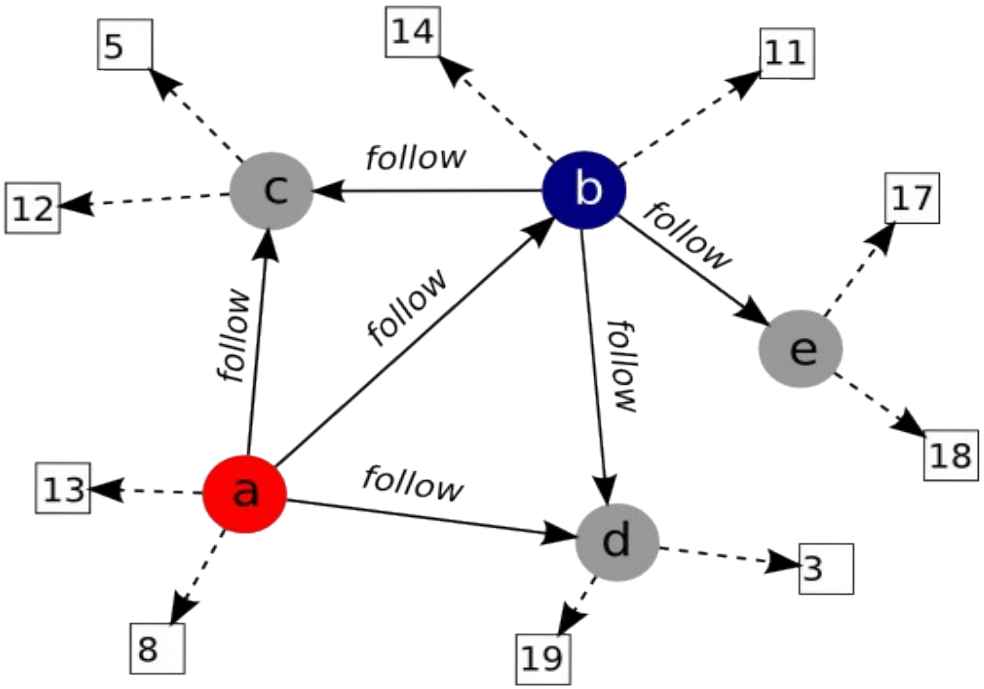
Graphity: An efficient Graph Model for Retrieving the Top-k News Feeds for users in social networks

René Pickhardt

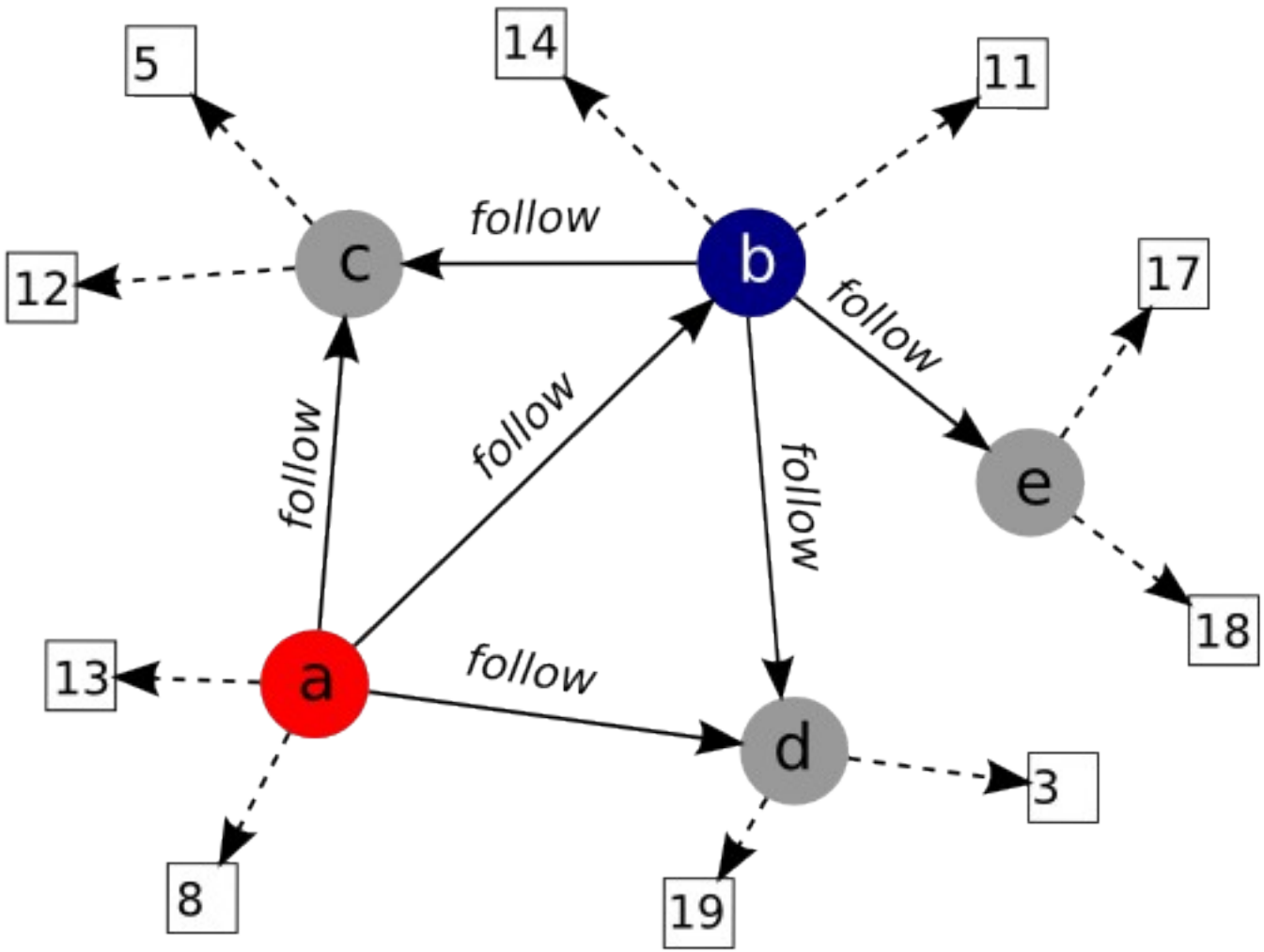


How to retrieve more than 10'000 temporal ordered news feeds per second in social networks with millions of users like Facebook and Twitter by using graph data bases (like neo4j) and Graphity

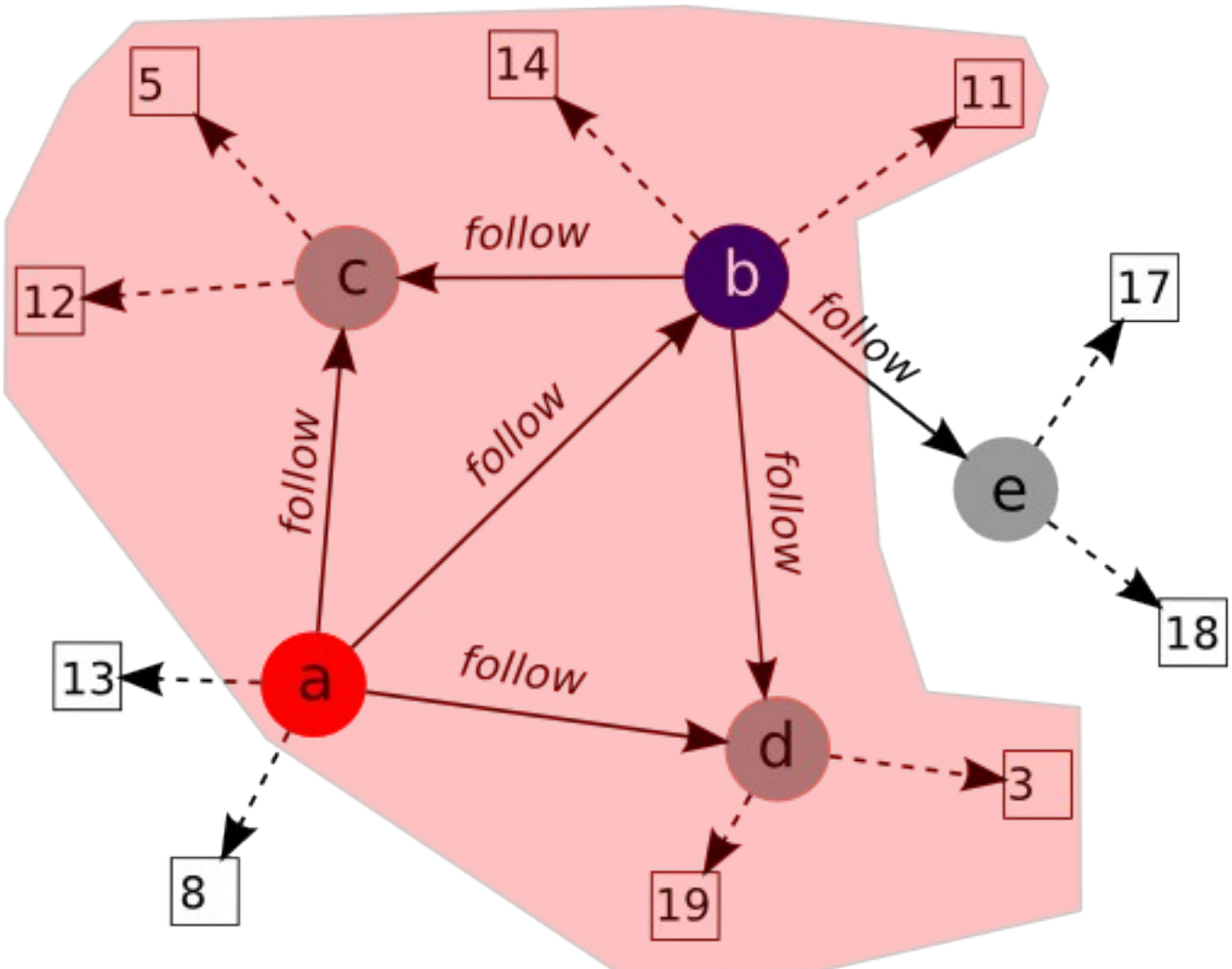
Add Title

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

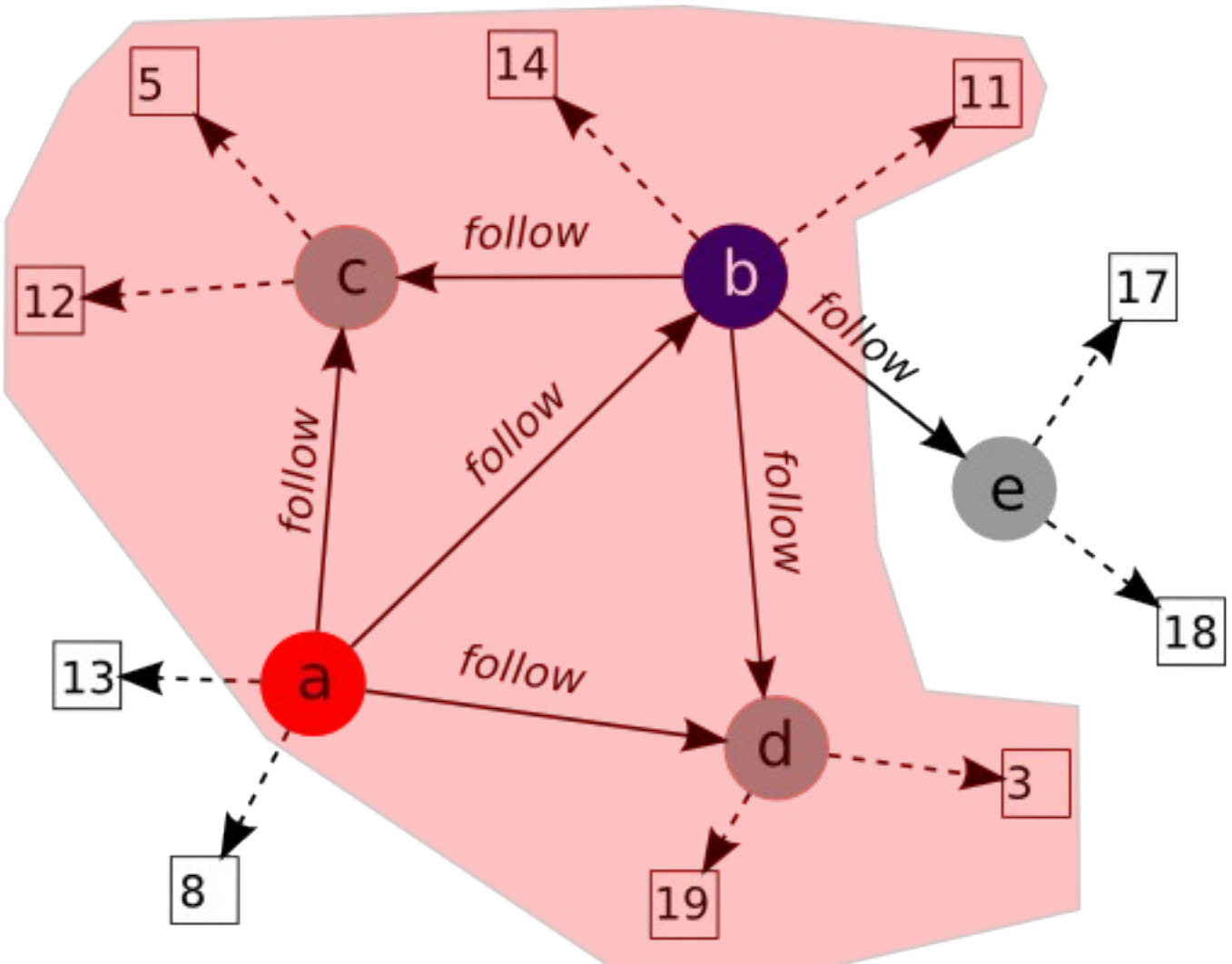
A "typical" social network graph

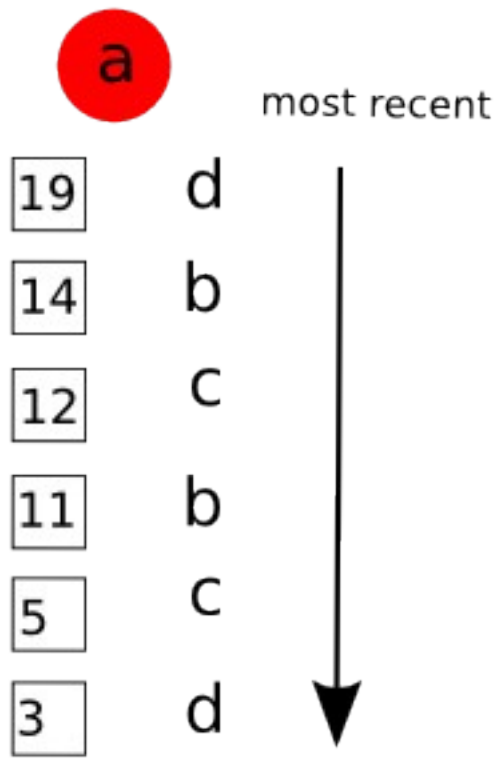


Retrieving Node A's news stream



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**

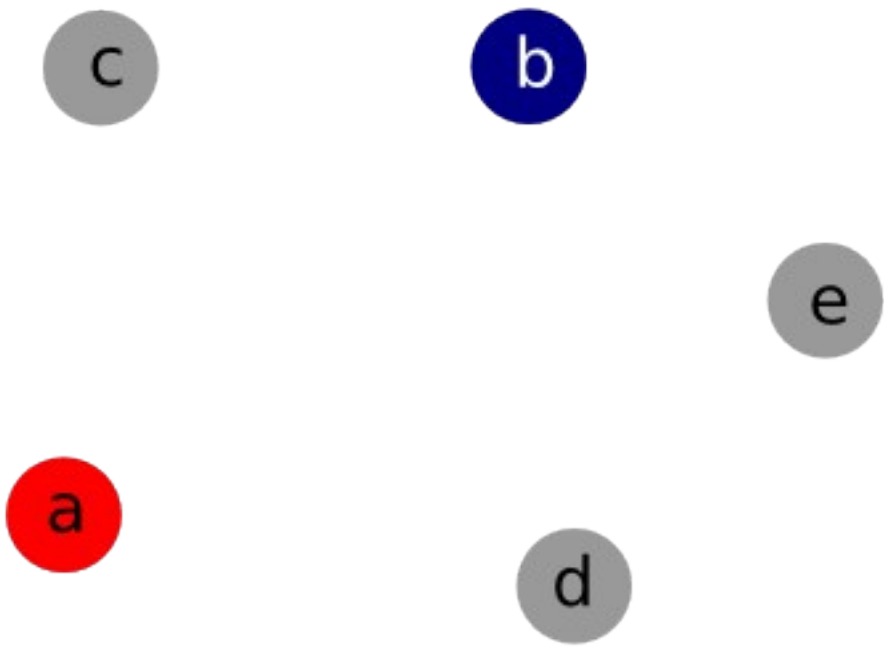
Add Title

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

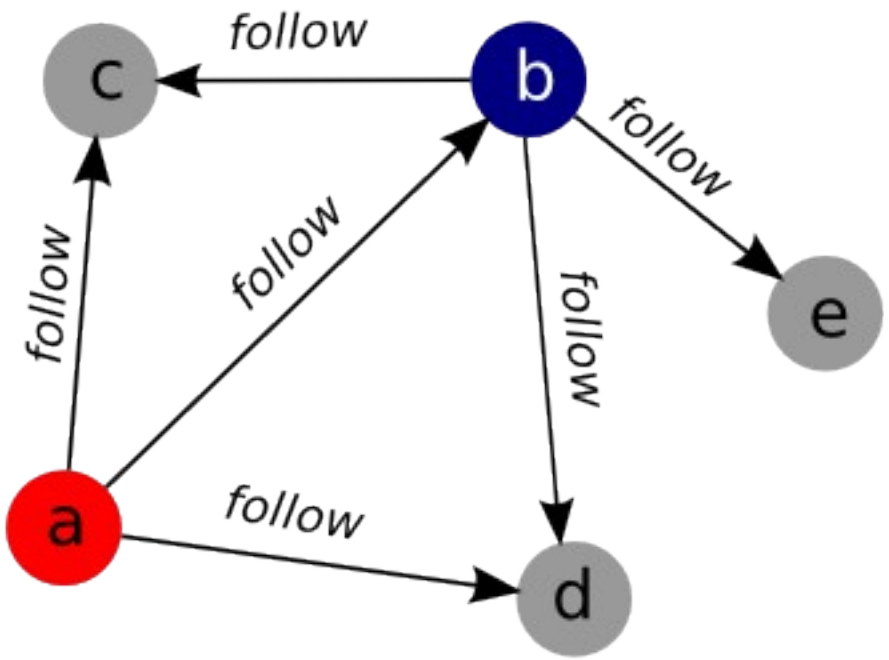
First we have some Users in a social Network

