Fast Similarity Search for Structured P2P Systems

Thomas Bocek[1], Fabio Hecht[1], Ela Hunt [2], David Hausheer[1], and Burkhard Stiller[1,3]

[1] CSG, IFI, UZH
[2] GlobIS, ETH Zurich
[3] CSG, TIK, ETH Zurich
E-Mail: bocek|hecht|stiller@ifi.unizh.ch, hunt@inf.ethz.ch

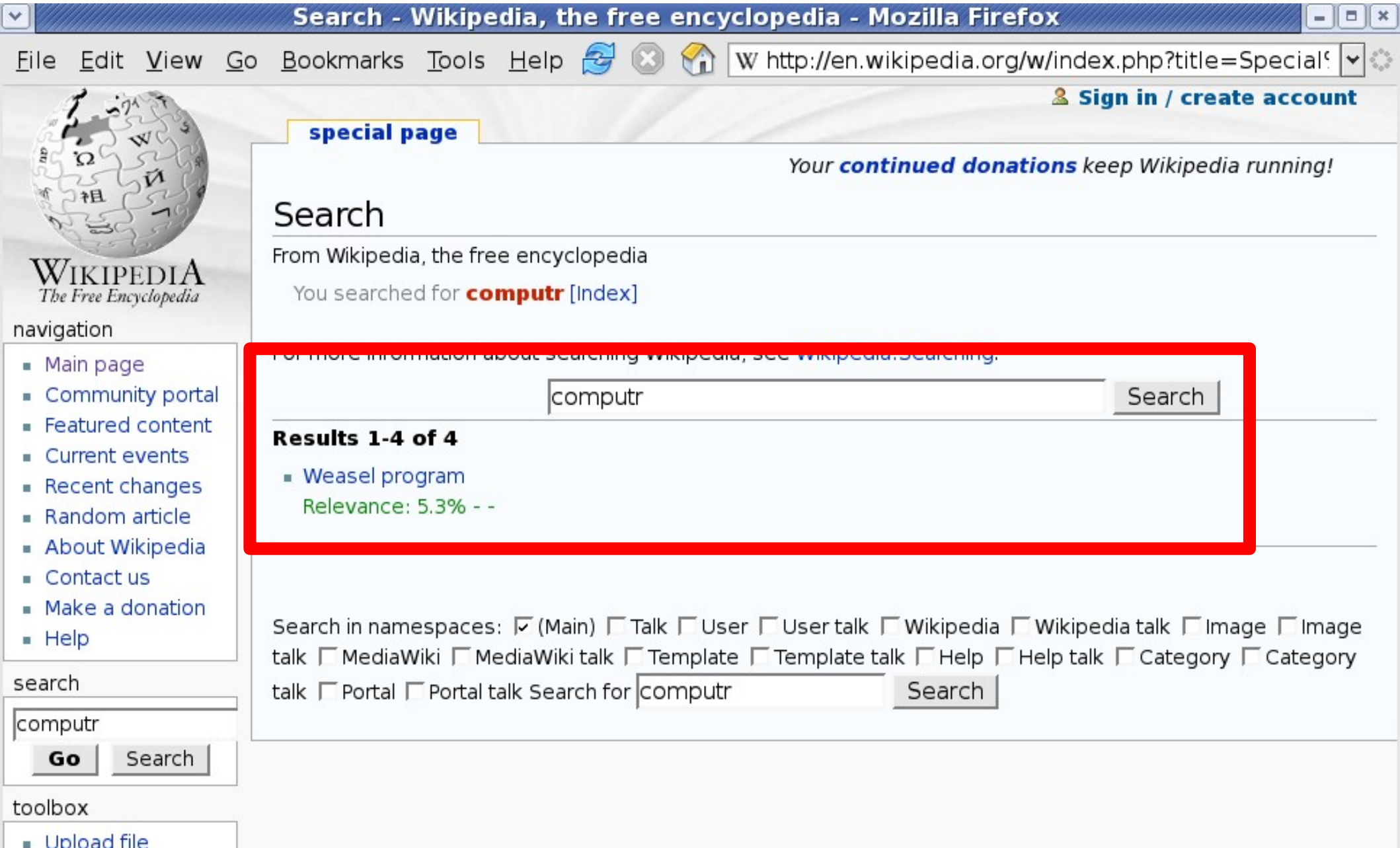
http://fastss.csg.uzh.ch

University of Zurich

Outline

- Motivation
- Related Work
- Fast Similarity Search
- Peer-to-peer **Fast S**imilarity **S**earch (P2PFastSS)
  - Examples
  - Algorithm
  - Performance
- Conclusion
- Demo

