

Top-K Algorithms: Concepts and Applications

by

Demetris Zeinalipour

Visiting Lecturer

Department of Computer Science

University of Cyprus

***Tuesday, March 20th, 2007, 15:00-16:00, Room #147 Building 12
EPL 671: Computer Science: Research and Technology
Department of Computer Science - University of Cyprus***



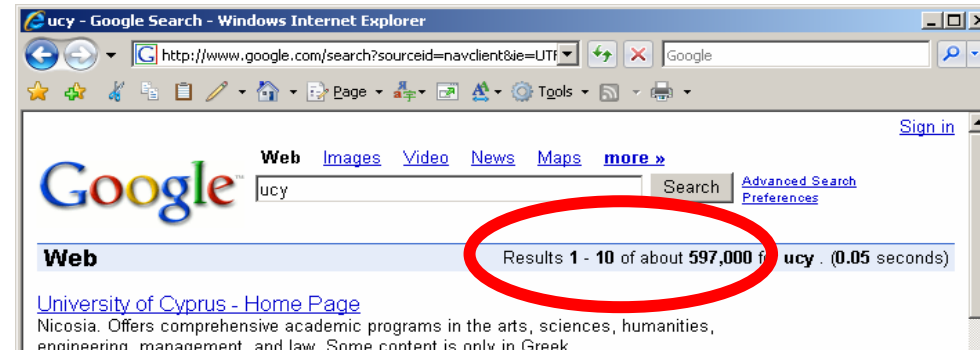
Presentation Goals

- To present the **concepts** behind **Top-K algorithms** for **centralized** and **distributed** settings.
- To present **applications** in which Top-K query processing can yield significant savings in CPU, bandwidth, latency, etc.
- To present the intuition behind the family of **Top-K query processing algorithms** we developed and evaluated.



Motivation

- Clients want to get the right answers quickly.
- Clients are not willing to browse through the complete answer-set.
- Service Providers want to **consume the least possible resources** (disks, network, etc).



In many scenarios it makes sense to focus on the K highest ranked answers (or Top-K) answers rather than finding all of them.



Presentation Outline

A. Top-K Algorithms: Definitions

B. Centralized Top-K Query Processing

- The Threshold Algorithm (TA)

C. Distributed Top-K Query Processing

- The Threshold Join Algorithm (TJA)
- Experimentation using 75 workstations

D. Other Applications of Top-K Queries

- Distributed Spatio-temporal Trajectory Retrieval



- In-Network Top-K Views (MINT Views)

Definitions

Top-K Query (Q)

*Given a **database** D of n objects, a **scoring function** (according to which we rank the objects in D) and the number of **expected answers** K , a Top-K query Q returns the K objects with the highest score (rank) in D .*

Objective:

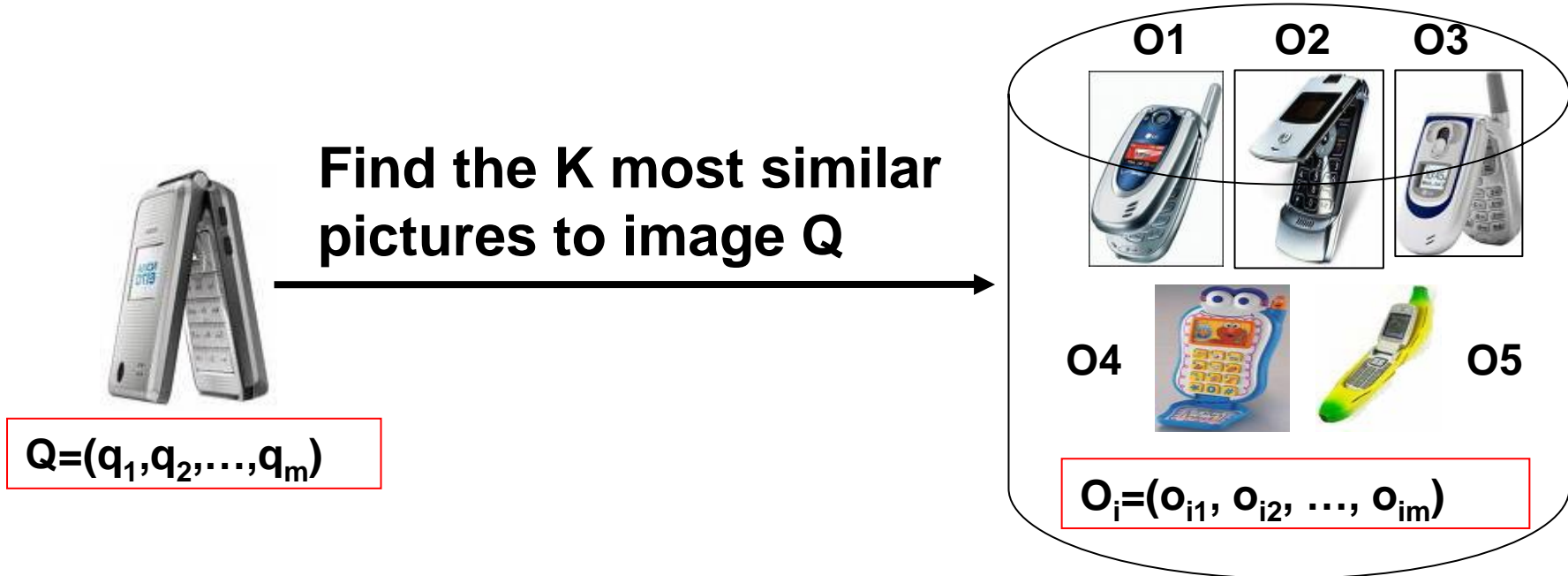
Trade # of answers with the query execution cost, i.e.,

- Return less results ($K \ll n$ objects)
- ...but minimize the **cost that is** associated with the retrieval of the answer set (i.e., disk I/Os, network I/Os, CPU etc)



Definitions

- Assume the following Query-By-Example Scenario in Multimedia Content-Retrieval



- Q and O_i ($i \leq m$) are expressed as **vectors of features**
e.g. $Q=(\text{"color=\#CCCCCC", "texture=110", shape="\wedge", \dots, })$
- Answers are inherently **fuzzy**, i.e., each answer is associated with a score $(O3, 0.95), (O1, 0.80), (O2, 0.60), \dots$



Definitions

The Scoring Table

An m -by- n matrix of scores expressing the similarity of Q to all objects in D (for all attributes).