User

ID
a
b
c
d
e



They follow other users



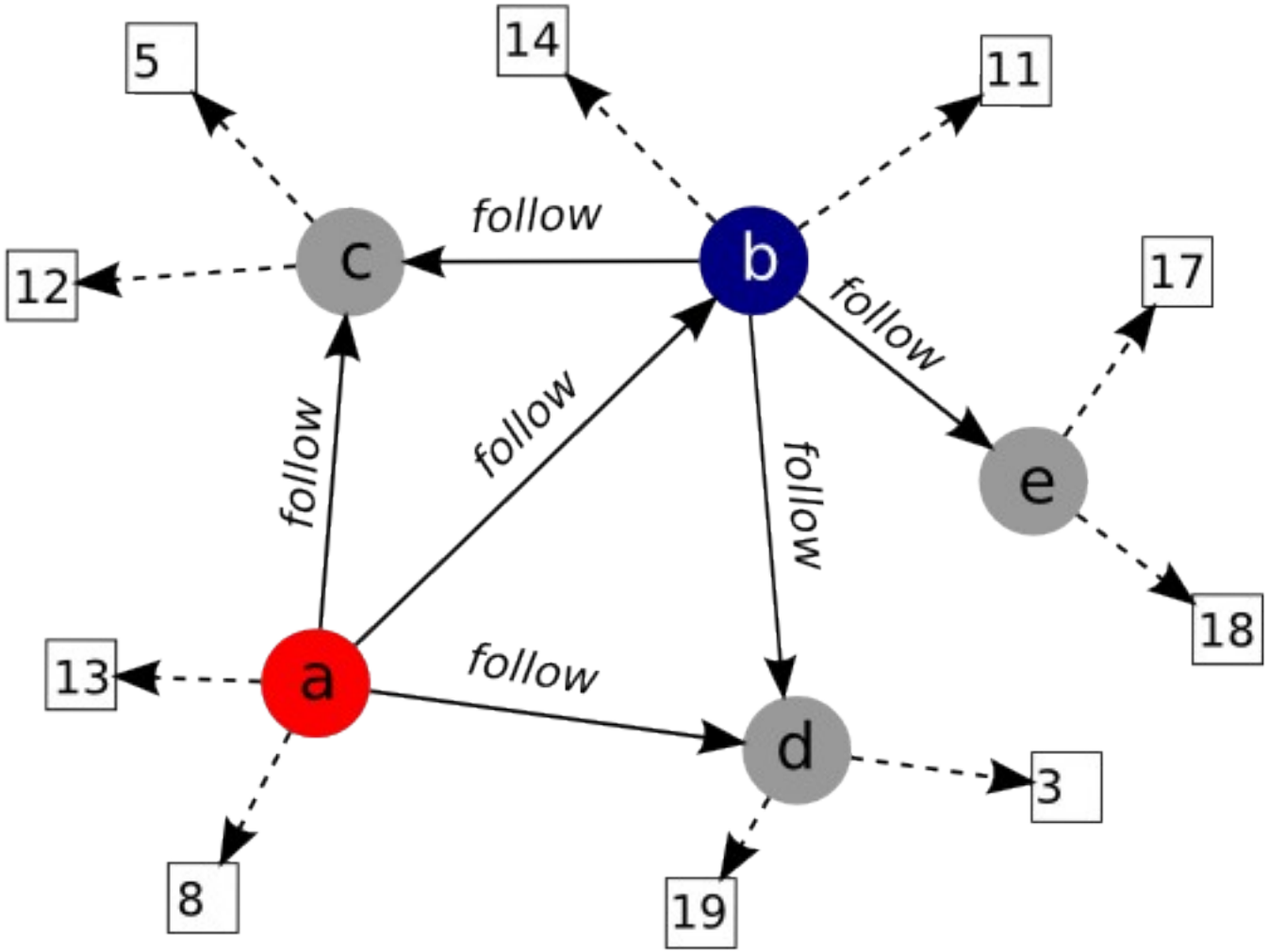
User

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

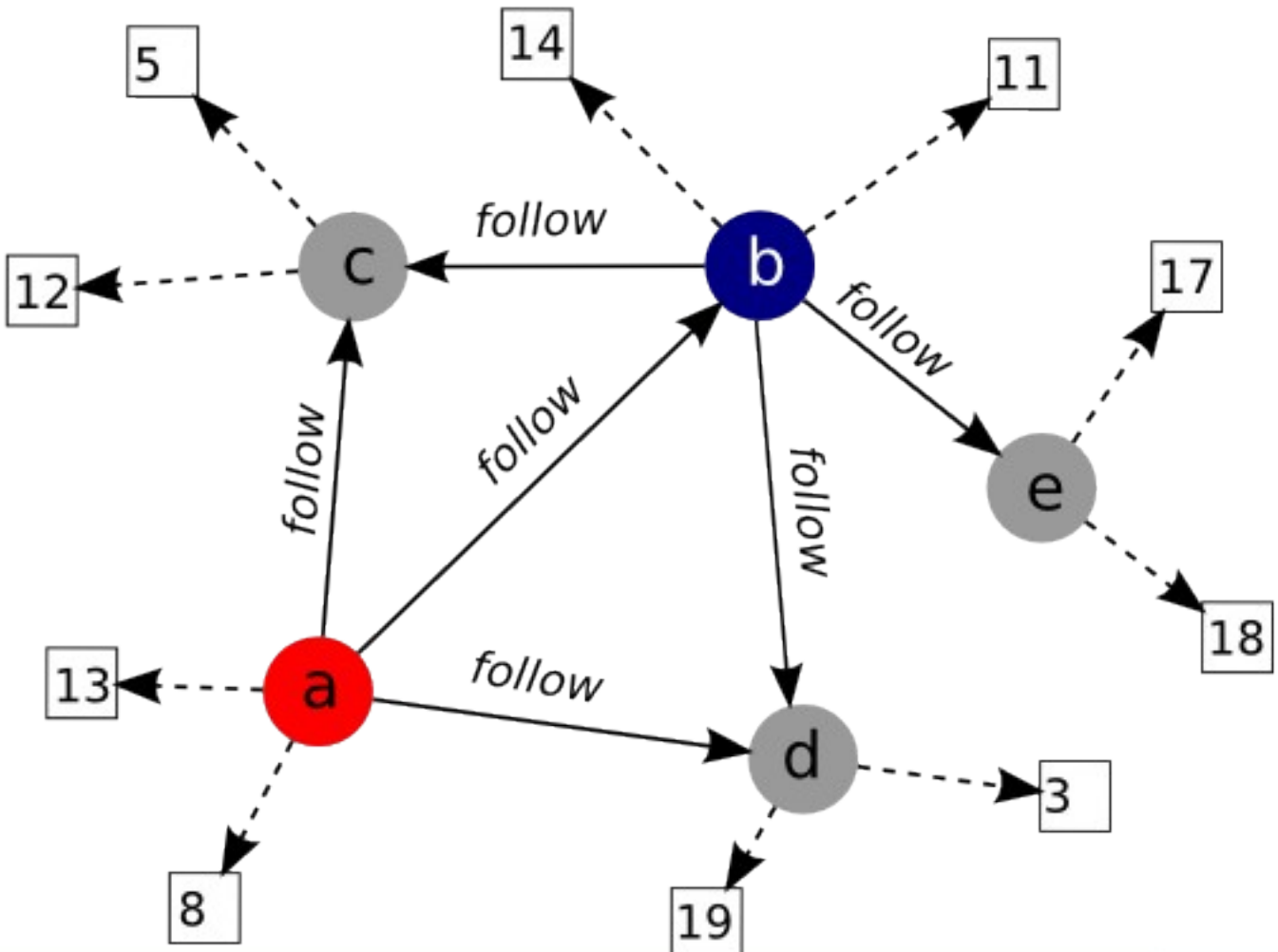
from	to
a	c
a	b
a	d
b	c
b	d
b	e

StatusUpdates

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 su.User, su.time, su.Content
  FROM StatusUpdates su
    JOIN Follower f on su.User=f.to
    JOIN User u on u.ID = f.from
  WHERE u.ID like "a" ORDER BY su.time DESC
```



User

ID
a
b
c
d
e

Follower

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

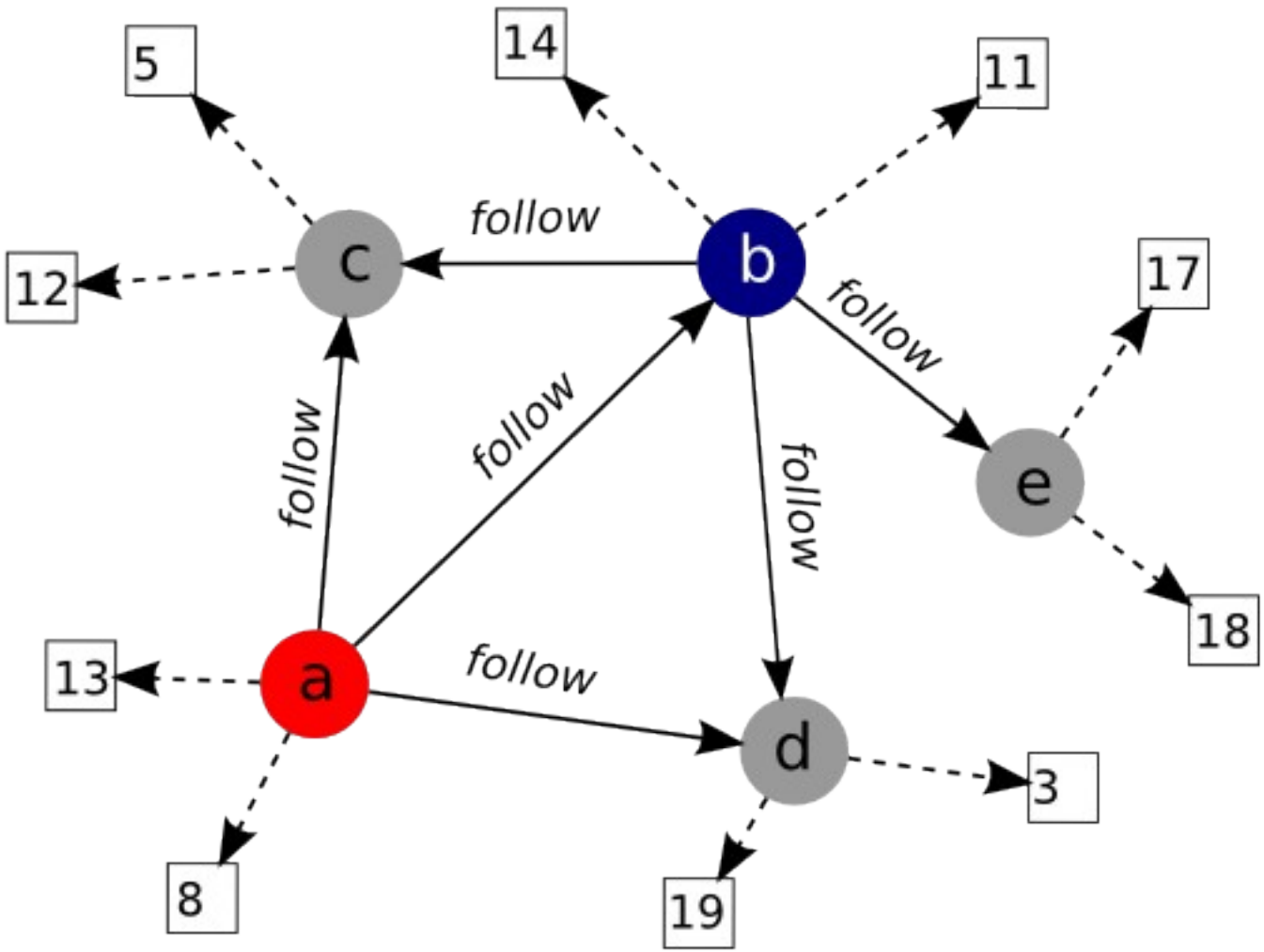
StatusUpdates

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

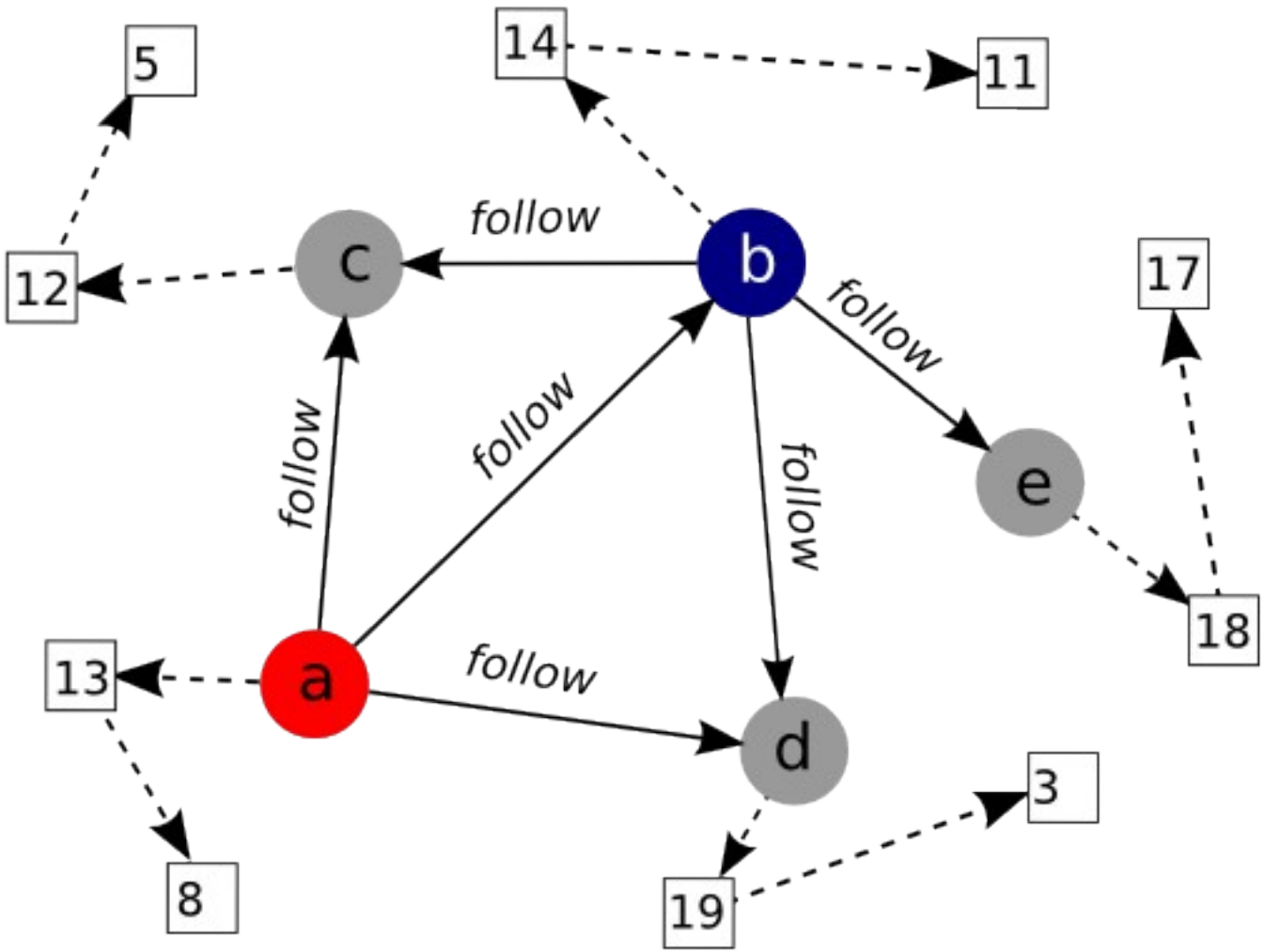
Add Title

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

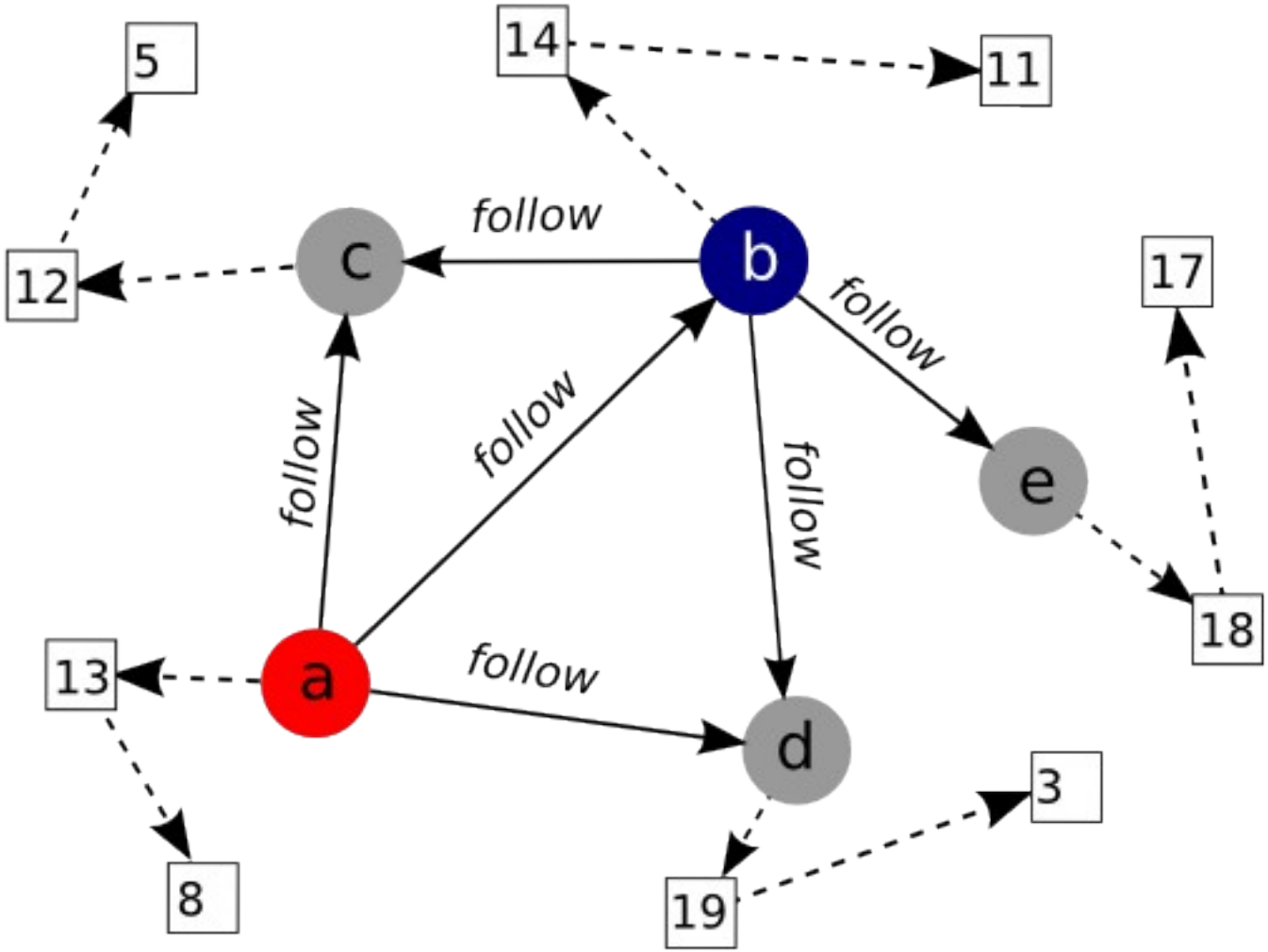