Motivation

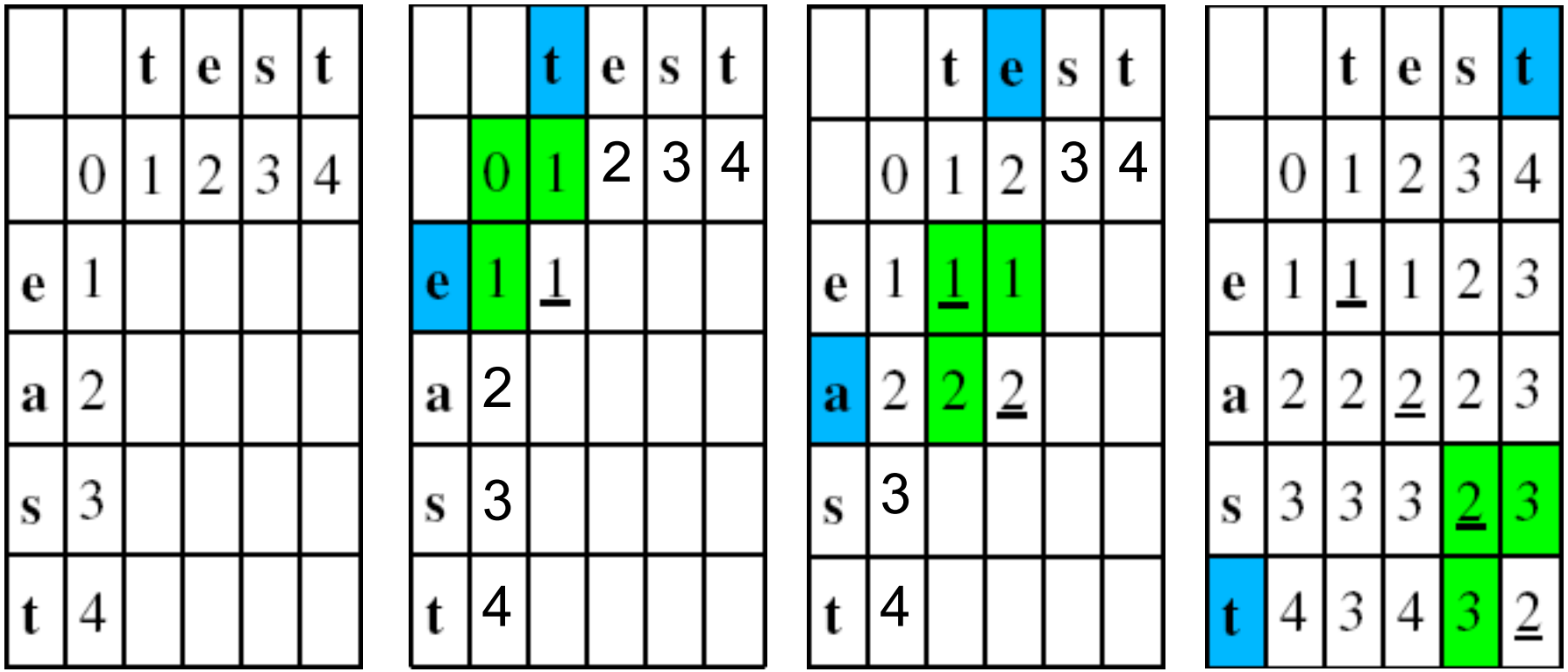

Search - Wikipedia, the free encyclopedia - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://en.wikipedia.org/w/index.php?title=Special
Sign in / create account
special page
Your continued donations keep Wikipedia running!
Search
From Wikipedia, the free encyclopedia
You searched for computr [Index]
computr
Search
Results 1-4 of 4
Weasel program
Relevance: 5.3% - -
navigation
Main page
Community portal
Featured content
Current events
Recent changes
Random article
About Wikipedia
Contact us
Make a donation
Help
search
computr
Go
Search
toolbox
Search in namespaces: (Main) Talk User User talk Wikipedia Wikipedia talk Image Image talk MediaWiki MediaWiki talk Template Template talk Help Help talk Category Category talk Portal Portal talk Search for computr
Search

Related Work – Edit Distance (1)

- Model of similarity: edit distance
- Edit distance between strings
  - Minimum # of operations to transform one into the other
  - Operations:
    - **insert**, **delete**, and **replace**
- Edit distance matrix calculation is costly
  - Uses matrix of size O(mn)

Related Work – Edit Distance (2)

- Example: edit distance (test,east) = 2



Related Work – Neighborhood Generation

- Neighborhood generation:
  - All possible strings for a given k are created
    - Neighbors for *test* with k=2: test, testa, testaa, testab, ... , tea, teb, tec, ..., teaa, teab, ...
  - Problem: large alphabet, large neighborhood size
    - Neighbors for *test* with k=2 result in **23883** neighbors