*In order to find the K highest-ranked answers we have to compute **Score**(o_i) for all objects*

*(requires $O(m*n)$ time). $Score(o_i) = \sum_{j=1}^n w_j * sim(q_j, o_{ij})$*

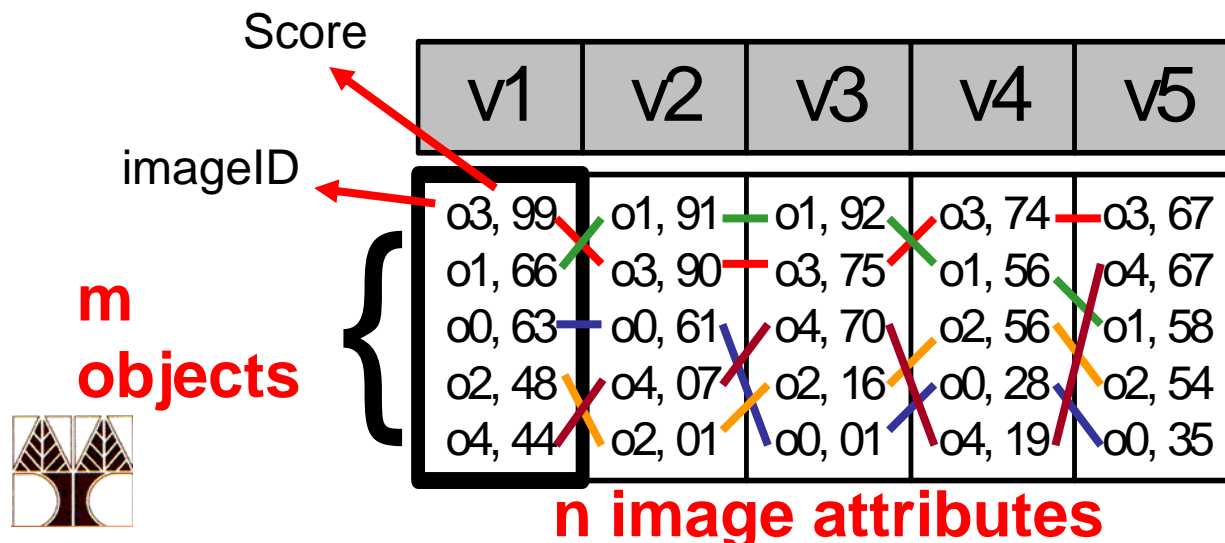
Score

imageID

m
objects

	v1	v2	v3	v4	v5
{	o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
	o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
	o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
	o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
	o4, 44	o2, 01	o0, 01	o4, 19	o0, 35

n image attributes



The diagram illustrates a scoring table with 5 columns (v1 to v5) and 5 rows of object data. A red arrow labeled 'Score' points to the top row of the table. A red arrow labeled 'imageID' points to the first column of the table. A red arrow labeled 'm objects' points to the first column of the table. A red arrow labeled 'n image attributes' points to the first row of the table. The table contains the following data:

	v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	

Annotations include colored lines connecting objects across columns: green lines connect o3, 99 to o1, 91 and o1, 91 to o1, 92; red lines connect o3, 99 to o3, 74 and o3, 74 to o3, 67; blue lines connect o1, 66 to o3, 90 and o3, 90 to o3, 75; orange lines connect o0, 63 to o4, 70 and o4, 70 to o2, 56; yellow lines connect o2, 48 to o4, 07 and o4, 07 to o2, 16; and purple lines connect o4, 44 to o2, 01 and o2, 01 to o0, 01.

Presentation Outline

A. Top-K Algorithms: Definitions

B. Centralized Top-K Query Processing

- The Threshold Algorithm (TA)

C. Distributed Top-K Query Processing

- The Threshold Join Algorithm (TJA)
- Experimentation using 75 workstations

D. Other Applications of Top-K Queries

- Distributed Spatio-temporal Trajectory Retrieval



In-Network Top-K Views (MINT Views)

Centralized Top-K Query Processing

Fagin's* Threshold Algorithm (TA):

(In ACM PODS'02)

* Concurrently developed by 3 groups

The most widely recognized algorithm for Top-K Query

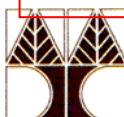
Processing in database systems



TA Algorithm

- 1) Access the n lists in parallel.
- 2) While some object o_i is seen, perform a **random access** to the other lists to find the complete score for o_i .
- 3) Do the same for all objects in the current row.
- 4) Now compute the threshold τ as the **sum of scores** in the current row.
- 5) The algorithm stops after K objects have been found with a score above τ .

v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35



Centralized Top-K: The TA Algorithm (Example)

v1	v2	v3	v4	v5	TOP-K
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	O3, 405 O1, 363 O4, 207
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	

Iteration 1 Threshold

$$\tau = 99 + 91 + 92 + 74 + 67 \Rightarrow \tau = 423$$

Have we found K=1 objects with a score above τ ?

