

# DetectIT: A Peer-to-Peer Plagiarism Detection System

Elif Tosun, Ben Wellington  
Honors OS: Final Project

# Outline

- Introduction
- Background
- Fingerprinting
- DetectIT: System Description
- Evaluation
- Conclusions

# Introduction

## THE PROBLEM

- 17 % of college students (in humanities) are submitting plagiarized documents.
- Mr. X case
- Need a system to:
  - Detect copy/plagiarism
  - Protect student's intellectual property
  - Easily scalable and deployable throughout universities
  - Easy maintenance

# Introduction

- Ideal Solution:
  - Keep track of all documents ever written
  - Compare a given document to all others
- Problems:
  - Takes too much space
  - Too much computing power/time
  - Single point of failure
- Nature of Problem: The more users the more effectiveness
- Solution: Peer2Peer System!

# Related Work

- Registry Servers
- Single DB Solutions
  - SCAM, COPS, CHECK, Koala, SE
  - Biggest Problem: one server
- Multiple DB Solution
  - dSCAM
  - Search many databases by refining...
  - Too complicated – not transparent at all
  - Not P2P

# Background

- ***Tapestry***
  - P2P location and routing system
  - Similar to Pastry, Chord, CAN...
  - Point to point links based on matching digits in IDs.
  - Neighborhood maps
  - Supports logarithmic number of hops.

# Background

- ***ATA (Approximate Text Addressing)***
  - Instead of one GUID, we want to search a vector of fingerprints that define the doc
  - Implemented by Zhou et al at Berkeley.
  - Storage:
    - An object/fp that has a list of all docs that have that fp
    - GUID of doc
  - Publishing:
    - Publish GUID of doc
    - Publish each fp, if already in sys append GUID, else create a new one.
  - Search:
    - Objects corresponding to fps are searched.
    - The GUID appearing in more than T lists is selected

# Fingerprinting

- A single fingerprint is a hash of set of characters
- Ideal Solution for copy detection:
  - Compare all sets of one document to other
    - EXTREMELY costly
- We want a subset of document's all fingerprints – which subset??