FastSS Algorithm

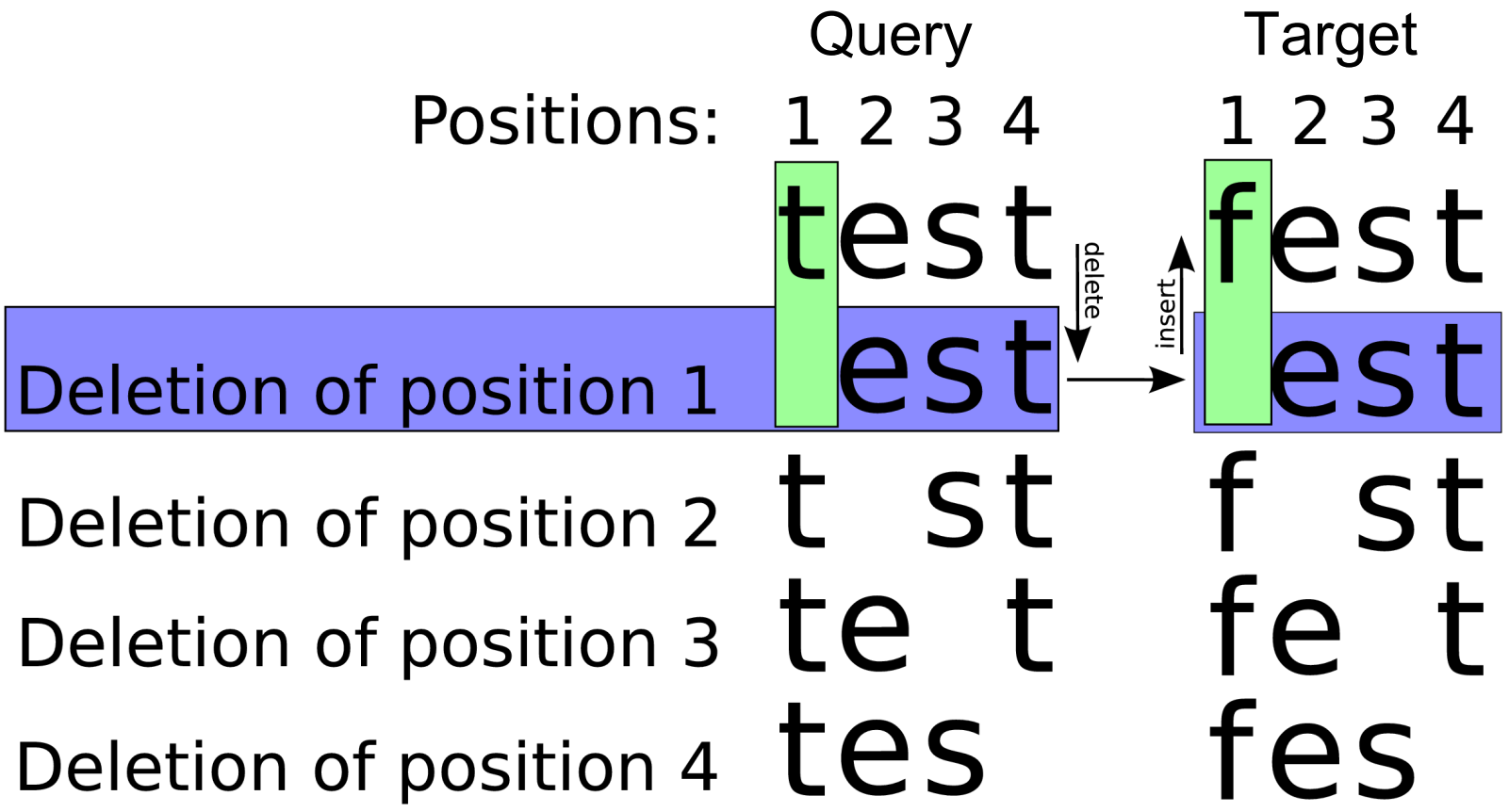
- **FastSS** uses edit distance metric
- Based on neighborhood generation
  - Generate neighbors with **deletions** on the query **and** target
  - Example test with k=2, neighborhood generation based on deletion:
    - test, est, st, et, es, tst, tt, ts, tet, te, tes

FastSS and NG

- FastSS does not generate as many neighbors as neighborhood generation
  - FastSS: 11 neighbors, enlarged target by 11 neighbors → **11 queries**
  - Full neighborhood generation: 23883 neighbors, target is not enlarged → **23883 queries**
- FastSS examples
  - In the following examples, search for k=1, 1-deletion neighborhood