Introducing linked lists for status updates! (1)



Introducing linked lists for status updates! (2)

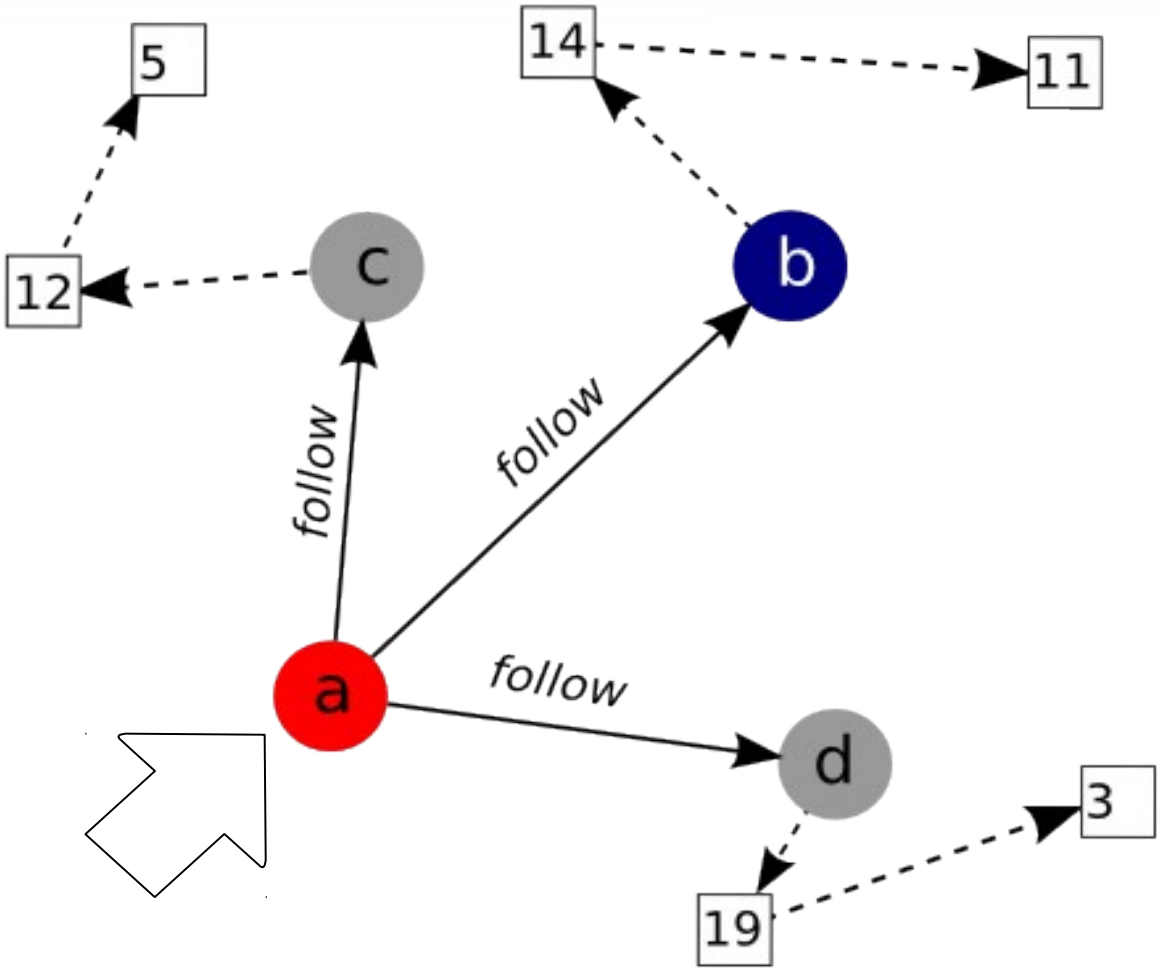


Pros of this approach



- dynamic retrival possible
- very flexible data structure
- inserts are very fast

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

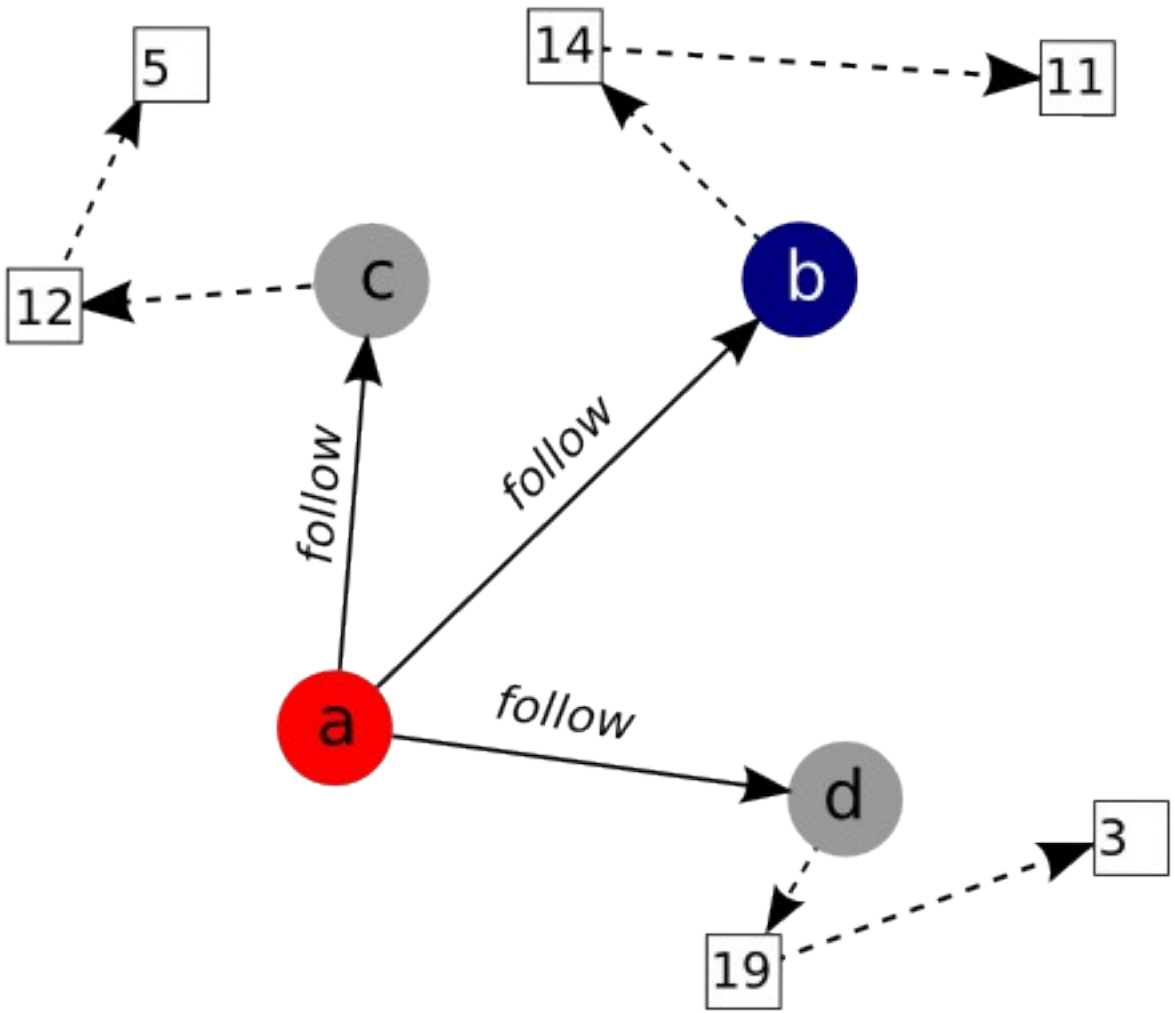


- entire ego network must be sorted
- Size *d of an* ego network is usually much bigger than the number of retrieved items *k*.
- no sorting ==> unclear which path to traverse first!

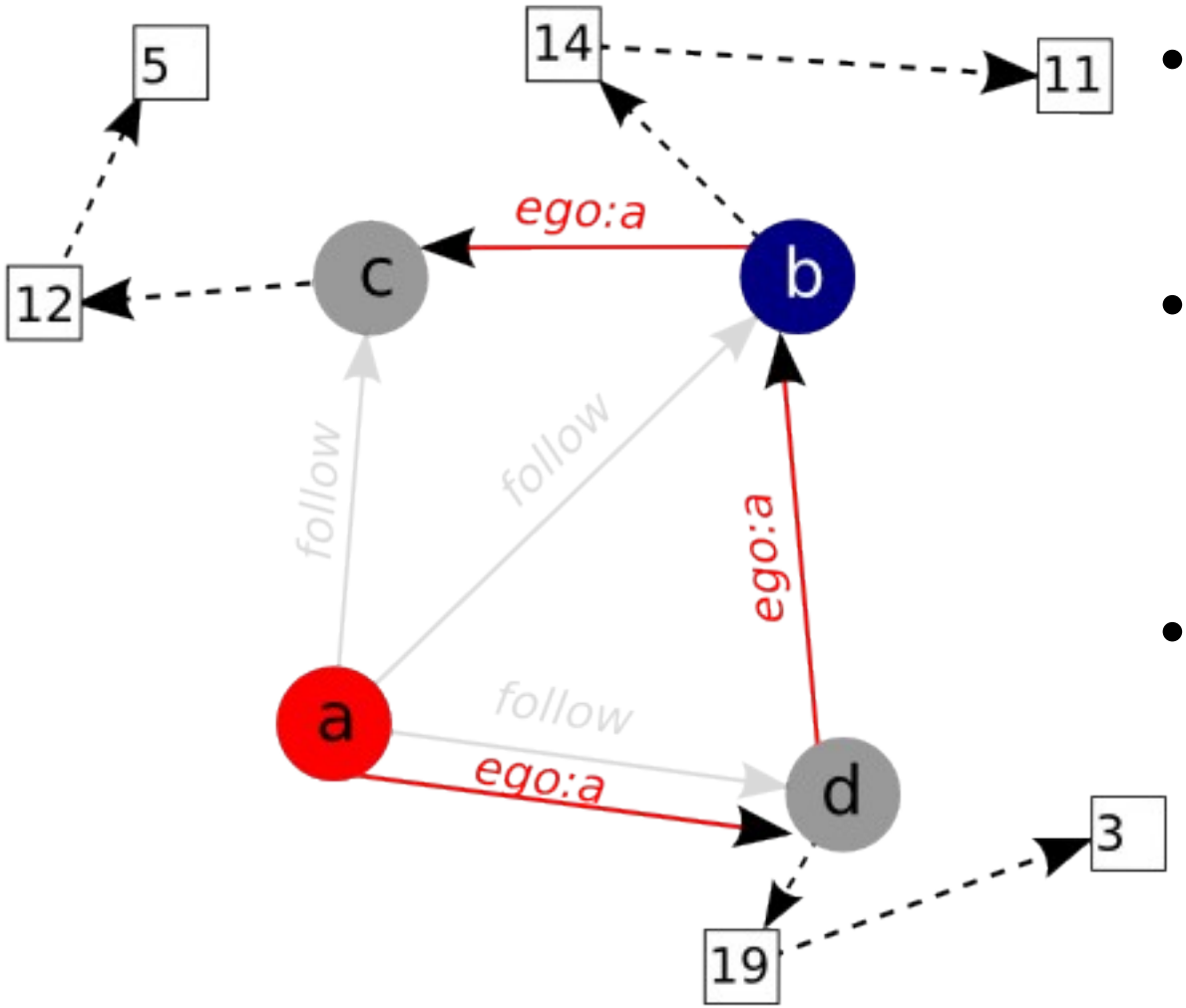
Add Title

- Introduction to the newsfeed problem
- Why relational Data bases won't do it
- An obvious graph data base approach
- The construction and idea of grahity
- 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