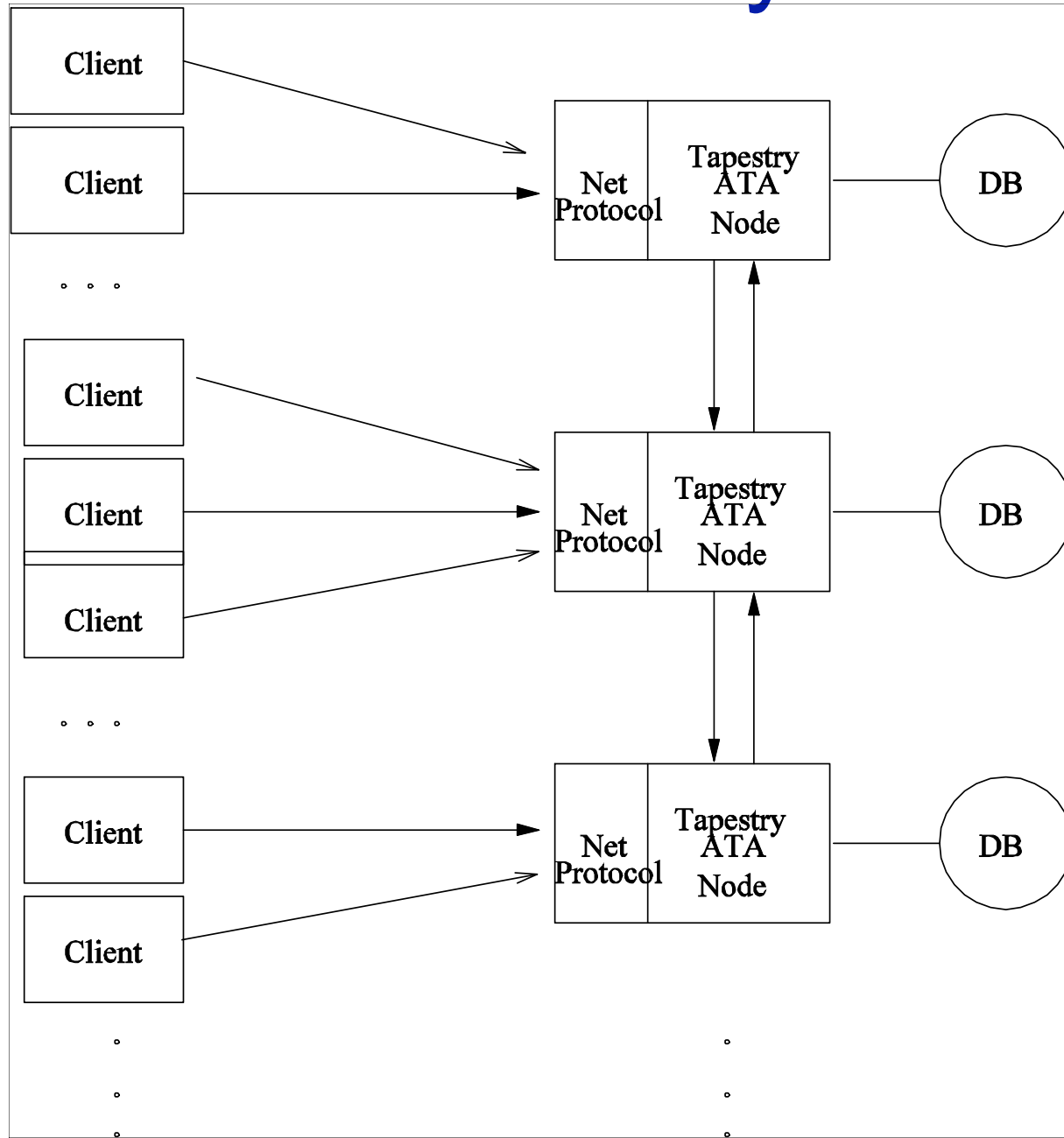
# Fingerprinting

- Random Selection ? – Poor Results

*Must select similar fingerprints from similar documents!*

4. Hash Fingerprints
5. Keep all fingerprints divisible by  $M$
6. Take minimum  $k$  of leftover prints

# DetectIT: The System



# Client Component

- Runs on any platform (yay JAVA!)
- Responsible for fingerprinting
- Uses sockets to communicate with tapestry nodes
- Sends: Document information and Finger Prints
- Receives: - “all clear” or “suspected” with relevant information.