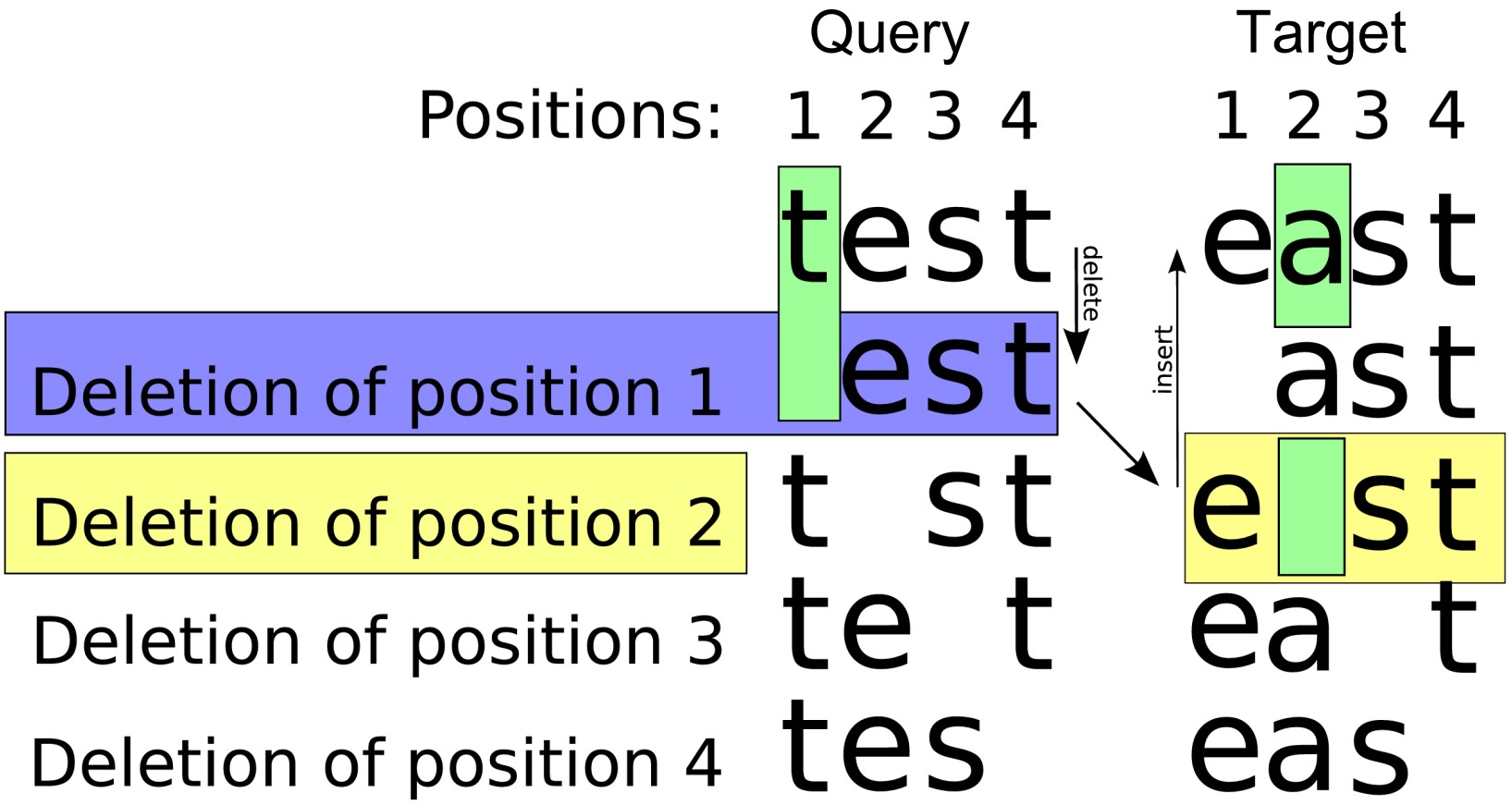
FastSS Example (1)