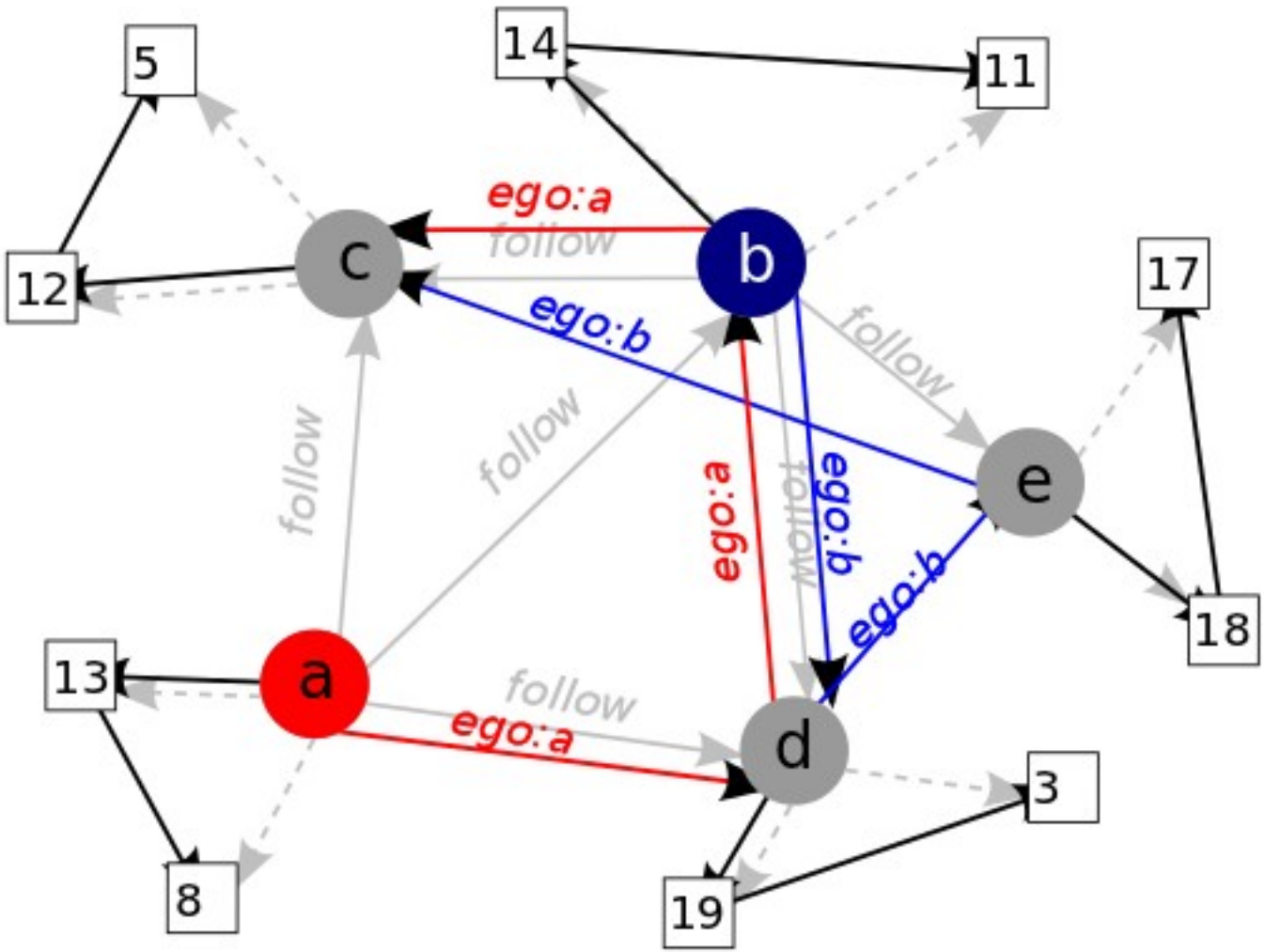
graphity index for the node "a"



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 status update

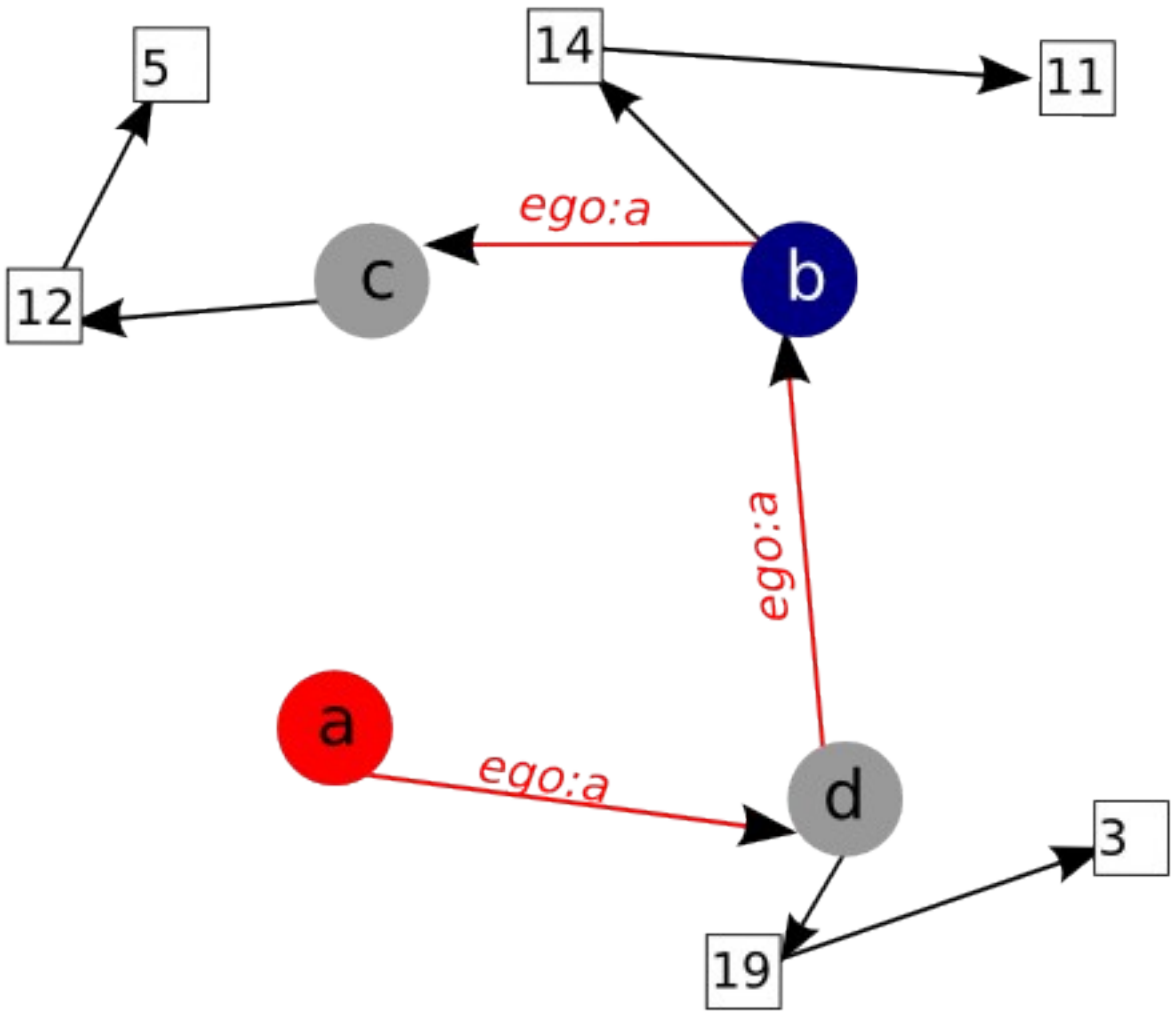
second graphity index for node "b"



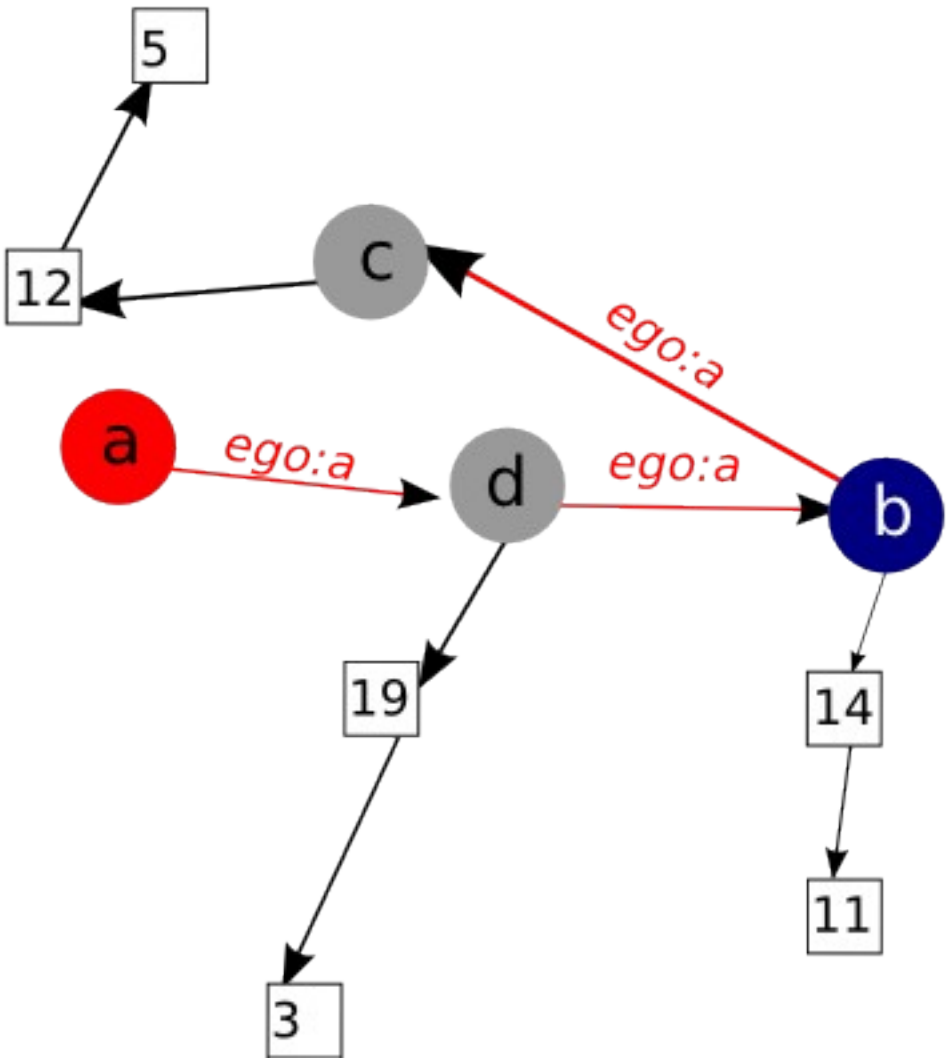
Add Title

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

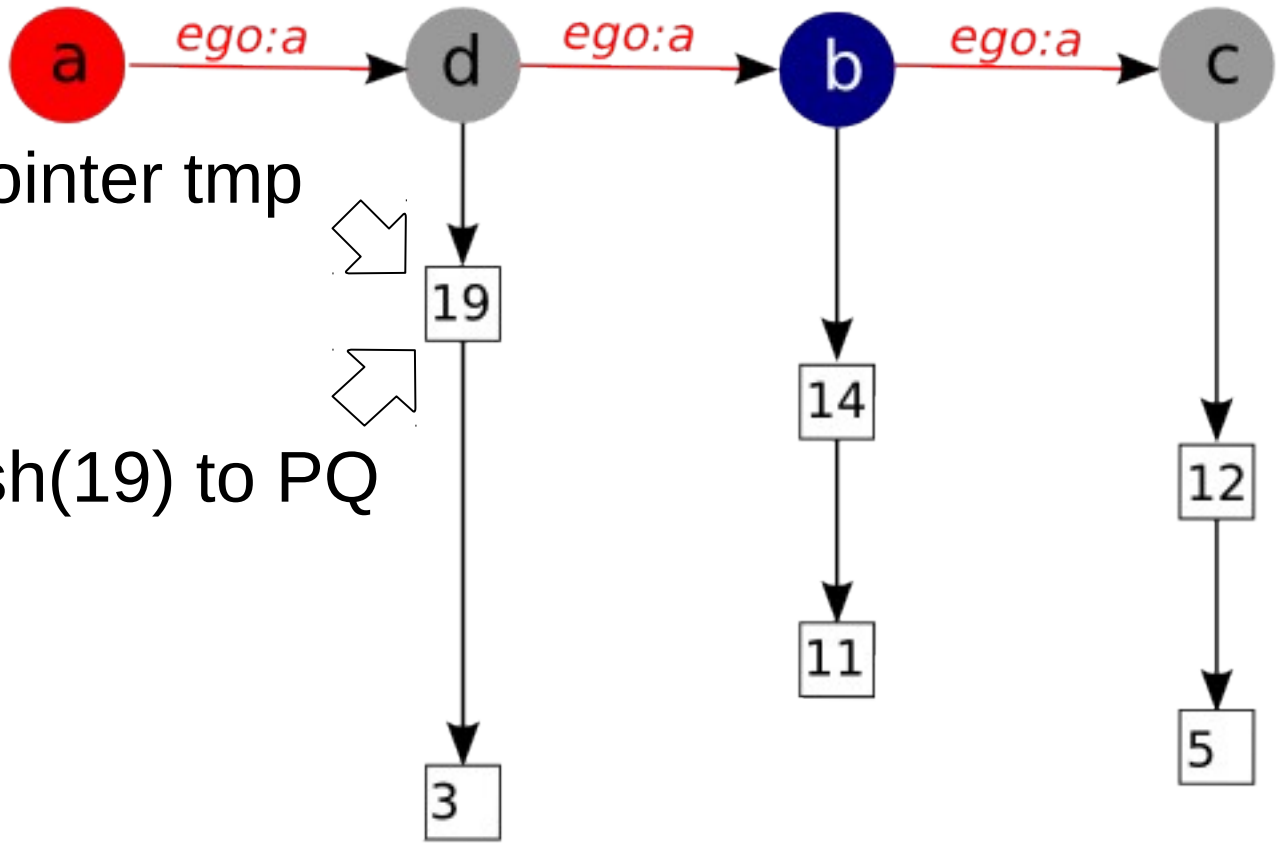
Rearanging the graph ...



Rearanging the graph...



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



Pointer tmp

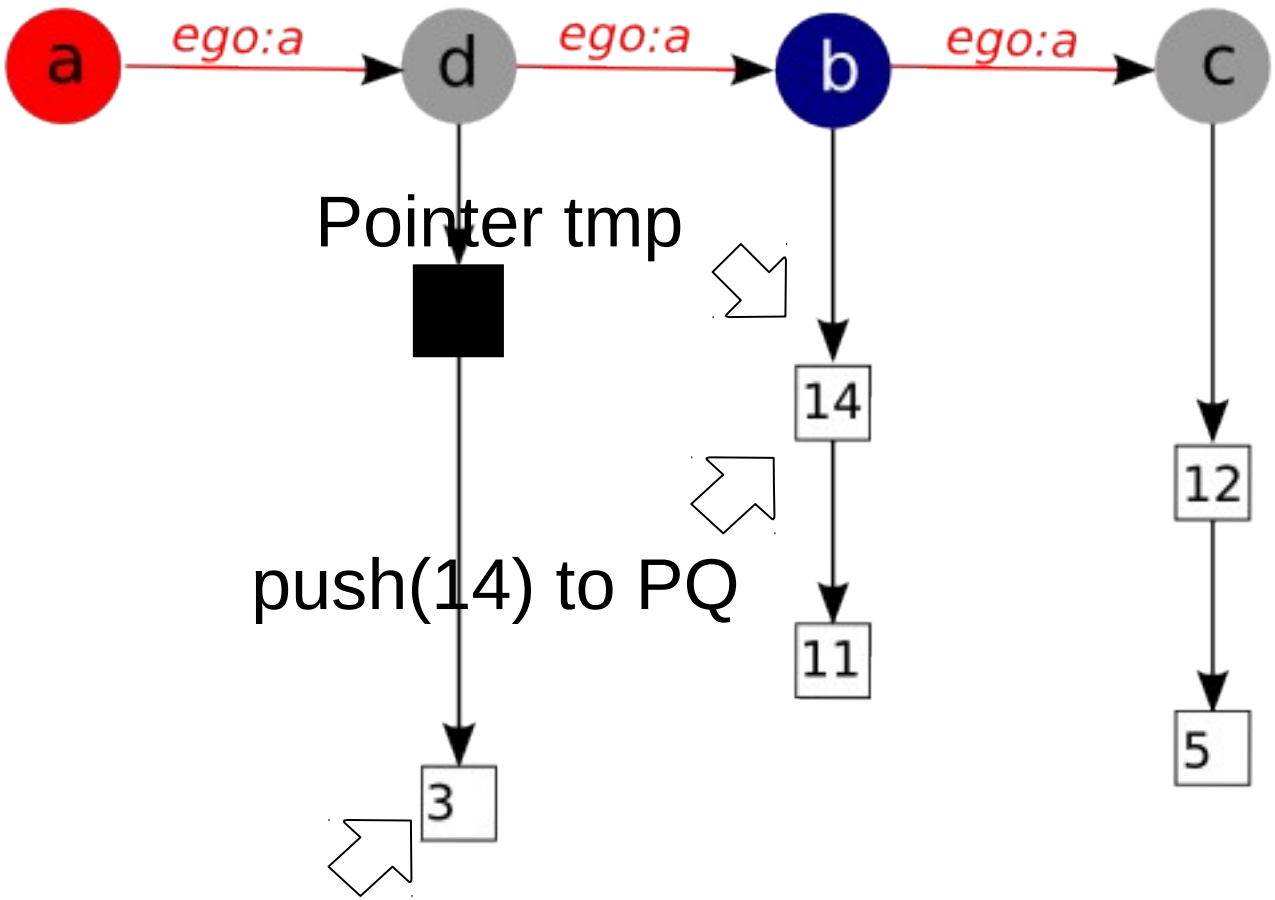
push(19) to PQ

Priority Queue:

19

Stream:

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



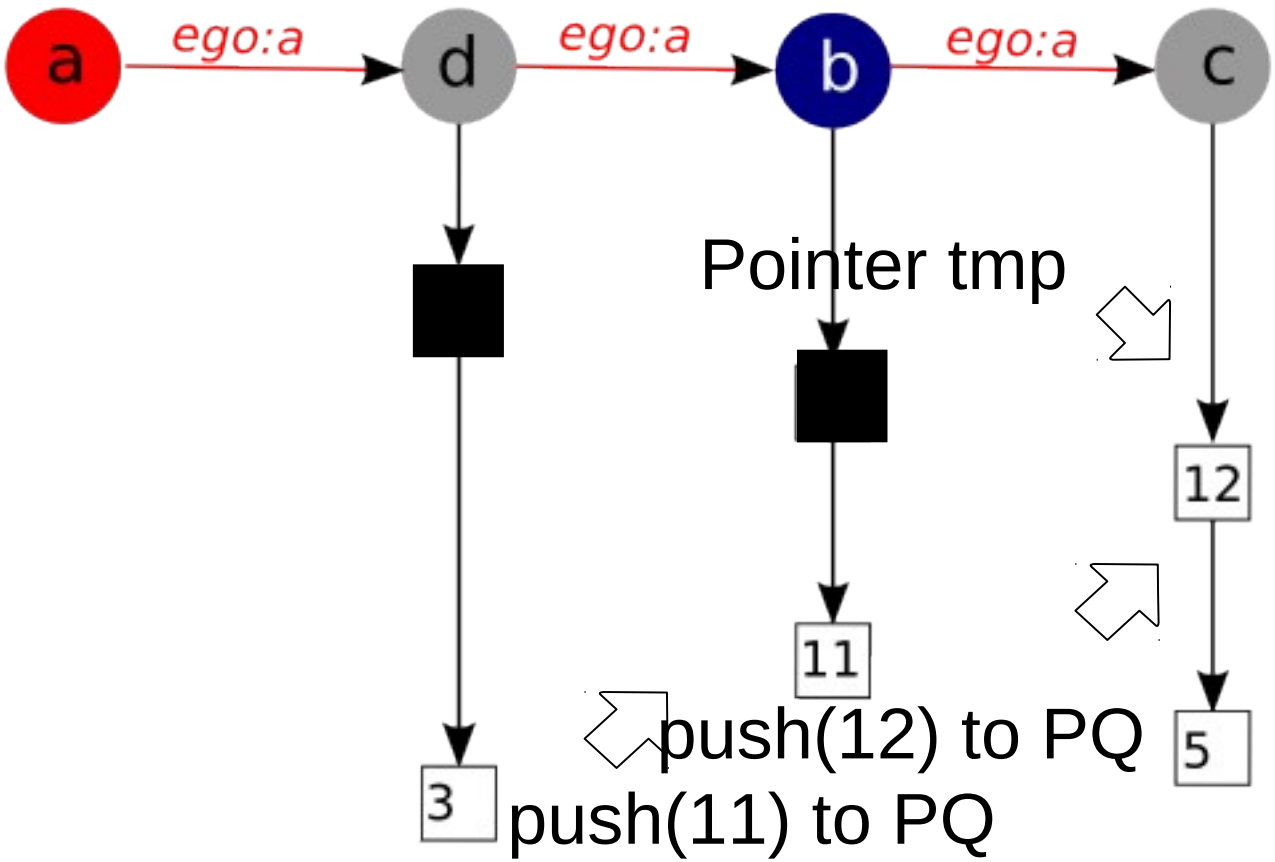
Priority Queue:

14
3

push(3) to PQ

Stream: (19,d)

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