# Tapestry/ATA Component

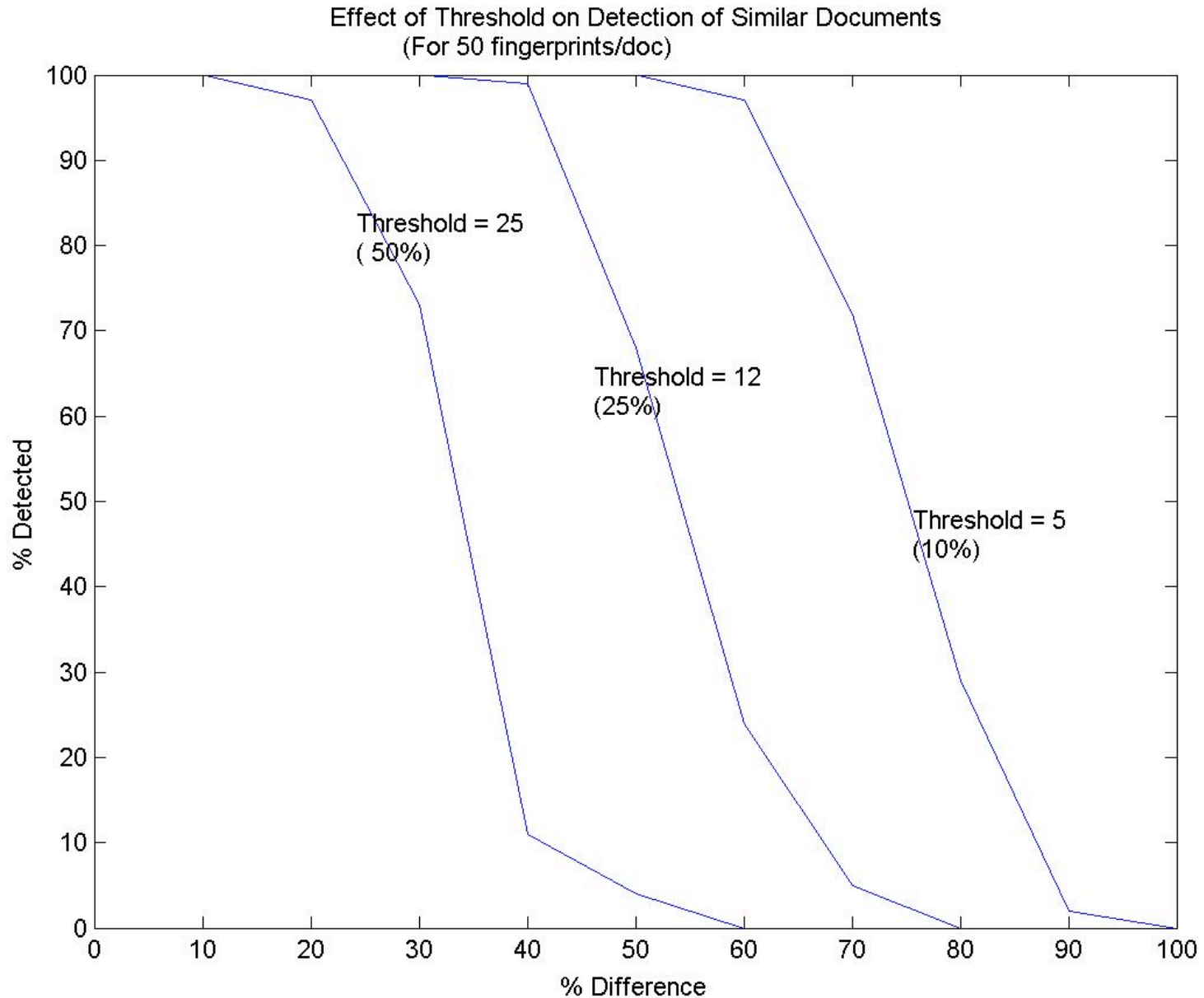
- Follow ATA search alg.
- Whether a doc is returned or not:  
STORE
- Compare authors before sending a rslt to client.

