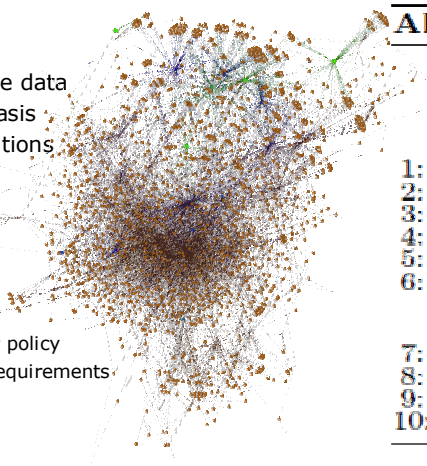
WISE 2014, Oct 12 - 14, Thessaloniki, Greece

A Faceted Crawler for the Twitter Service

George Valkanas, Antonia Saravanou, Dimitrios Gunopulos
Dept. of Informatics & Telecommunications
University of Athens, Greece

Motivation

- **Social Networks**
  - ✔ Are a prolific research area
  - ✔ Have high volumes of diverse data
  - ✔ Used in real life on a daily basis
  - ✔ May boost numerous applications
    - Emergency management
    - News reporting
    - Big Data problem solving
  - … but data retrieval
    - ✘ Is difficult, even with APIs
    - ✘ Requires technical effort
    - ✘ Must respect crawl *politeness* policy
    - ✘ Depends on the application requirements



Objective

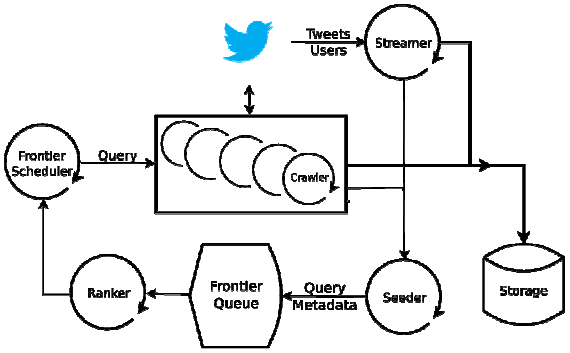
- **Build a Social Network crawler**
  - Focus on Twitter
    - Open to researchers
    - Very active user base
    - High Diversity, in content & users
    - Real-time Social Network
    - Provides APIs
  - Simplify the crawling process
  - Respect crawling constraints
    - Politeness principle
    - Social Network service constraints
  - Allow applications with different requirements to be built

Twitter Service Time Constraints

- Twitter imposes time constraints differently
  - #queries / 15 minutes
  - Different query type → Different constraint

Query	Rate	Max Result	Probe Result	API limit
User Lookup	180	∞	100	100
Tweet Show	180	1	1	1
Friends	15	∞	5000	1
Followers	15	∞	5000	1
Timeline	180	3200	200	1
Retweets	15	100	100	1

Twitter Crawler Architecture



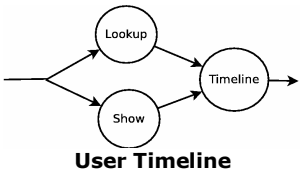
Time-Based Crawling

Algorithm 1 Scheduler Algorithm

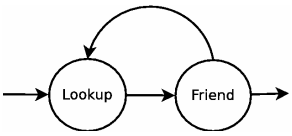
```
Input: Database db, Ranker ranker
Output: outQueue
Shared Queue queue, timedQueue

//Main Thread
1: while !stopped do
2:     qry ← queue.dequeue();
3:     data ← ranker.getNext( qry );
4:     outQueue.enqueue( qry, data );
5:     db.store( qry.qryMeta );
6:     timedQueue.enqueue( qry, NOW + qry.ival );

EventTrigger()
7: nextQuery ← timedQueue.dequeue();
8: top ← timedQueue.top();
9: queue.enqueue( nextQuery );
10: resetTimer( top.TIME - NOW );
```

Simplifying the Crawling Process

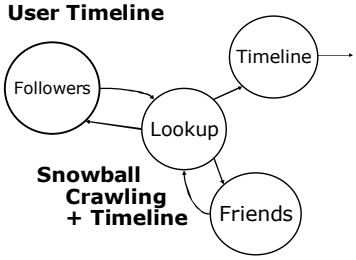
- **Using a Crawl Chain**
  - Sequence of queries that describe the application and the data to be retrieved
  - Only describes *what* queries to use
    - Querying restrictions are handled *internally* by the crawler
  - Implement Seeder / Ranker interfaces
    - Default implementations provided
  - Examples (Implemented)



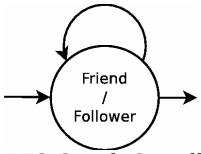
User Timeline



Social Graph Sampling (Metropolis-Hastings)

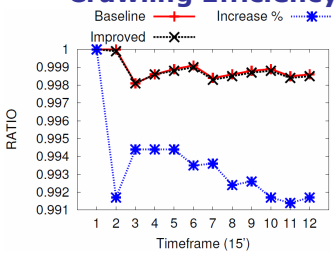


Snowball Crawling + Timeline

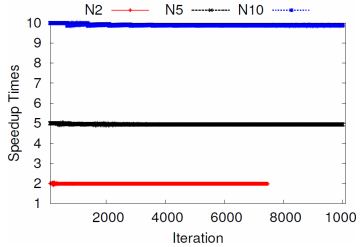


BFS Graph Crawling

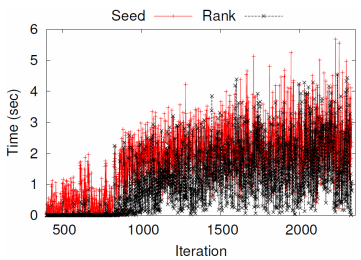
Crawling Efficiency & Effectiveness



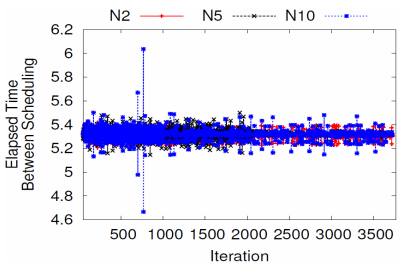
User Info Retrieval



Crawler Speedup



Seeding & Ranking Time



Timeline Interscheduling Time