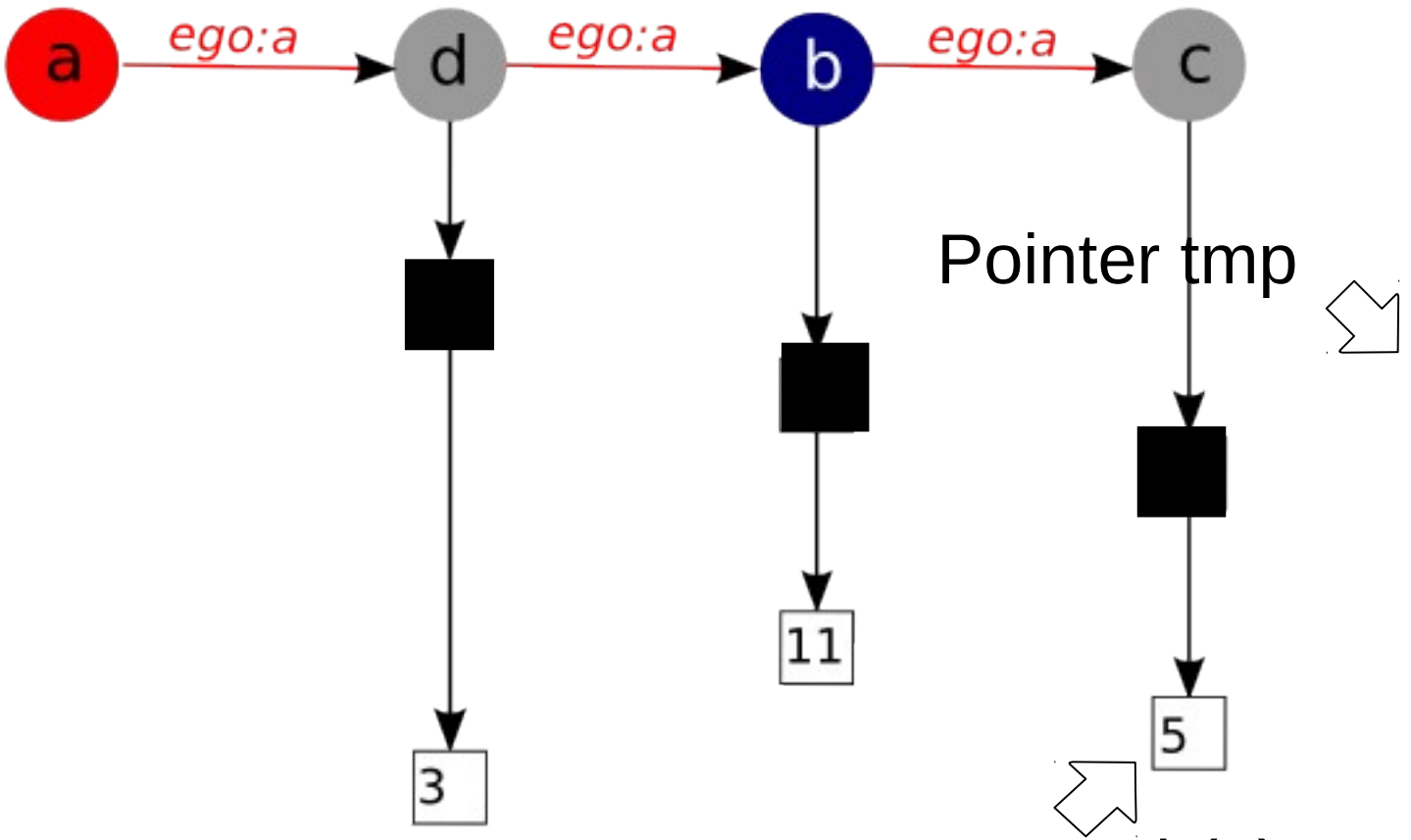


Priority Queue:

12
11
3

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

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

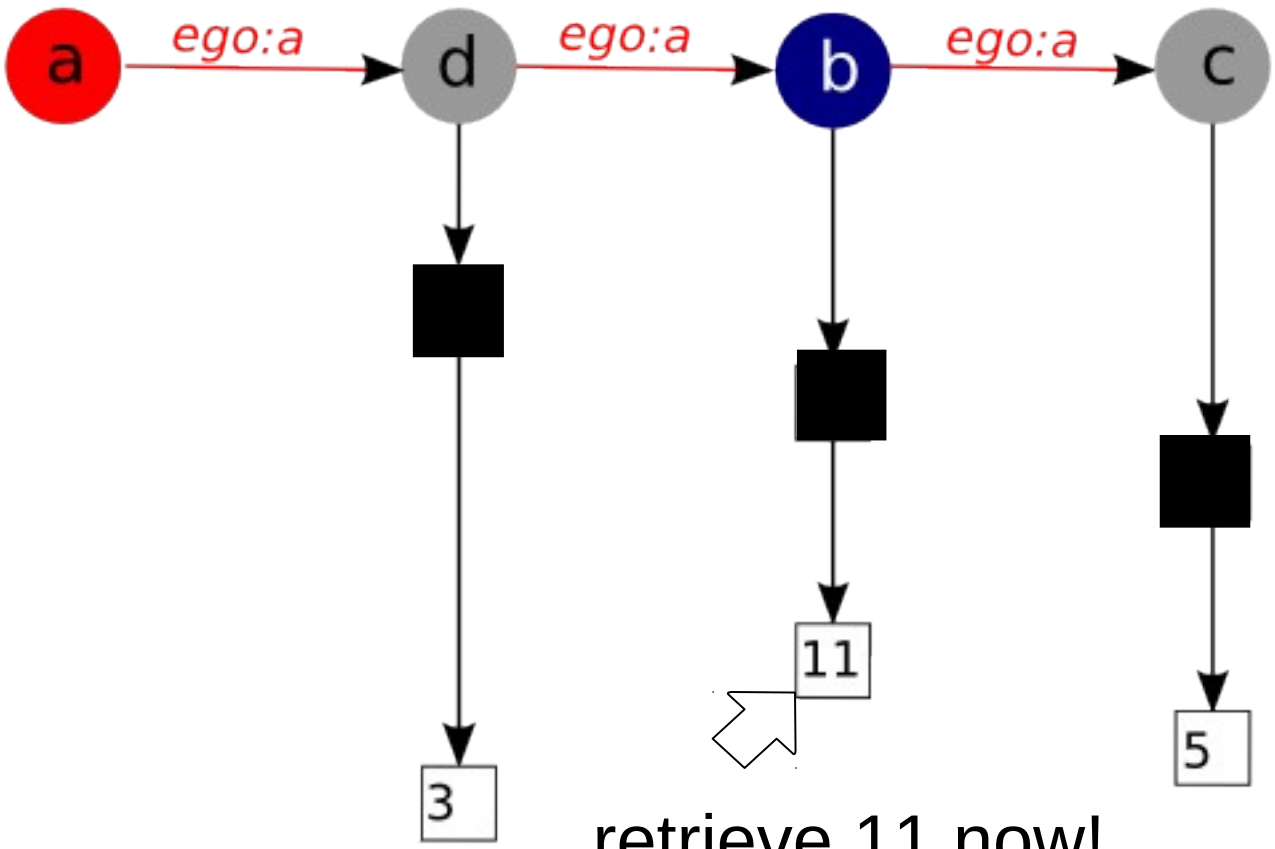


Priority Queue:

11
5
3

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

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



Priority Queue:

11
5
3

retrieve 11 now!

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

Add Title

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

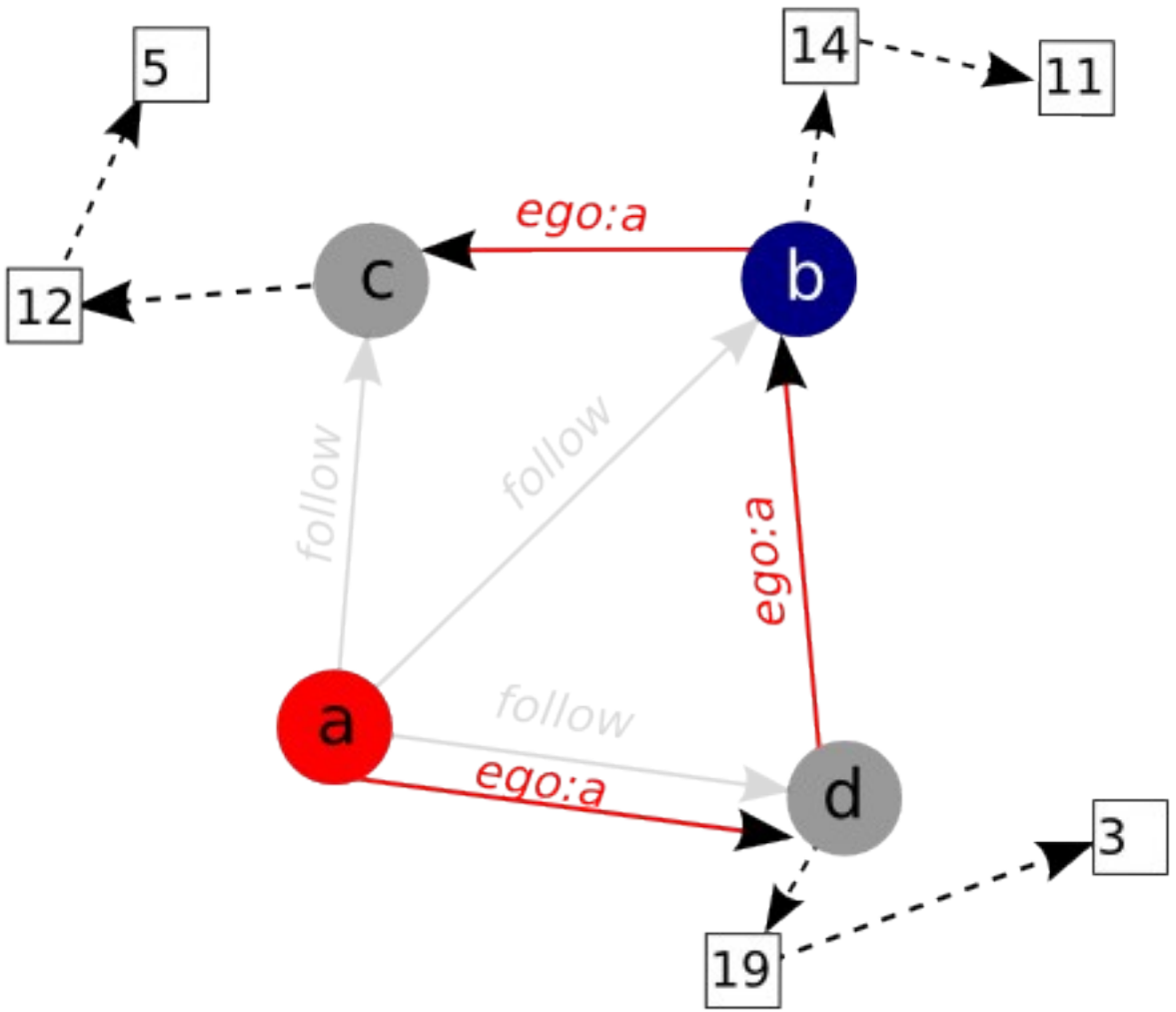
Updates need to be done in the following situation

Updates need to be done in the following situations

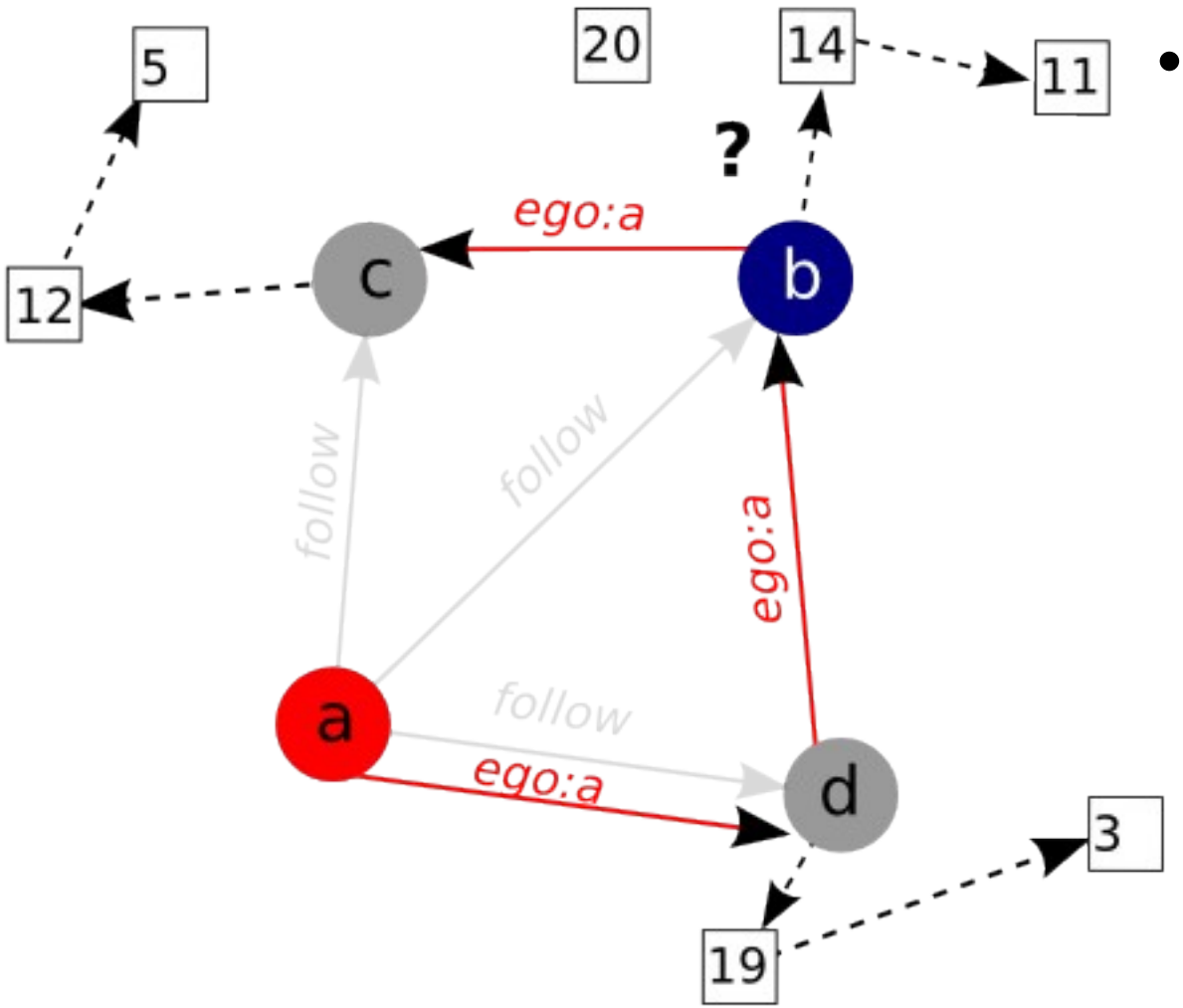
- **new created content item** (O(d))
  - index of every follower needs to be updated