# DB Component

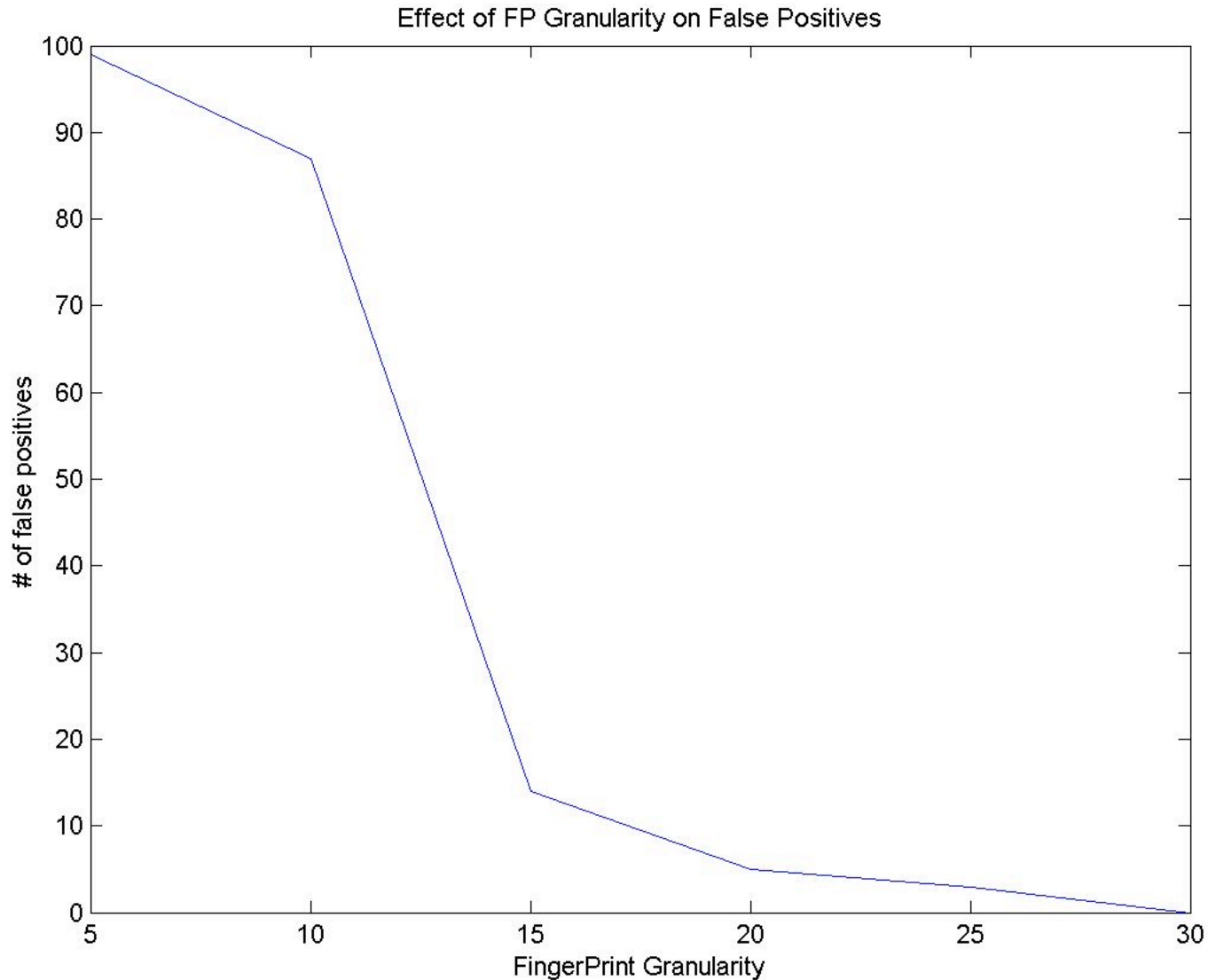
- 2 databases:
  - Set of fps -> ID
  - ID ->author, prof, date of submission