=> NO

Iteration 2 Threshold

$$\tau \text{ (2nd row)} = 66 + 90 + 75 + 56 + 67 \Rightarrow \tau = 354$$

Have we found K=1 objects with a score above τ ?

=> YES!

Why is the threshold correct?

It gives us the maximum score for the objects we have not seen yet ($\leq \tau$)

Presentation Outline

A. Top-K Algorithms: Definitions

B. Centralized Top-K Query Processing

- The Threshold Algorithm (TA)

C. Distributed Top-K Query Processing

- The Threshold Join Algorithm (TJA)
- Experimentation using 75 workstations

D. Other Applications of Top-K Queries

- Distributed Spatio-temporal Trajectory Retrieval

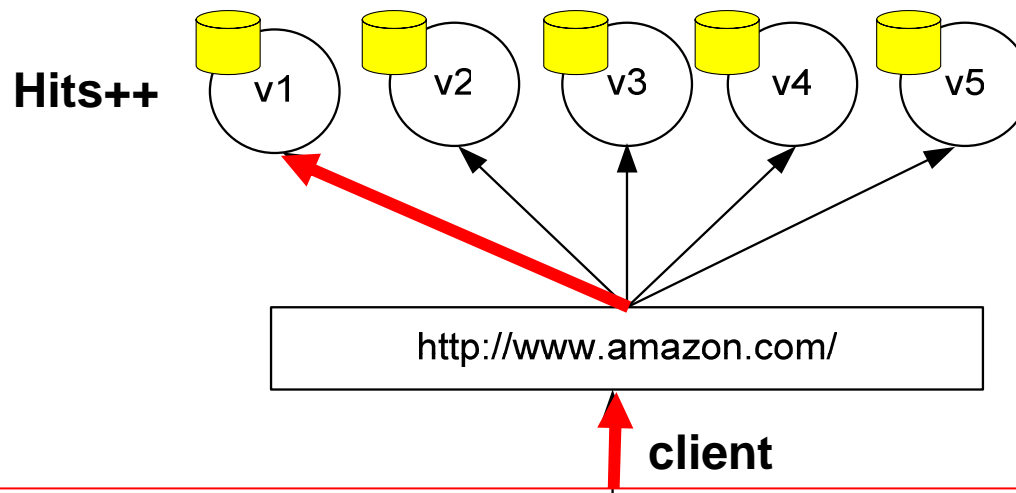


- In-Network Top-K Views (MINT Views)

Distributed Top-K Query Processing

Motivating Example

- We have a cluster of **n=5 Web-servers**.
- Each server maintains locally a replica of the same **m=5 static Web-pages**.
- When a web page is accessed by a client, the respective server increases a local **hit counter** by one.



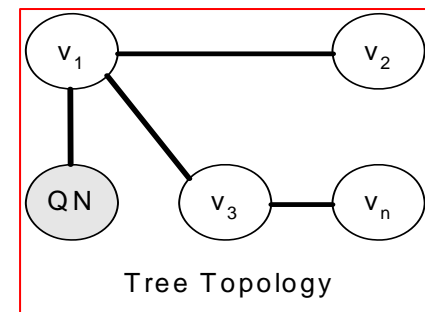
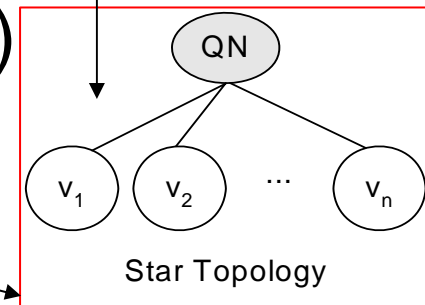
TOP-1 Query: “Find the webpage with the highest number of hits across all servers”

Distributed Top-K Query Processing

- The scoring table is now **vertically fragmented** across N remote sites.
- Each site is accessible over a **fundamentally expensive network**.
- Each site is accessible **directly** (Our example) or **indirectly** (P2P and Sensor Nets)



v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35



Distributed Top-K Query Processing

- **Is the TA Algorithm efficient when the scoring table is vertically fragmented?**

v1	v2	v3	v4	v5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35

- **Answer: No**, because in TA we have an **arbitrary number of phases** (iterations).
- Each iteration introduces additional **latency and messaging**, making it expensive for a distributed environment.



The Centralized Join Algorithm (CJA)

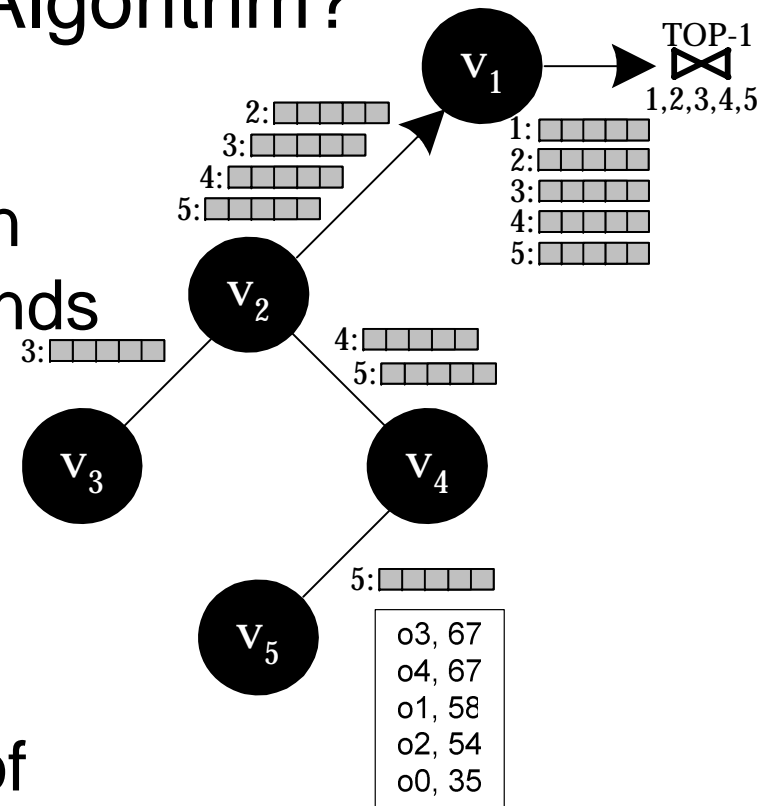
- **Problem:** How to overcome the arbitrary phases of the Threshold Algorithm?