- **new created follow relation** (O(d))
  - index of follower needs to be updated
- **friendship relation breaks** (O(d))
  - index of the former follower needs to be updated
- **most recent content item of a user is deleted** (O(d²))
  - index of every follower nees to be updated

Creating new Content items

b creates a new node



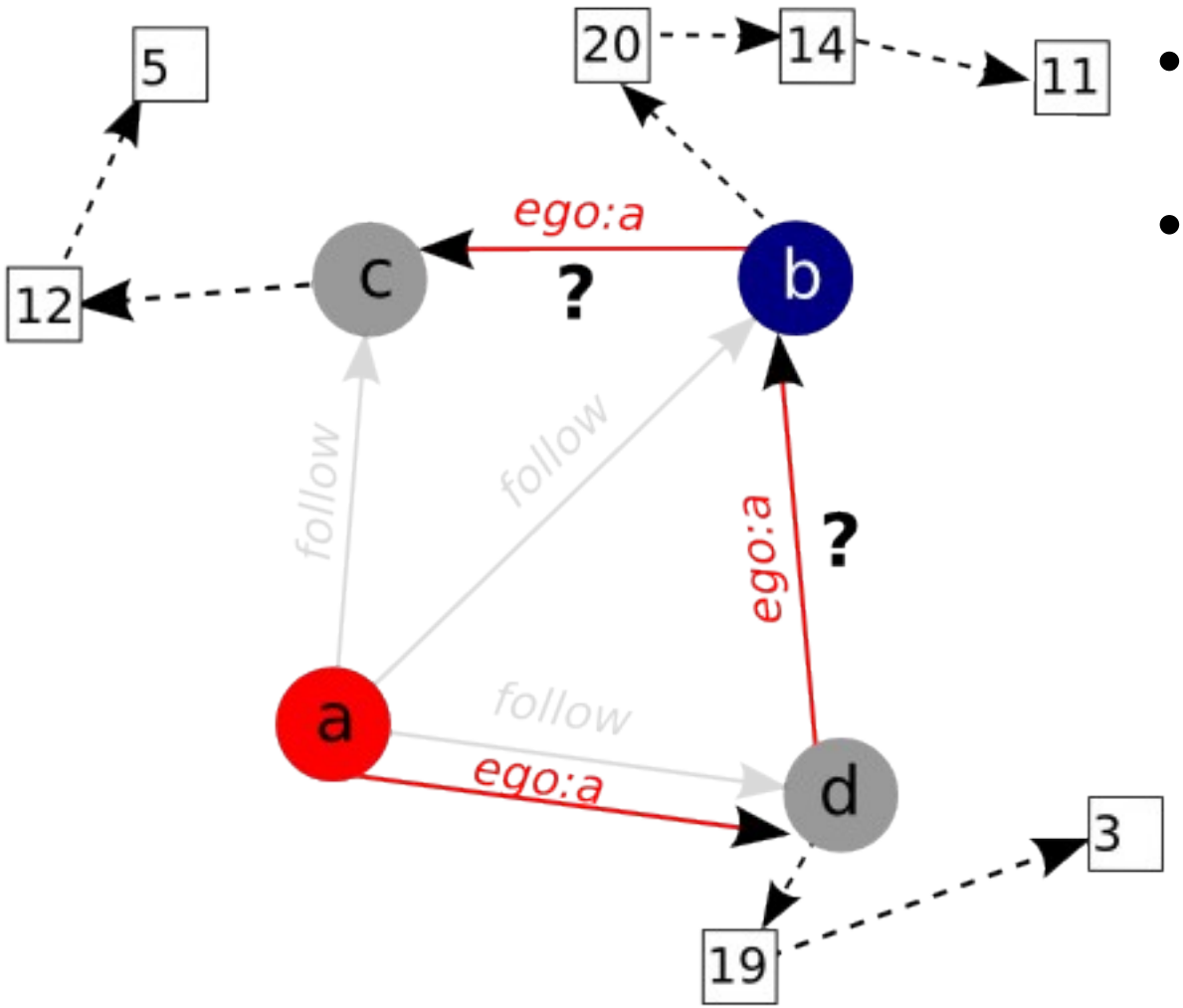
Creating new Content items



b created 20

- update linked list of b's content items

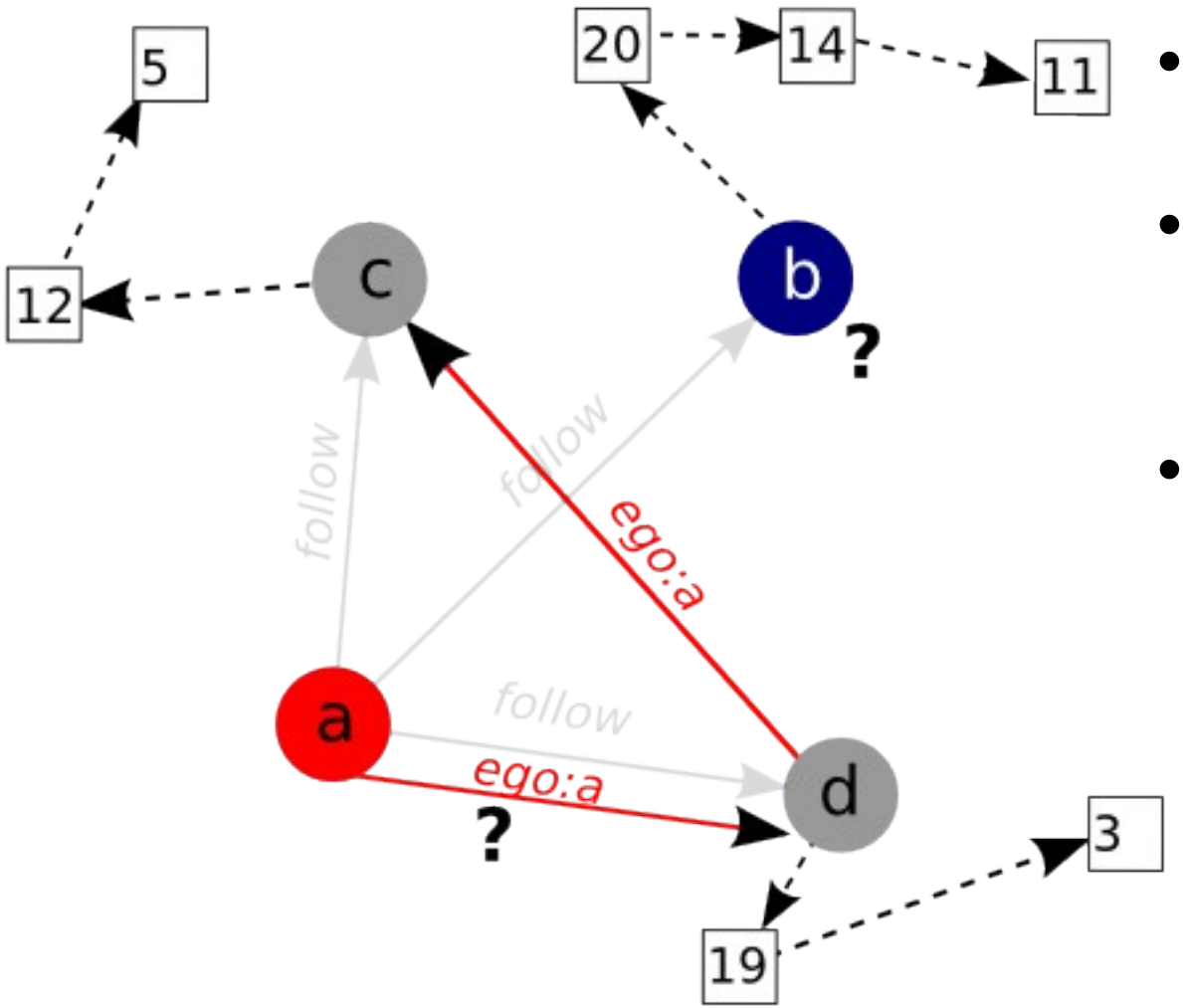
Creating new Content items



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)

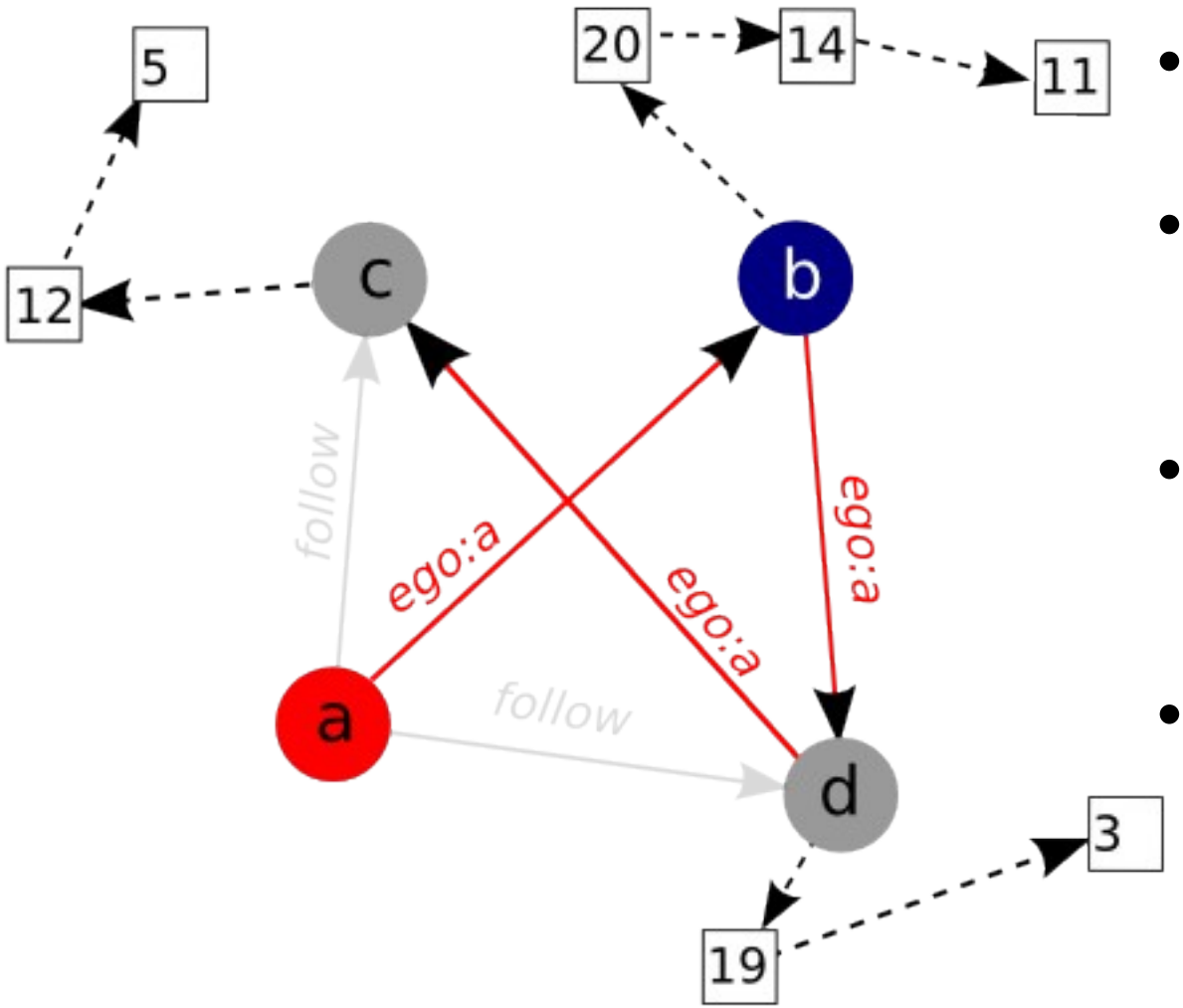
Creating new Content items



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

Creating new Content items



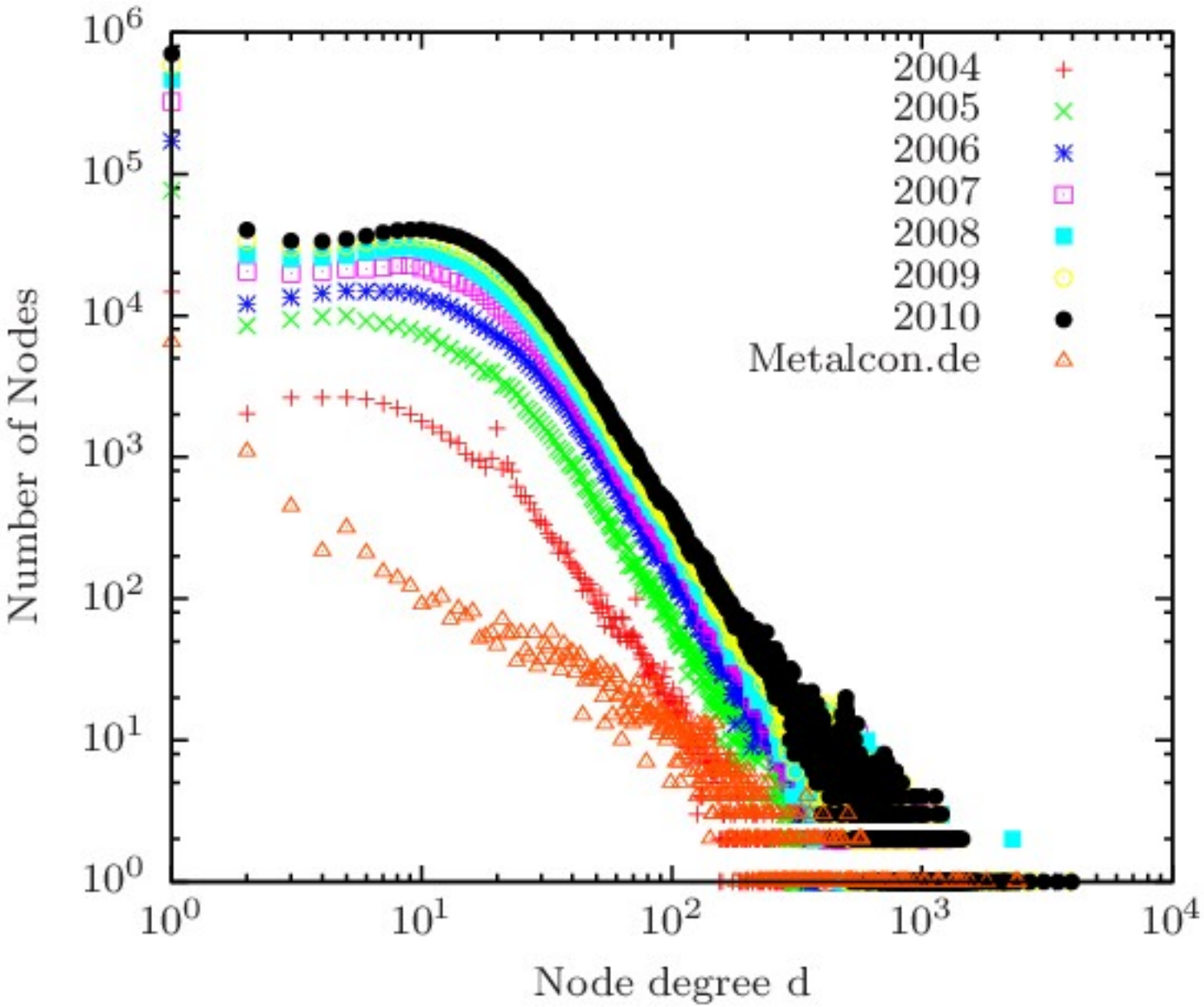
b created 20

