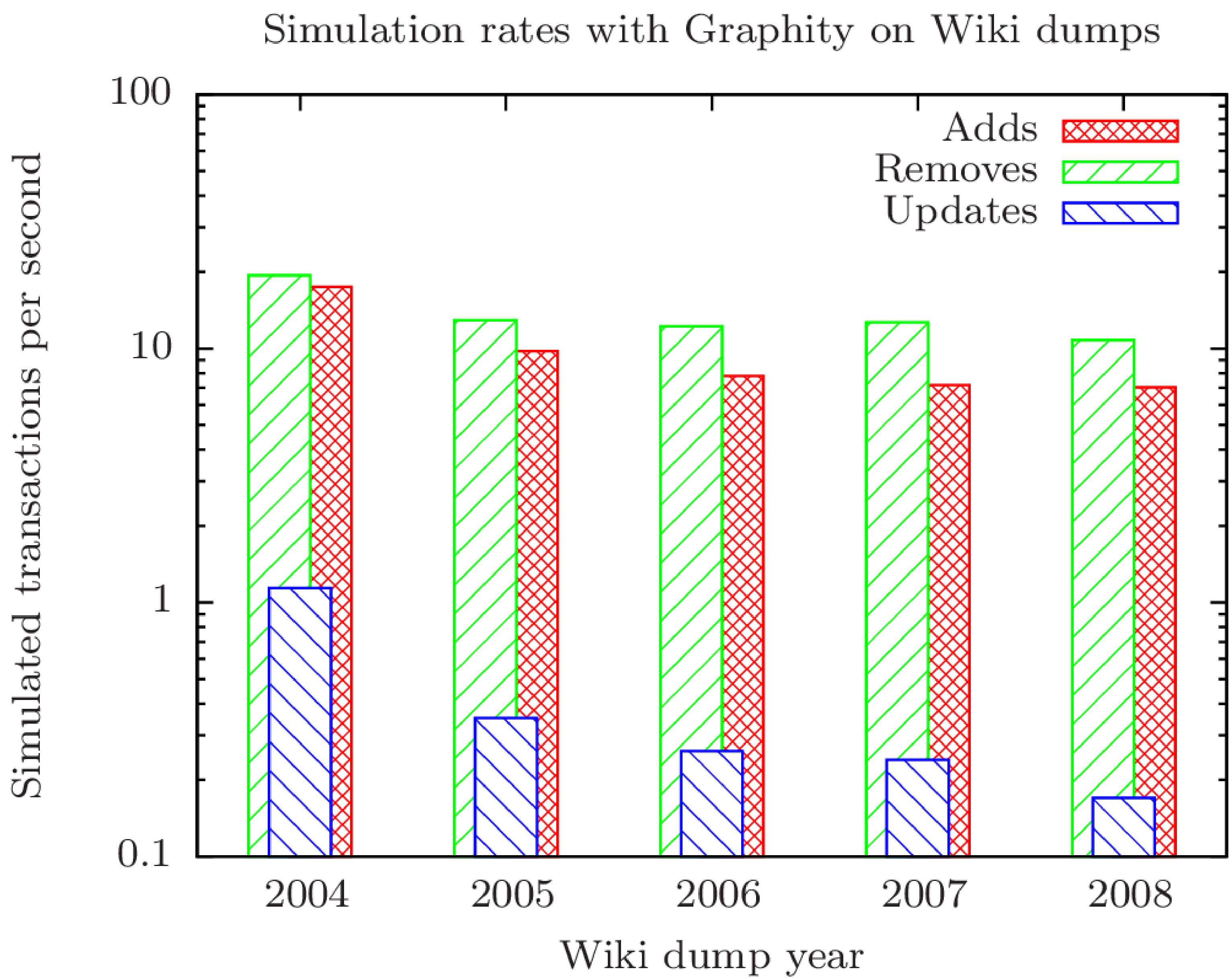
Retrieving streams on all data sets



Retrieving streams Node degre > 10



Simulating a social network



Summary - We created a graph model with:

- fast retrieval of social news feeds of k items in O(k log(k))
- dynamic retrieval method
- no redundancy in content data
- Creating new Status Updates yields updating of d graphity indices of following nodes
- Each Graphity index update is O(1)

We also conducted an evaluation of a graph with :

- ~ 2 mio. users
- ~32 mio. follow relations
- ~50 mio. Status updates

giving empirical proof of our theoretical findings.