- **Naive solution:**

- Perform the computation in **one phase**: each node sends its complete list of scores
- Each intermediate node forwards all received lists

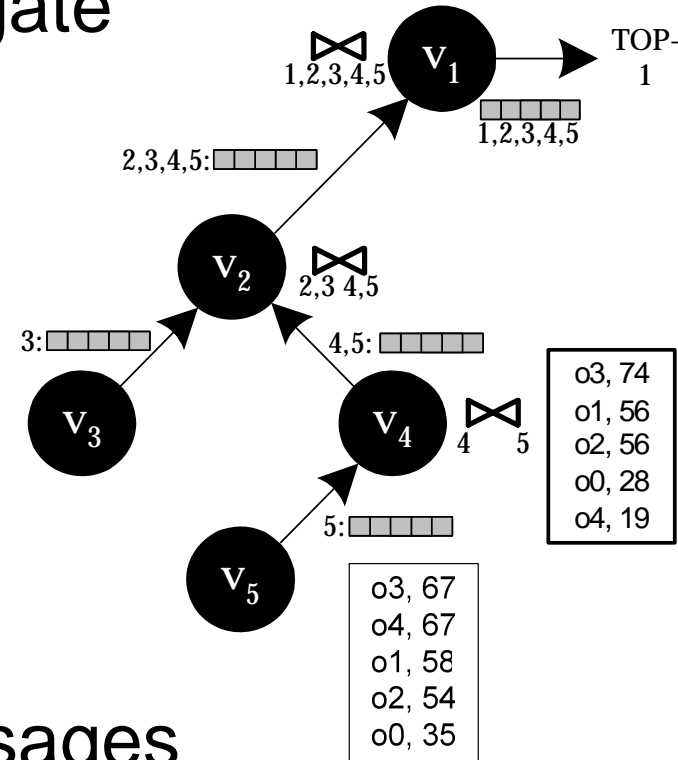
- **Disadvantage**

- Overwhelming amount of messages.
- Huge Query Response Time



The Staged Join Algorithm (SJA)

- **Improved Solution:** Aggregate the lists before these are forwarded to the parent:
- This is the **In-network aggregation** approach
- **Advantage:** Only $O(n)$ messages
- **Disadvantage:** The size of each message is still very large in size (i.e., the complete list)



Threshold Join Algorithm (TJA)

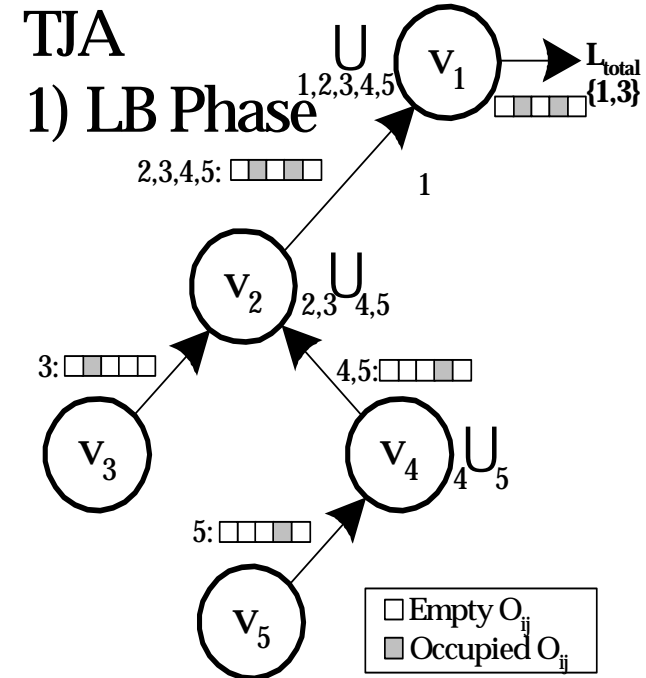
- TJA is our **3-phase algorithm** that optimizes top-k query execution in distributed (hierarchical) environments.
- **Advantage:**
 - **It usually** completes in **2 phases**.
 - **It never completes** in more than **3 phases** (LB Phase, HJ Phase and CL Phase)
 - It is therefore highly appropriate for distributed environments

"The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks", D. Zeinalipour-Yazti et. al, Proceedings of the 2nd international workshop on Data management for sensor networks DMSN (VLDB'2005), Trondheim, Norway, ACM Press; Vol. 96, 2005.