***When a system first starts, all elements in the db are published into the system!***

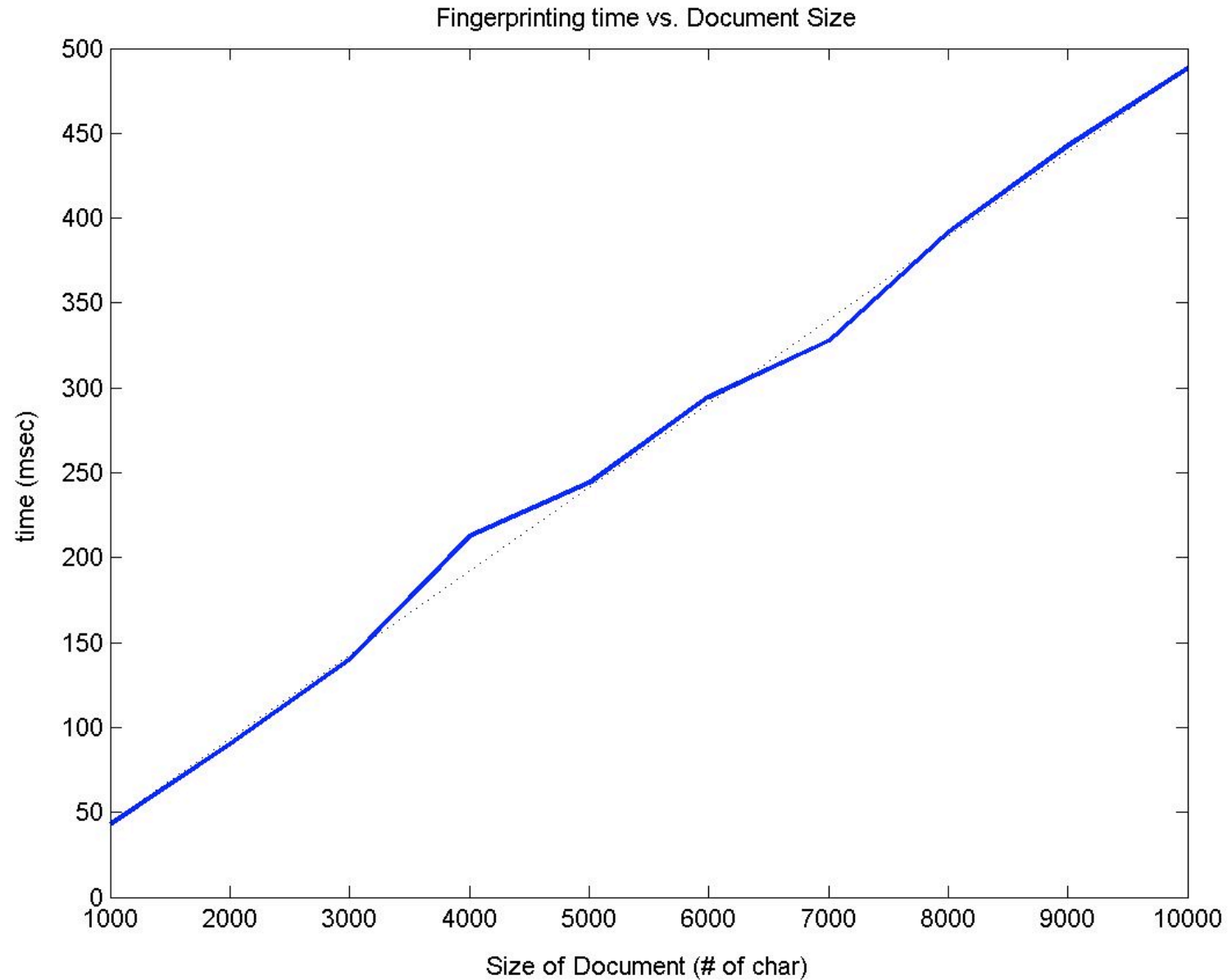
# Evaluation - Correctness



# Evaluation – Correctness



# Evaluation - Performance



# Evaluation - Performance

	Misses (sec/doc)	Hits (sec/doc)
1 Tapestry Node	.075	0.75
2 Tapestry Nodes	.083	0.75
3 Tapestry Nodes	.130	0.74
4 Tapestry Nodes	.150	0.75

# Evaluation – Short Comings

- Fixed fingerprints
- Can't detect if “combined” plagiarism
- Cannot categorize: plagiarized, subset, exact copy, related => only have “suspected/cleared”
- DetectIT vs. Internet services
- Different document format support

# Conclusions

- Goal Achieved!
    - Built a peer 2 peer copy detection system
    - Fault tolerant, Scalable ...
  
    - Avg Turn around time < .75 sec / doc  
(Including fingerprinting and detection)
- Reasonably fast performance!!