- Edit distance (test,fest) = 1



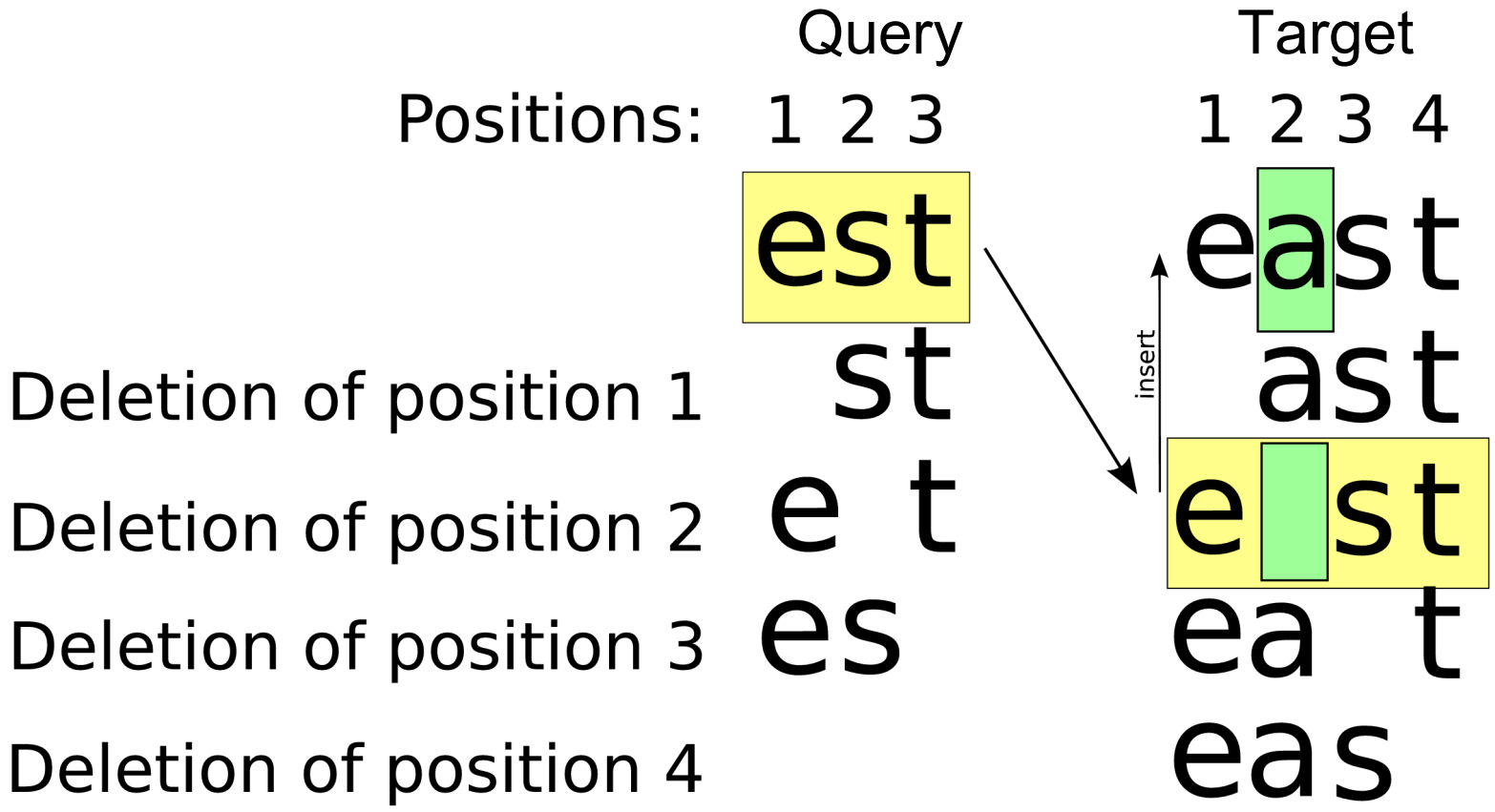
FastSS Example (2)

- Edit distance (test,east) = 2
  - Different delete positions



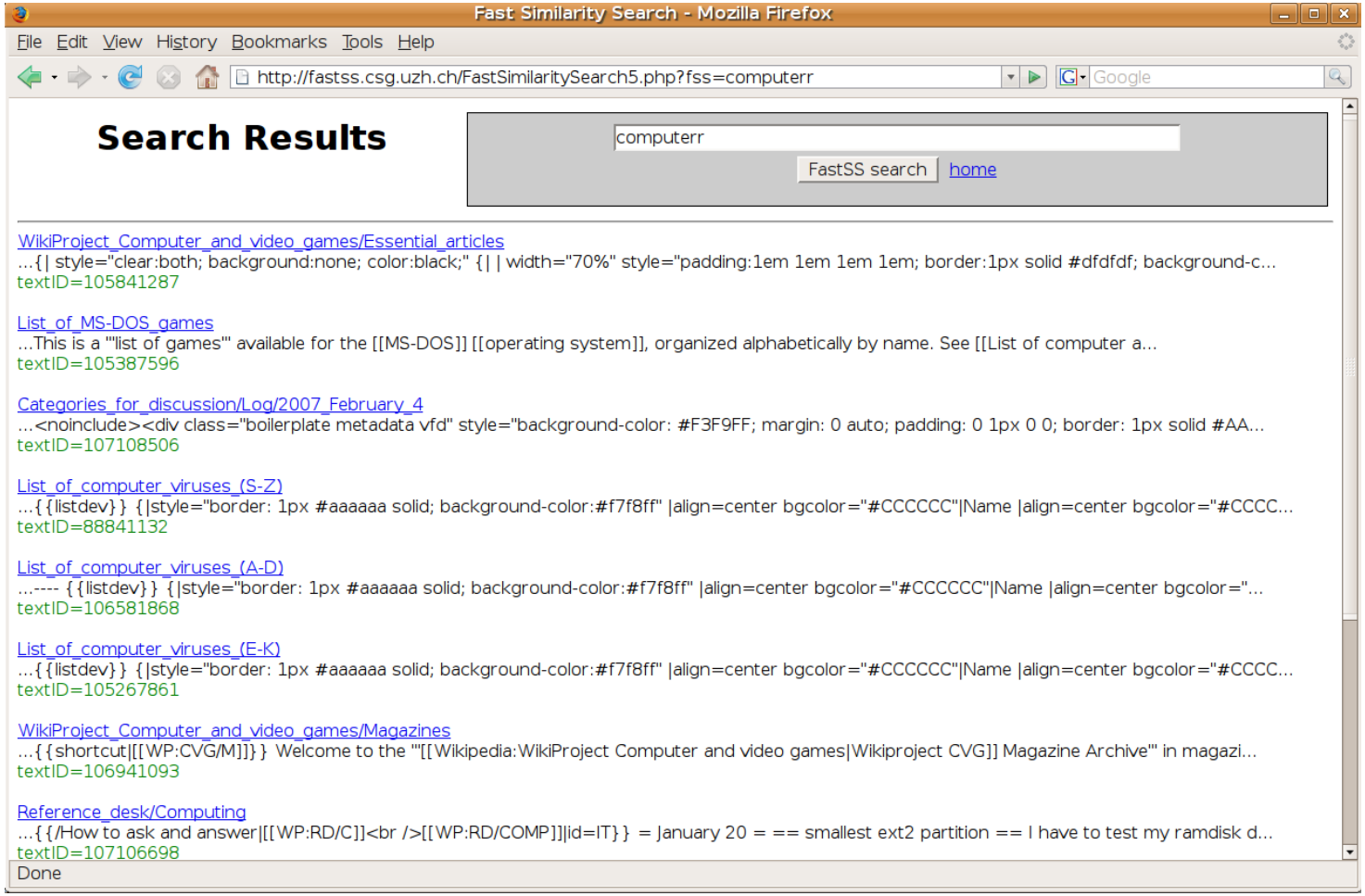
FastSS Example (3)