Step 1 - LB (Lower Bound) Phase

- Recursively send the K highest objectIDs of each node to the sink.
- Each intermediate node performs a **union** of the received results (defined as τ):

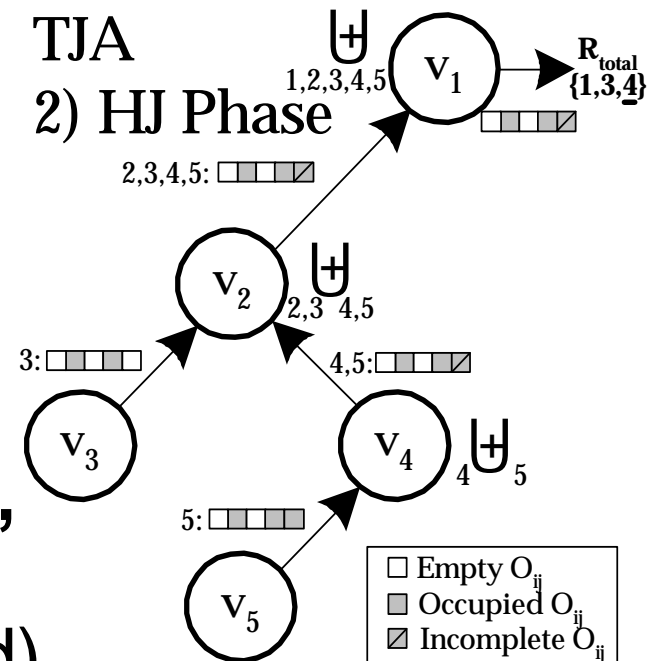


v1	v2	v3	v4	v5	LB
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	$T=\{o3, o1\}$ Query: TOP-1
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	



Step 2 – HJ (Hierarchical Join) Phase

- Disseminate τ to all nodes
- Each node sends back everything with score above all objectIDs in τ .
- Before sending the objects, each node tags as **incomplete**, scores that could not be computed exactly (upper bound)



v1	v2	v3	v4	v5	HJ
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	<div>o3, 405</div> <div>o1, 363</div> <div>o4', 354</div> <div>Complete</div> <div>Incomplete</div>
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	



Step 3 – CL (Cleanup) Phase

Have we found K objects with a complete score?

Yes: The answer has been found!

No: Find the *complete score* for each incomplete object (all in a single batch phase)

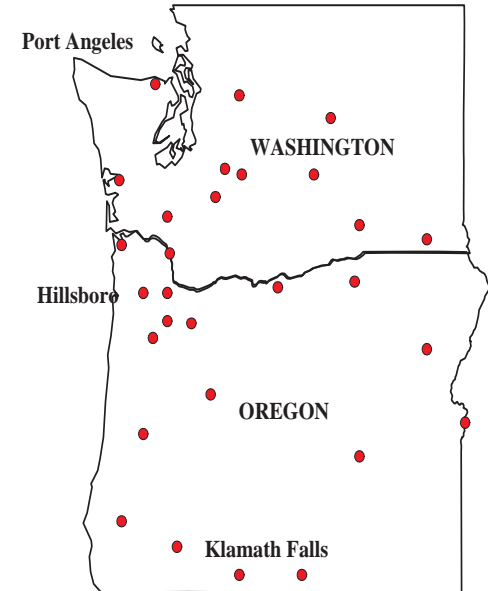
- **CL ensures correctness!**
- **This phase is rarely required in practice.**

v1	v2	v3	v4	v5	TOP-5
o3, 99	o1, 91	o1, 92	o3, 74	o3, 67	o3, 405
o1, 66	o3, 90	o3, 75	o1, 56	o4, 67	o1, 363
o0, 63	o0, 61	o4, 70	o2, 56	o1, 58	o4, 207
o2, 48	o4, 07	o2, 16	o0, 28	o2, 54	o0, 188
o4, 44	o2, 01	o0, 01	o4, 19	o0, 35	o2, 175



Experimental Evaluation

- We implemented a real P2P middleware in JAVA (sockets + binary transfer protocol).
- We tested our implementation with a network of 1000 real nodes using 75 Linux workstations.
- We use a trace driven experimentation methodology.