- 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

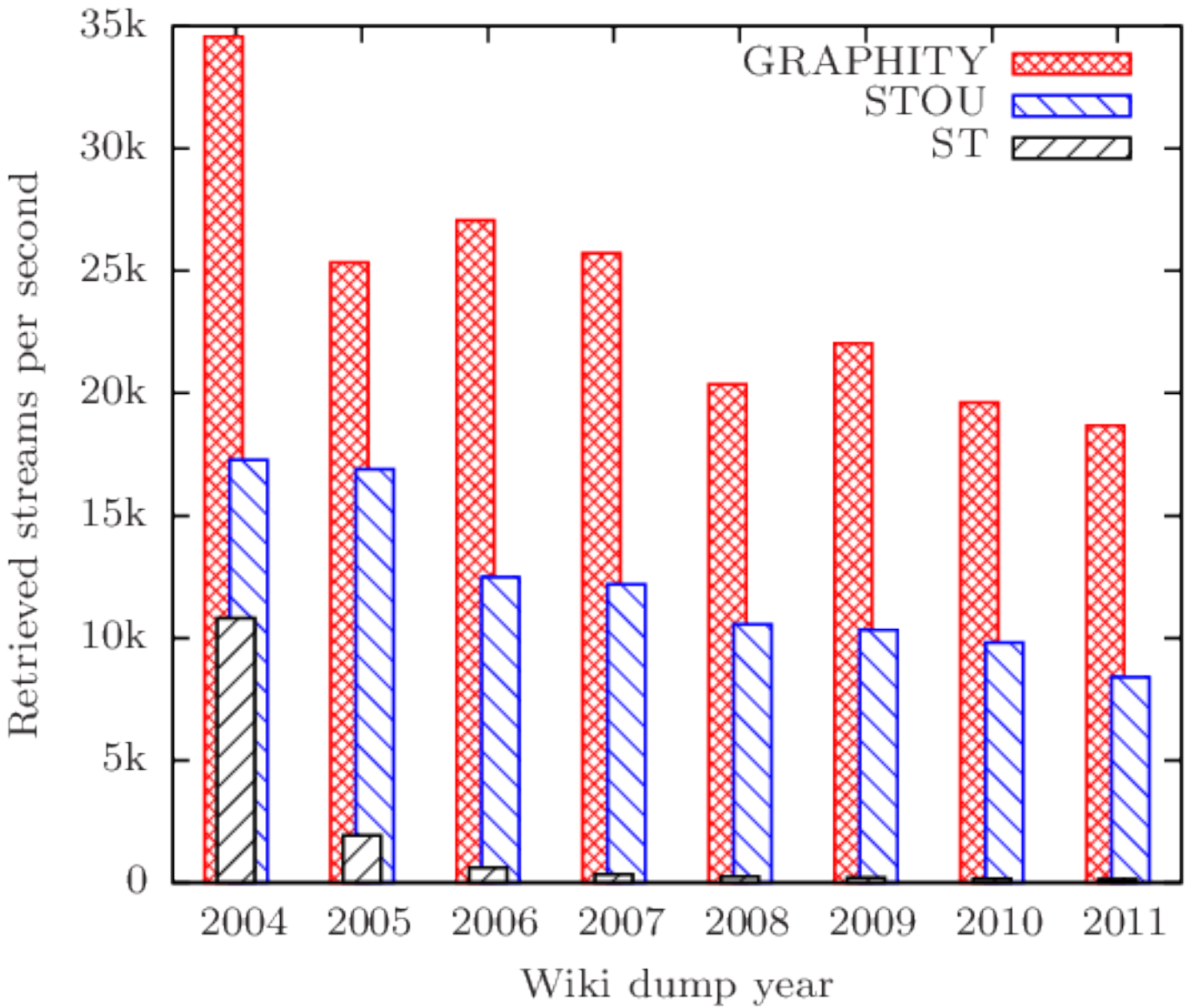
Add Title

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

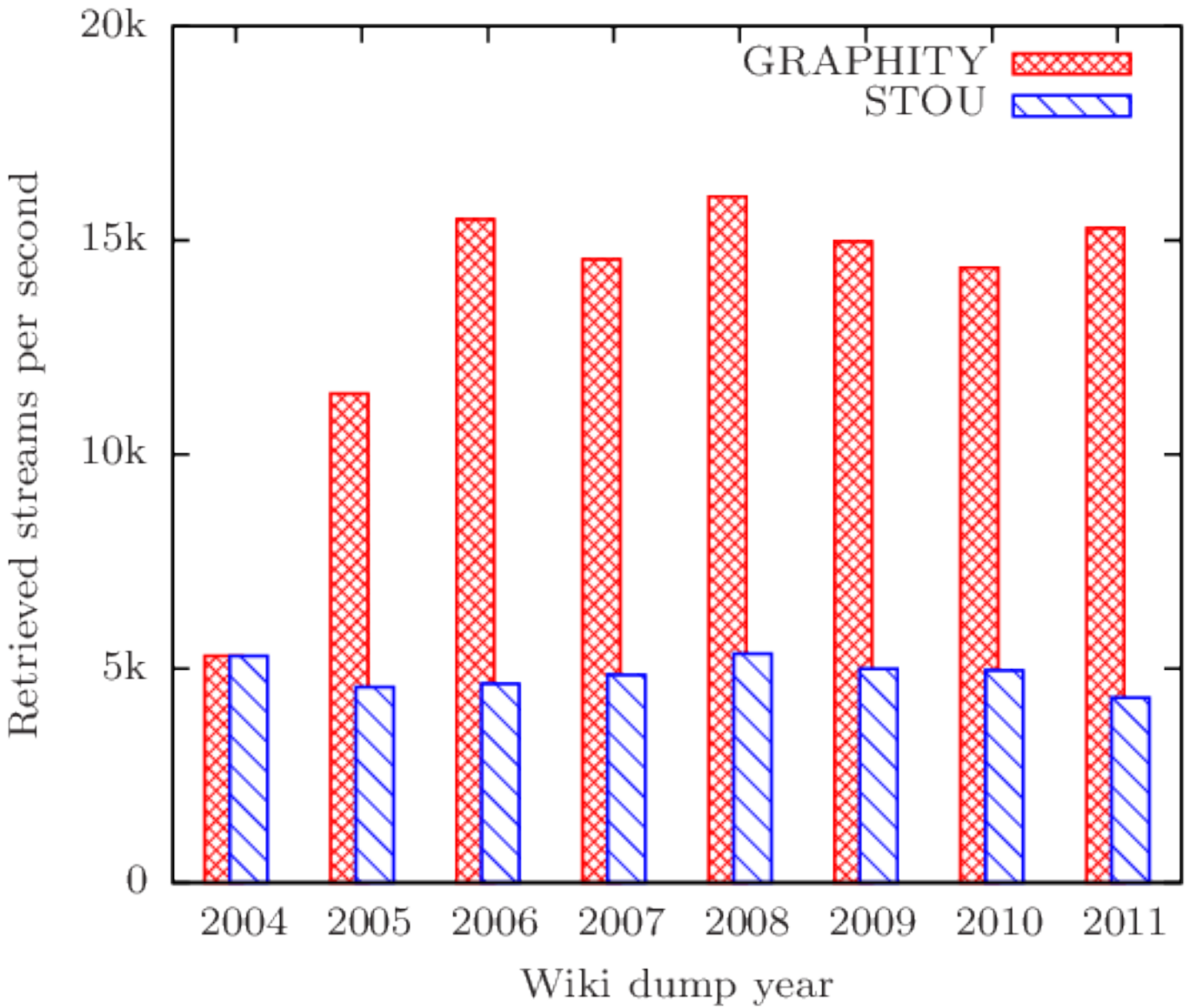
Evaluation Characteristics of data sets



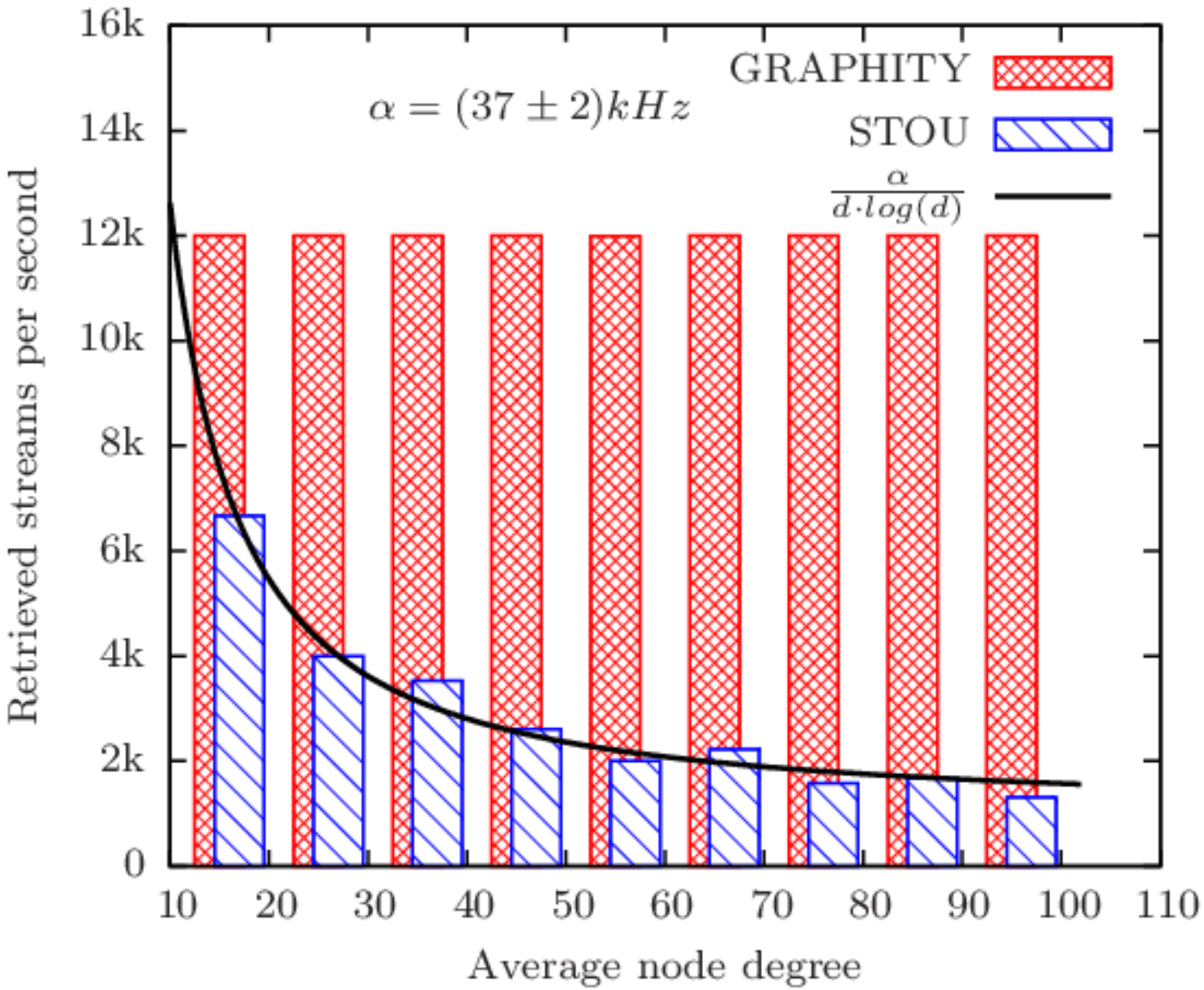
Retrieving streams on all data sets



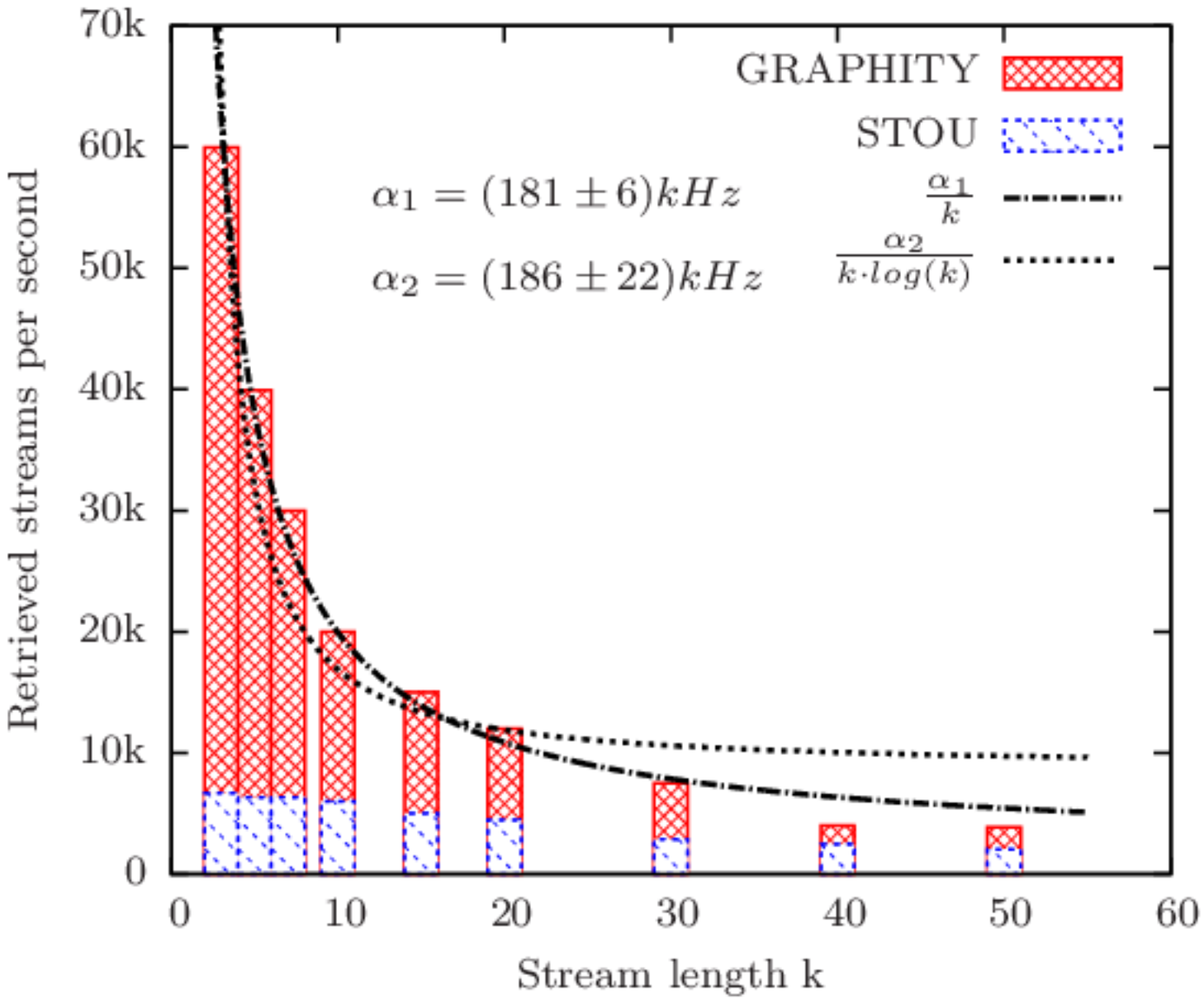
Retrieving streams Node degre > 10



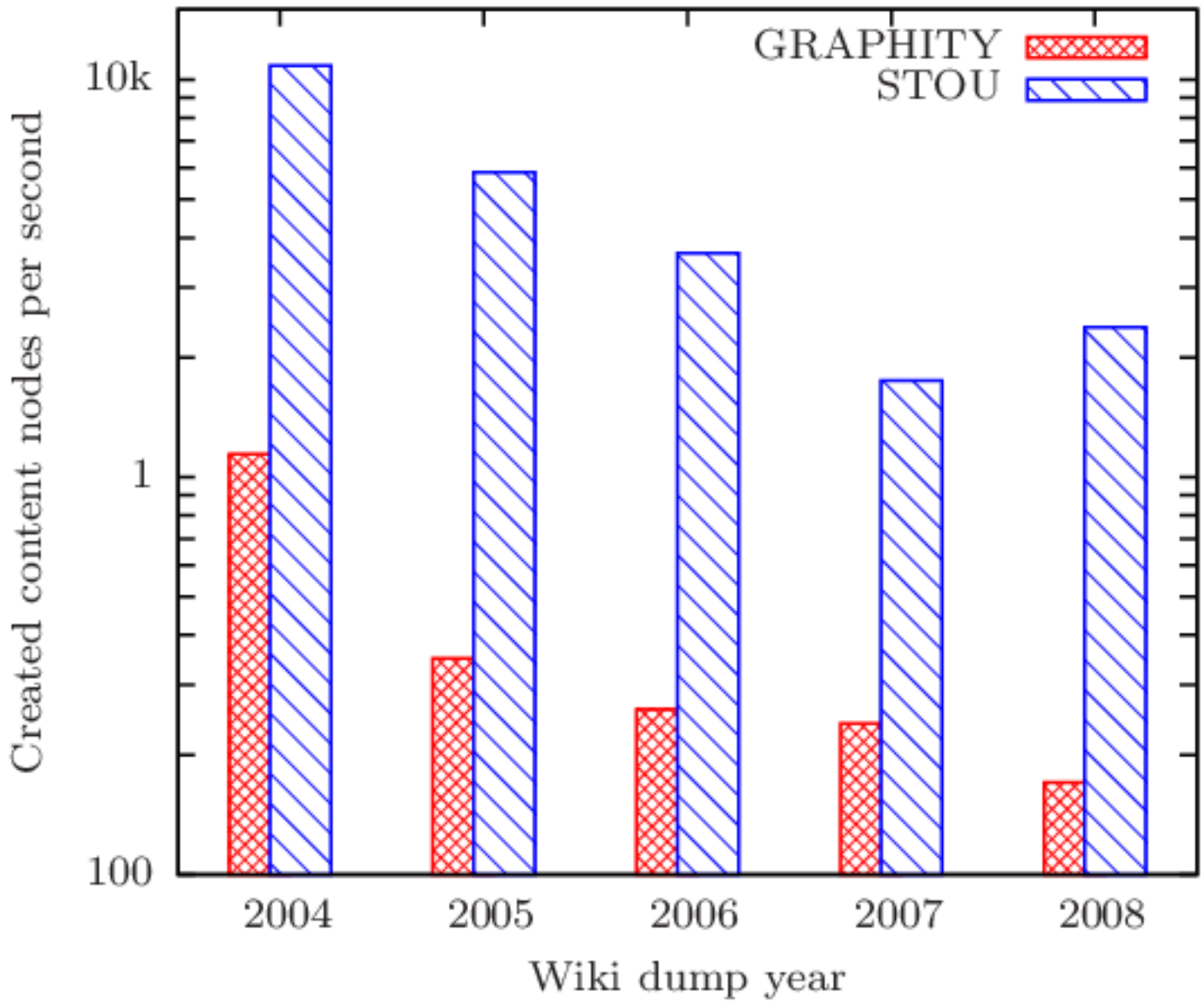
demonstrating independence of node degree



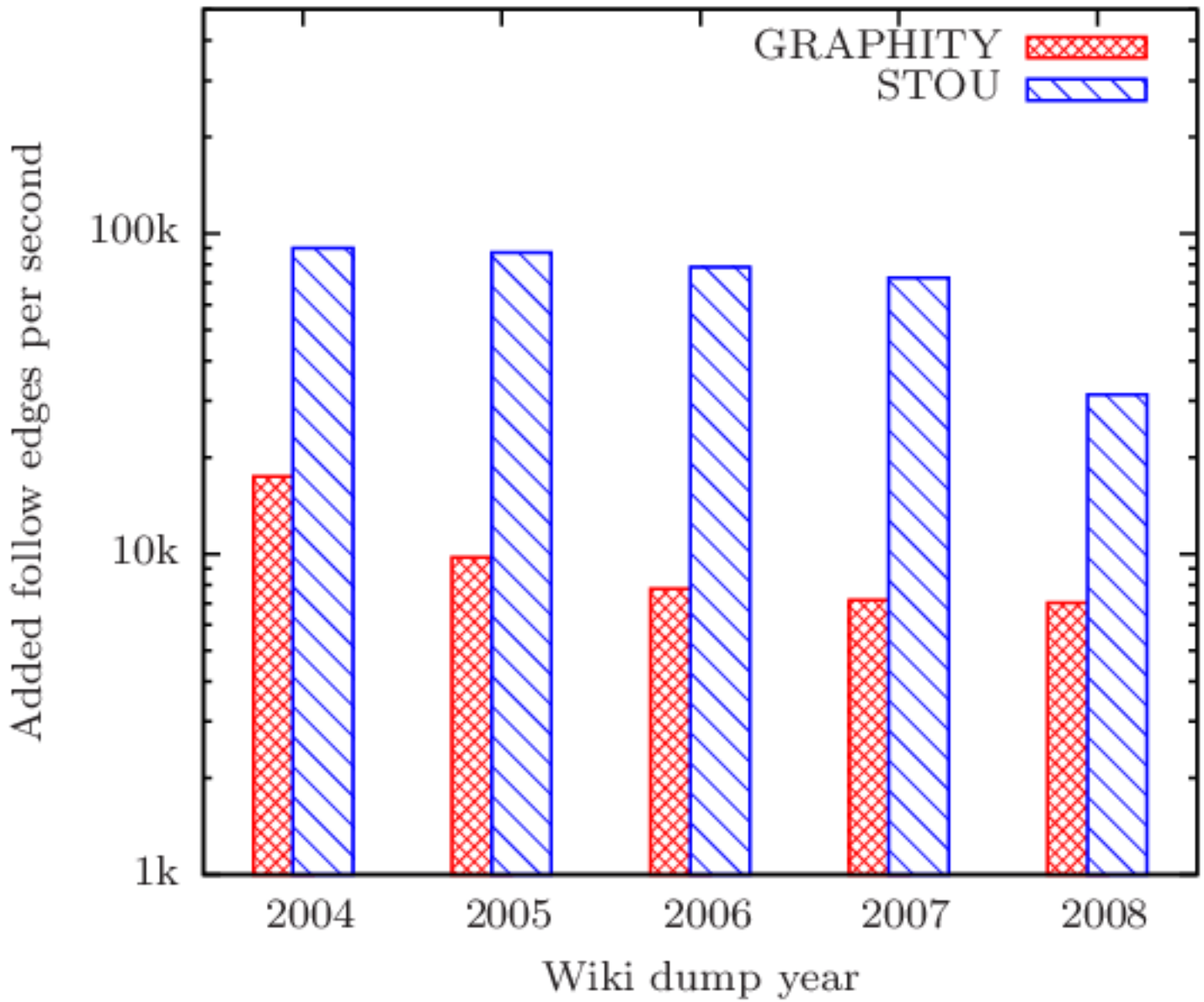
demonstrating linear dependence of k



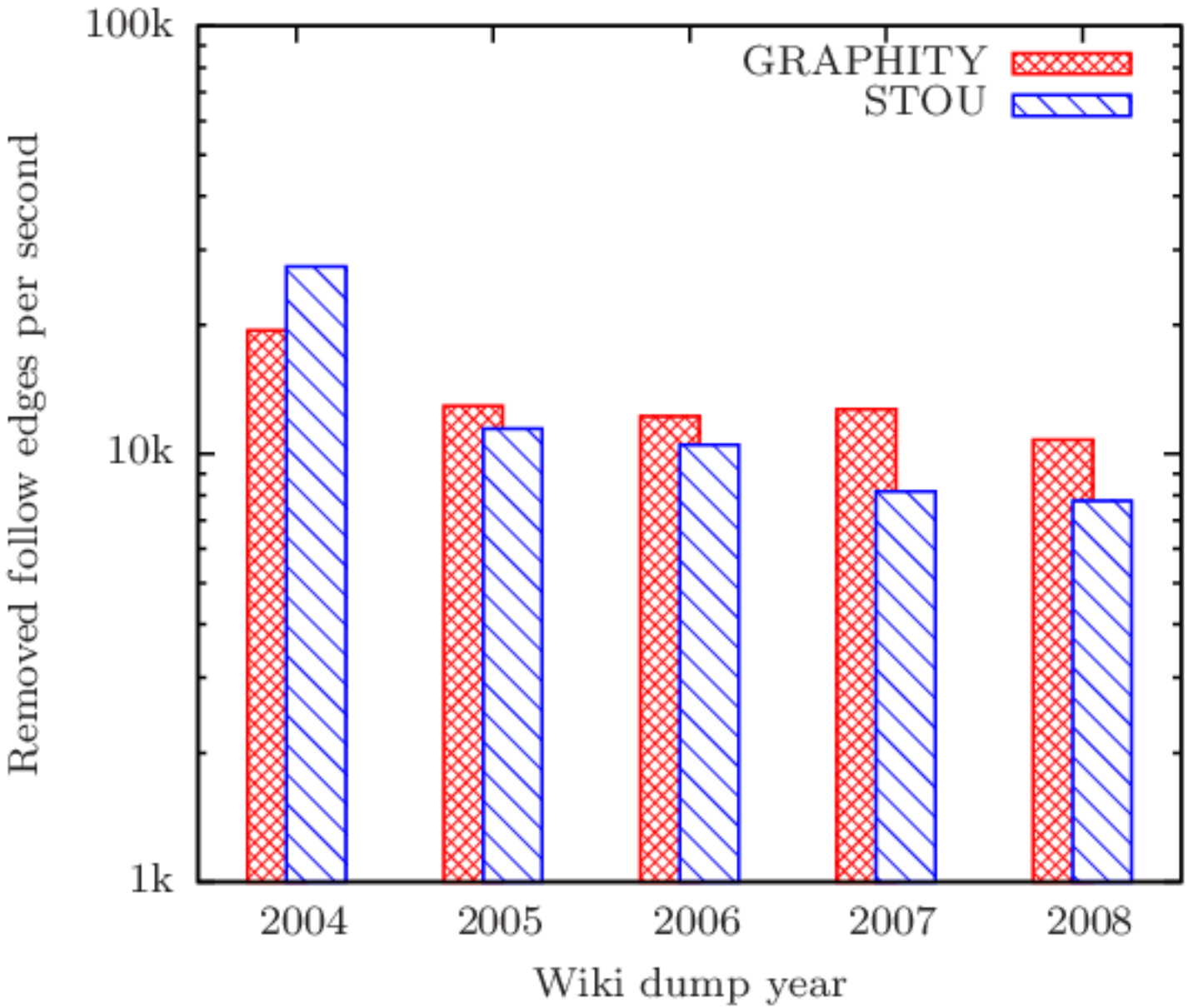
Index maintaining - inserting new content items



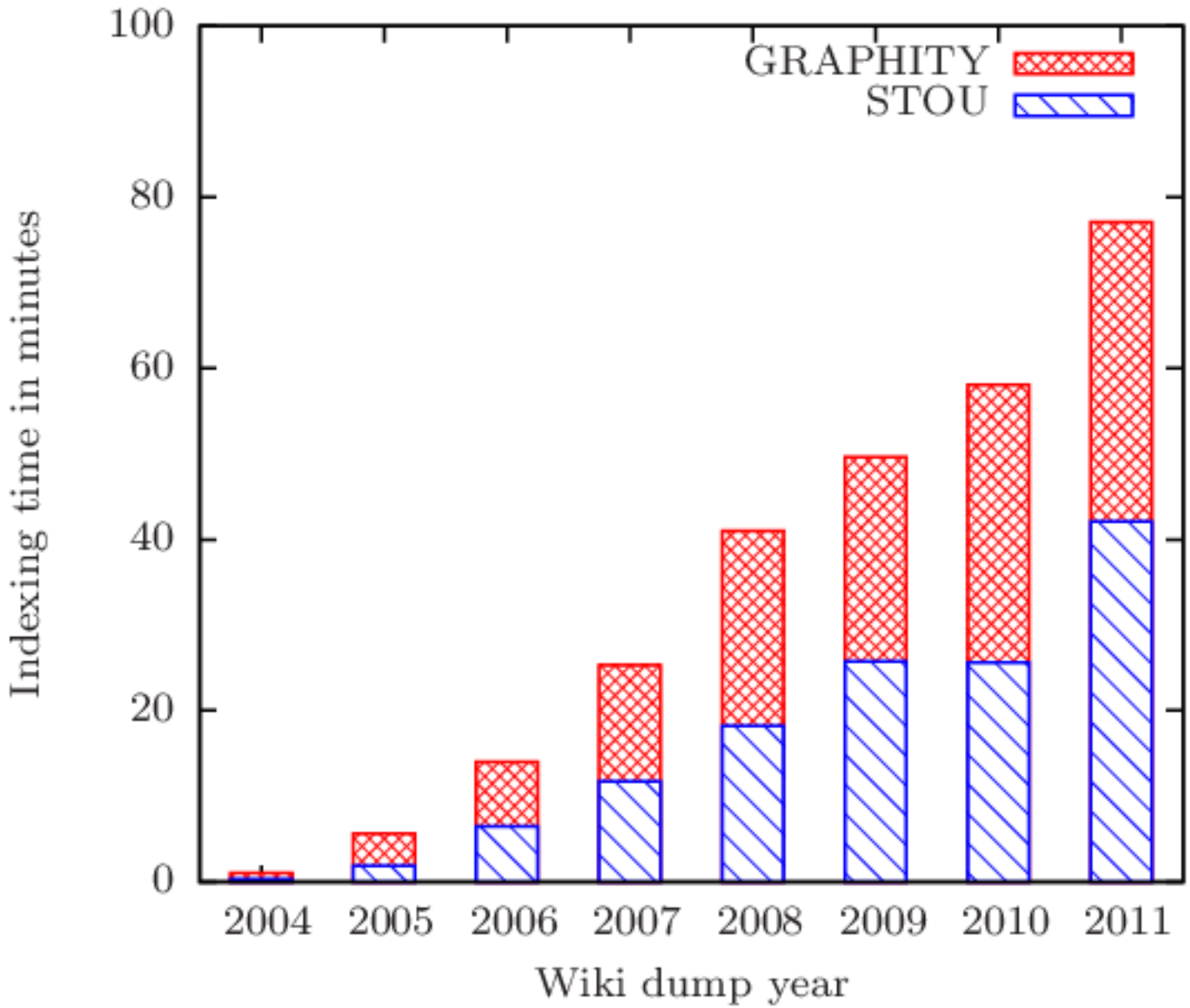
updating graphity for new friendship relations



updating graphity if friendships break



time to build the index



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.

Thank you for your attention

More information + Slides on:

http://www.rene-pickhardt.de/graphity

Subscribe to my newsletter to be the first to receive the paper and get access to the source code as soon as the paper is published.

Thanks to

- my Co-Authors (Dr. Steffen Staab, Jonas Kunze, Dr. Thomas Gottron and Dr. Ansgar Scherp)
- Mattias Persson and Peter Neubauer from neotechnology.com for providing a neo4j fork that was able to store that many different relationship types.
- the community on the neo4j mailinglist for helpful advices on their technology and
- Knut Schumach for coming up with the name GRAPHITY
- Matthias Thimm for helpful discussions

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