- Edit distance (est,east) = 1
  - Different word length



FastSS on Wikipedia

- FastSS in a centralized system using PHP and MySQL (indexed complete English Wikipedia)



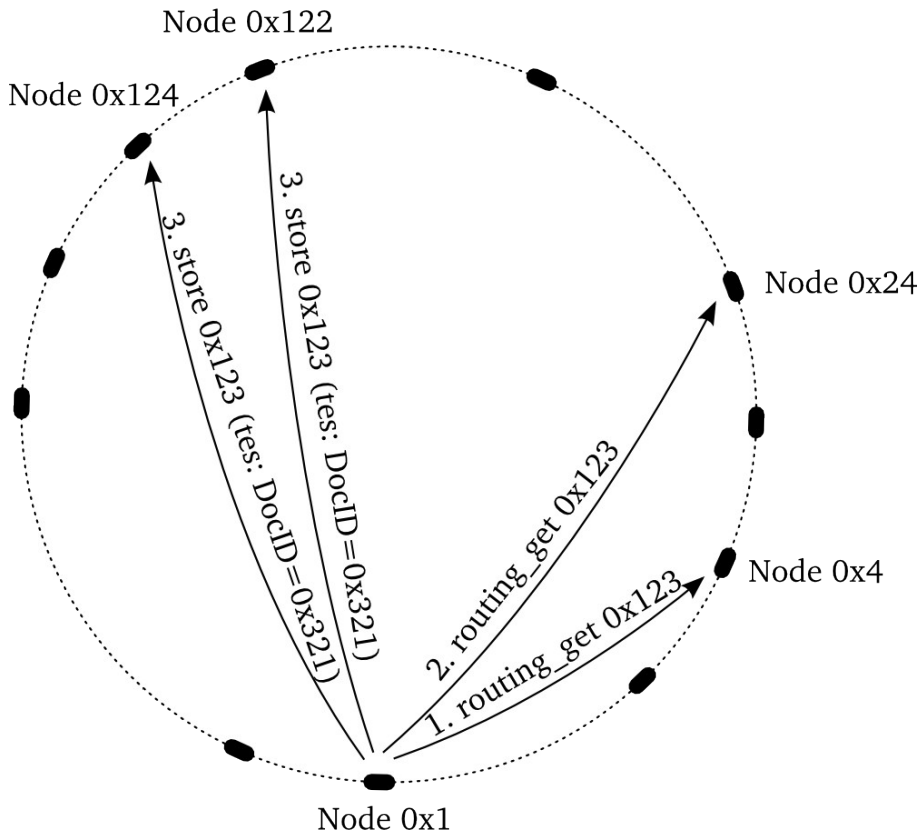
FastSS on Wikipedia

- Scalability issues (1 similarity search ~ 1-2s)
  → Distributed system for better scalability
- No support for similarity search in DHT
  - Operations: `get(hash)`, `put(hash,value)`
  - Only exact matches are returned