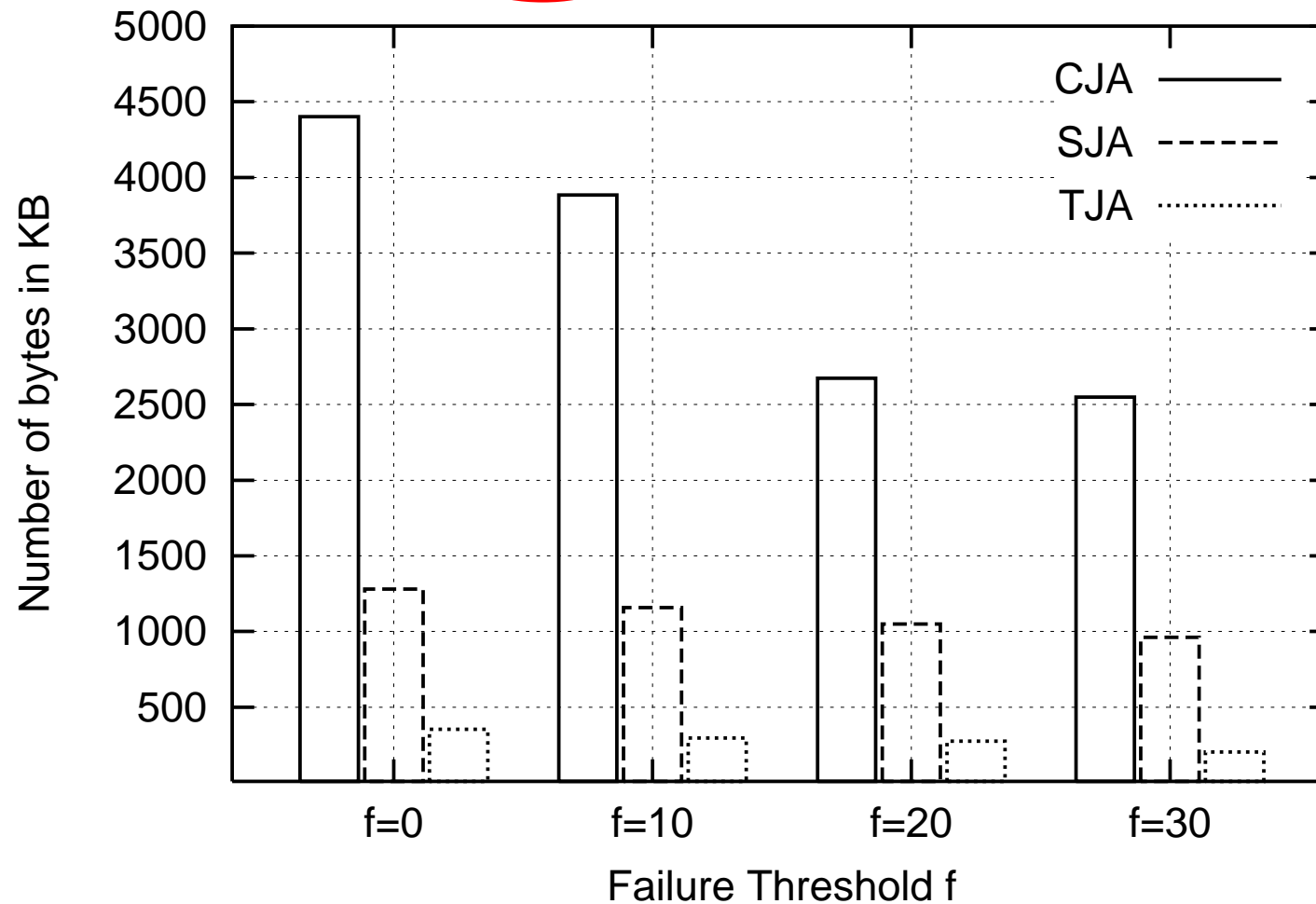
For the results presented in this talk:

- **Dataset:** Environmental Measurements from atmospheric monitoring stations in Washington & Oregon. (2003-2004)
- **Query:** Find the K timestamps on which the average temperature across all stations was maximum.
- **Network:** Random Graph (degree=4, diameter 10)
- **Evaluation Criteria:** i) **Bytes**, ii) **Time**, iii) **Messages**



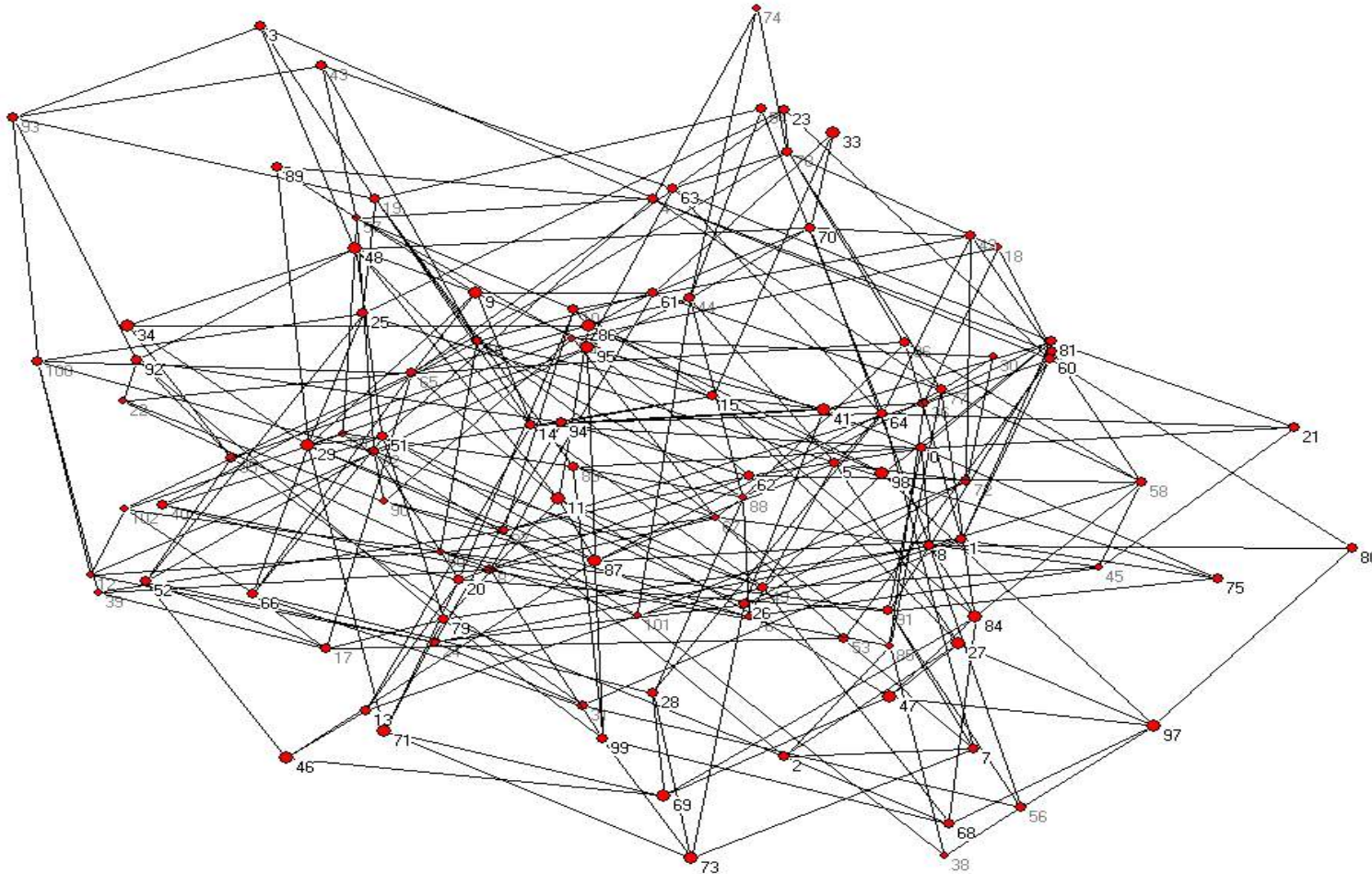
Experimental Results

Bytes Using the Atmon Dataset



TJA requires one order of magnitude less bytes than CJAs!

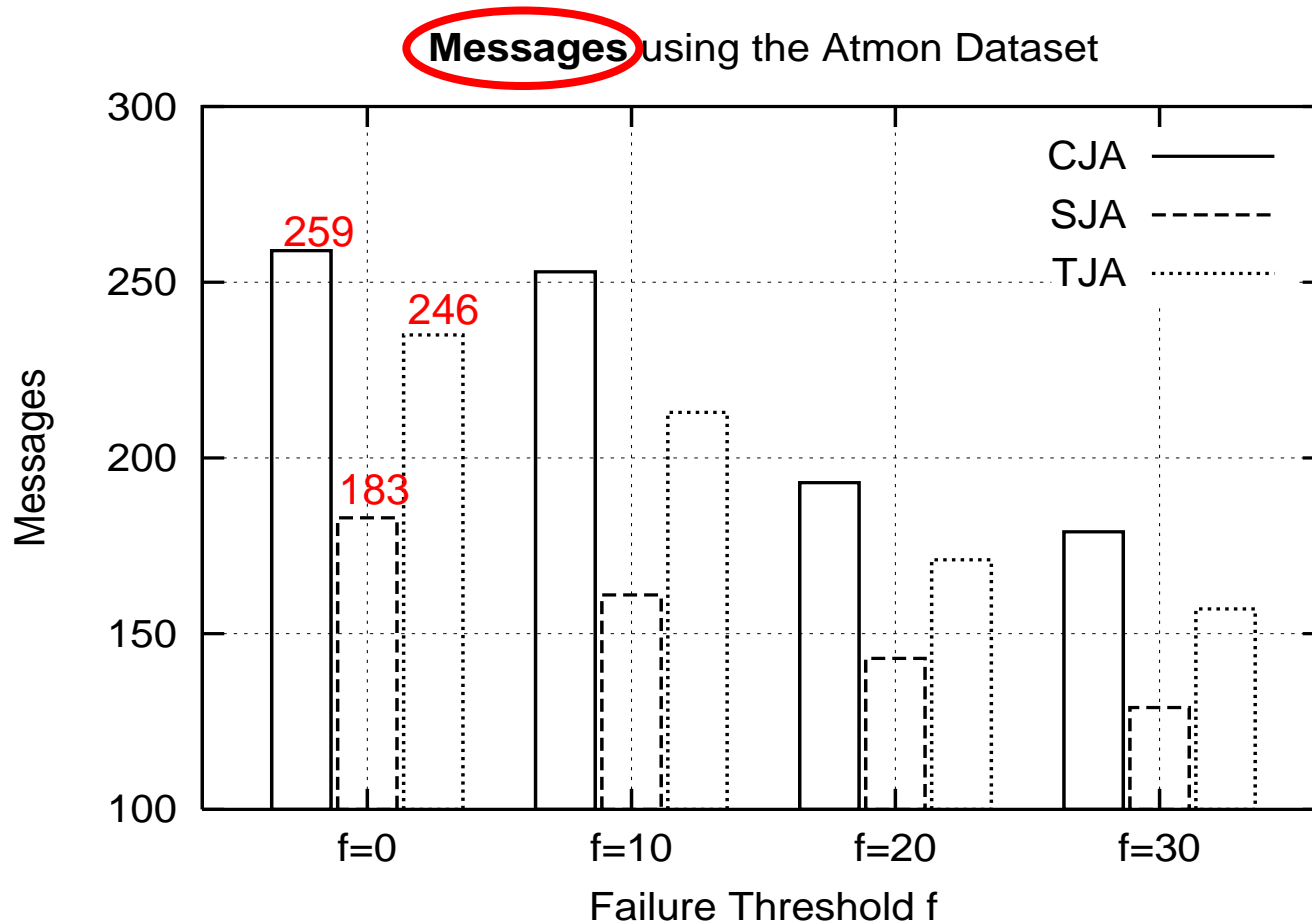
Experimental Results



TJA: 3.7sec [LB:1.0sec, HJ:2.7sec, CL:0.08sec]

SJA: 8.2sec CJA:18.6sec

Experimental Results



Although TJA consumes more messages than SJA these are small-size messages

The TPUT Algorithm



v1	v2	v3	v4	v5	TOP-1
o3, 99 o1, 66 o0, 63 o2, 48 o4, 44	o1, 91 o3, 90 o0, 61 o4, 07 o2, 01	o1, 92 o3, 75 o4, 70 o2, 16 o0, 01	o3, 74 o1, 56 o2, 56 o0, 28 o4, 19	o3, 67 o4, 67 o1, 58 o2, 54 o0, 35	o1=183 o3=405 o5=240 o1=363 o2'=158 o4'=137 o0'=124

Q: TOP-1 - - - P1 **— —** P2 P3

Phase 1 : $o1 = 91+92 = 183$, $o3 = 99+67+74 = 240$

$\tau = (Kth \text{ highest score (partial)} / n) \Rightarrow 240 / 5 \Rightarrow \tau = 48$

Phase 2 : Have we computed K exact scores ?