→ Use FastSS on top of DHT

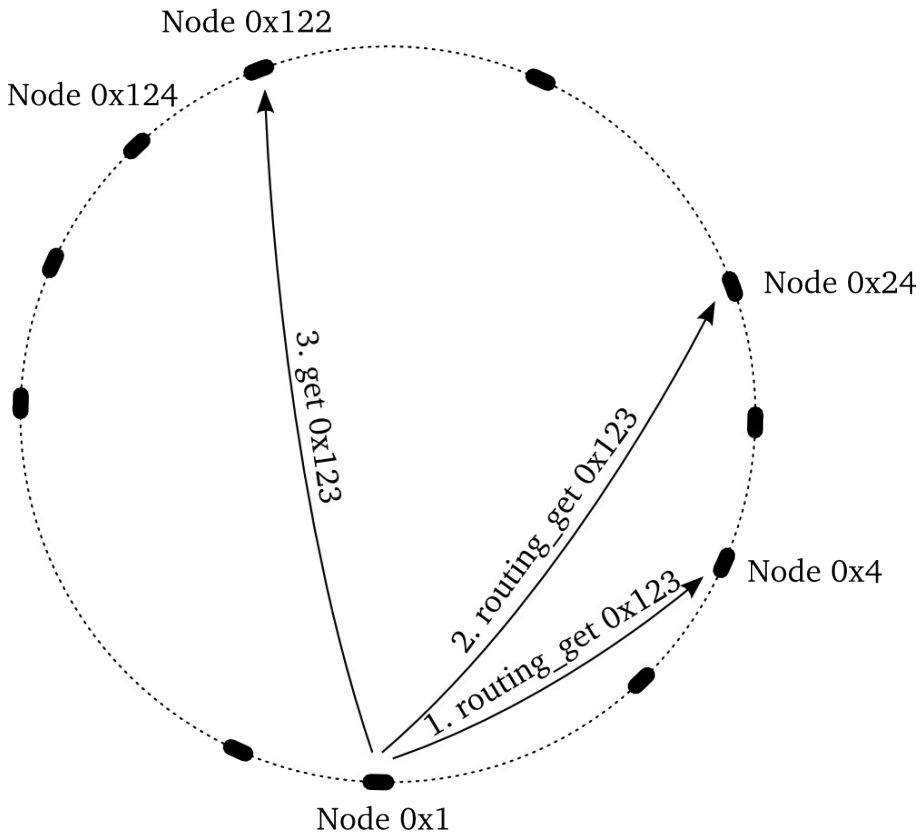
P2PFastSS – Algorithm Indexing

- Index documents using `put(hash(document),document)`
- Index all neighbors (test,tes,tst,tet,est) using `put(hash(neighbor),point to document)`



P2PFastSS – Algorithm Search

- User searches for “tesx”
- Neighbors are generated (tesx, esx, tsx, tex, tes)
  - `get(hash(neighbor))`
  - Yields pointer to document
  - `get(hash(document))`



P2PFastSS – Implementation

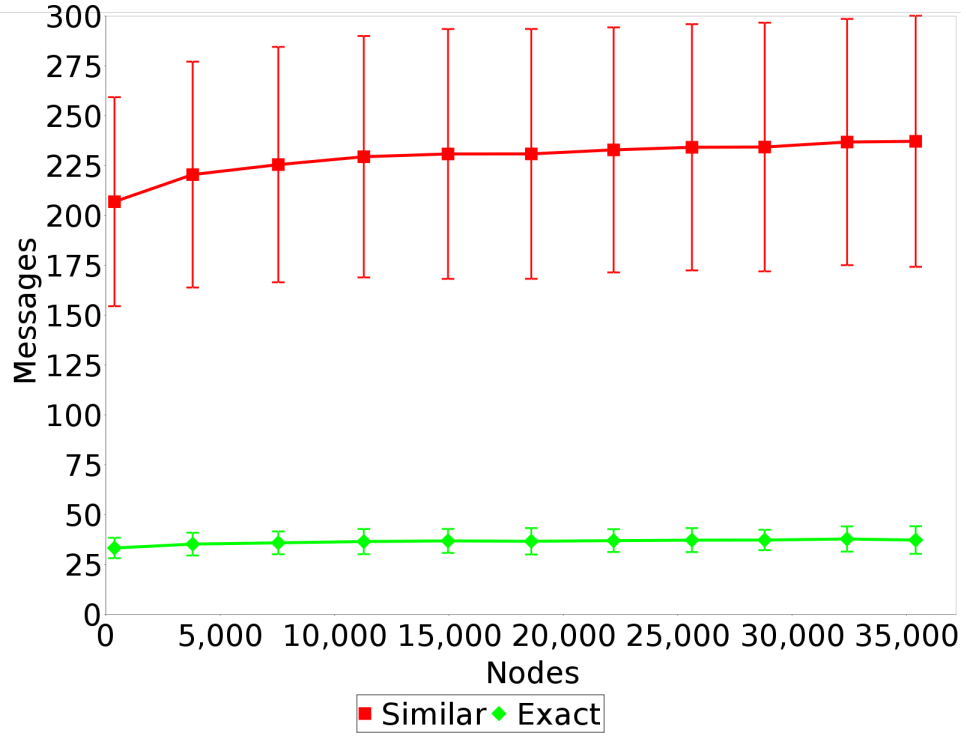
- P2PFastSS
  - Implemented in Java
  - Uses a DHT based on the Kademlia routing algorithm
  - Deployed on ~360 PlanetLab hosts
    - up to 100 nodes on each PlanetLab host
    - up to 34,200 nodes in total
    - new tests use EMANICSLab
  - Edit distance (k) set to 1
  - Every word with length 3 to 16 was indexed

P2PFastSS – Performance

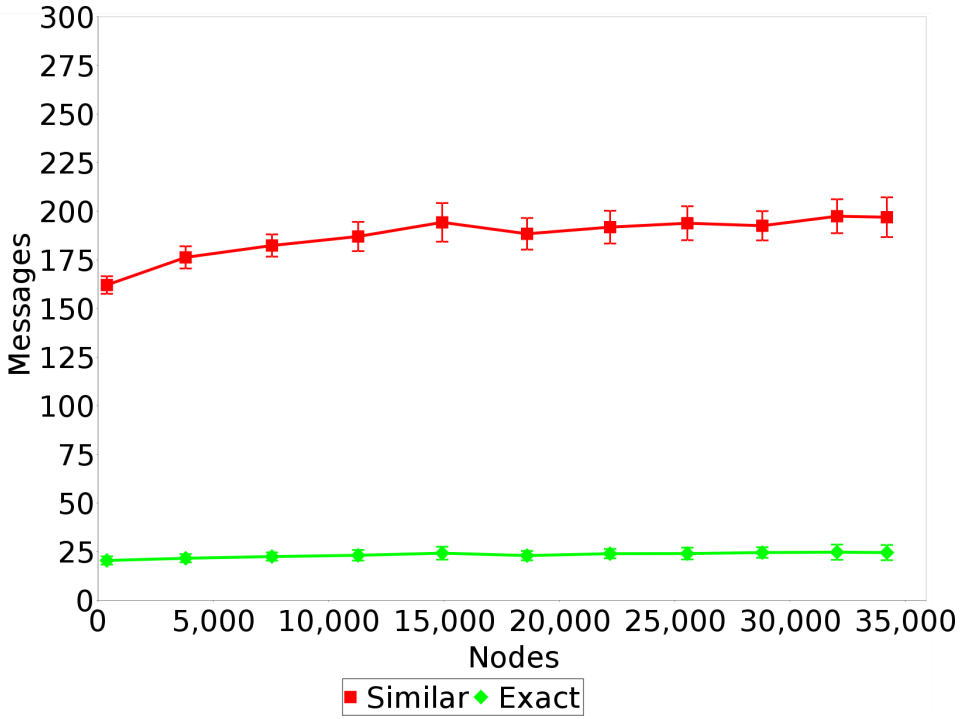
- 100 Wikipedia abstracts indexed
  - Total 2,392 words
  - Average word length is 7 characters
- Message, time, and storage measurements
- All experiments carried 50 times
  - Average values shown, with error bars

P2PFastSS – Performance
Number of Messages for Indexing

- m is between 3 and 16
  - High value for standard deviation
  - Short words need less messages
- Redundantly stored in 2 nodes



P2PFastSS – Performance
Number of Messages on Search



- Word length 7
- Overhead introduced by P2PFastSS is m^k
  - m is average word length (7)
  - k is edit distance (1)
- Logarithmic growth observed

P2PFastSS – Performance Time Measurements

- Indexing time
  - Similarity indexing
    - 0.67 to 16.99s
  - Exact indexing
    - 0.18 to 15.94s
- Lookup time
  - Similarity search
    - 0.5 to 11.6s (average is less than 3s)
  - Exact search
    - 0.2 to 4.5s (average about 2s)
- High variability due to real-world conditions
- Storage operation is slower than searching
  - keywords are stored with the redundancy factor r (2)

Conclusions

- Message overhead
  - Ca. seven times that of exact search
- P2PFastSS
  - Only 1.5 times slower than an exact search
  - Edit distance 1
- Difference due to benefits of distributed parallel computation
- P2PFastSS performs a similarity search in average in less than 3 s with more than 34,000 nodes on PlanetLab
  - Average load average on PlanetLab ~9.9

Demo

Demo

Thank You For Listening

Questions?

bocek@ifi.uzh.ch
hecht@ifi.uzh.ch

http://fastss.csg.uzh.ch