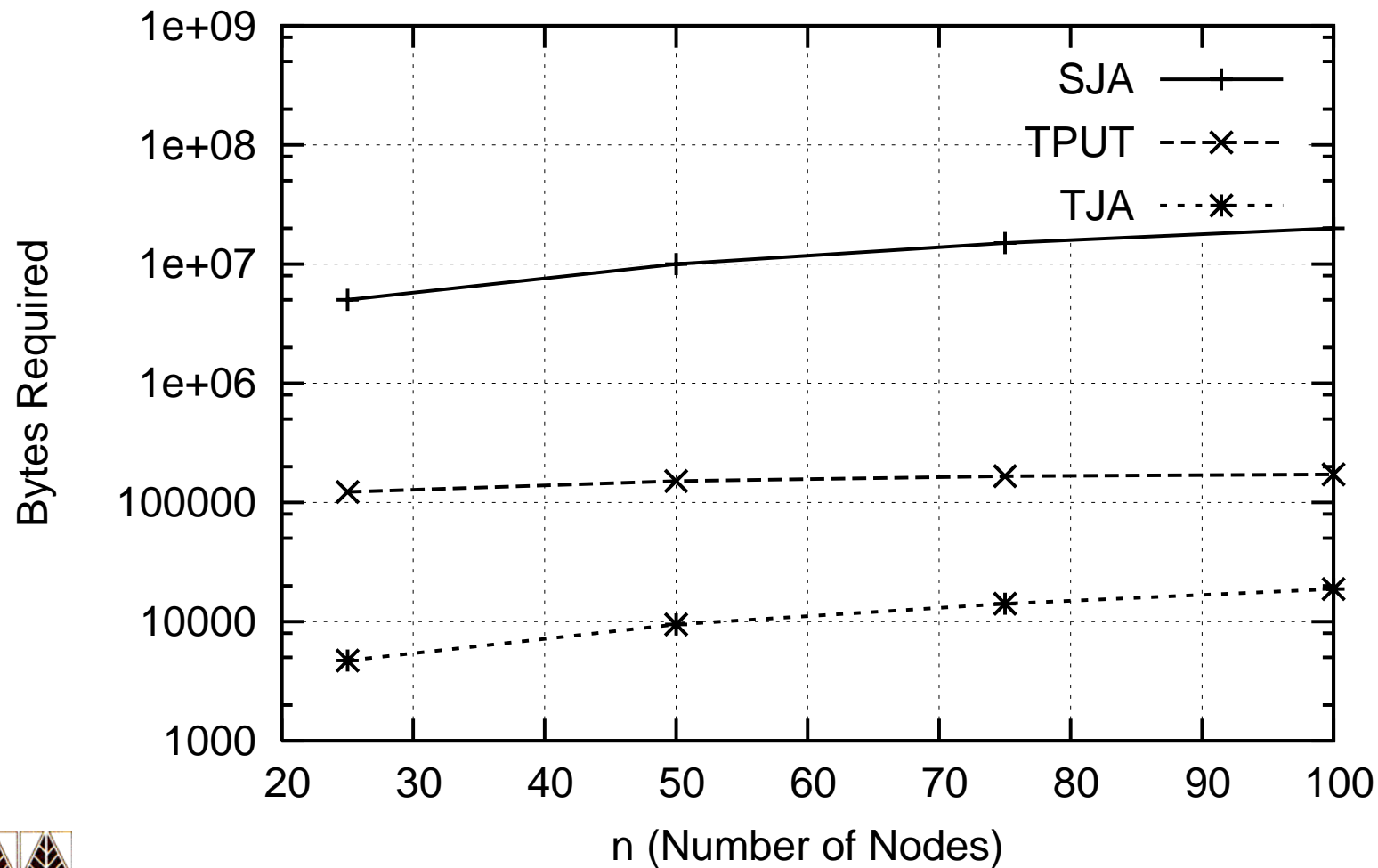
Computed Exactly: [**o3**, o1] Incompletely Computed: [o4, o2, o0]



Drawback: The threshold is uniform (too coarse)

TJA vs. TPUT

Bytes Required for Distributed Top-K Algorithms
(Star Topology, K=5, m=25K)



Presentation Outline

A. Top-K Algorithms: Definitions

B. Centralized Top-K Query Processing

- The Threshold Algorithm (TA)

C. Distributed Top-K Query Processing

- The Threshold Join Algorithm (TJA)
- Experimentation using 75 workstations

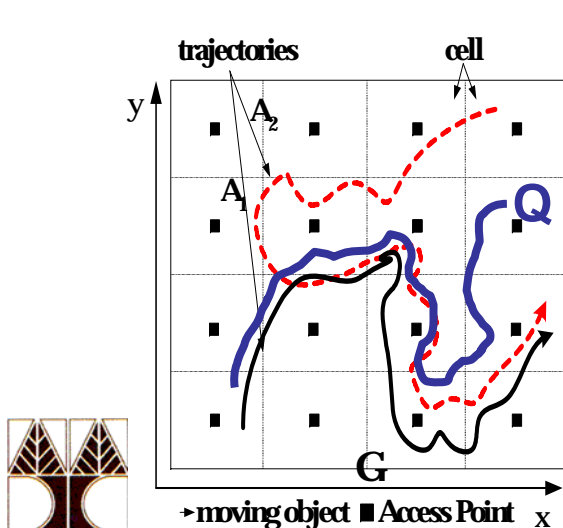
D. Other Applications of Top-K Queries

- Distributed Spatio-temporal Trajectory Retrieval (UB-K and UBLB-K Algorithms)
- In-Network Top-K Views (MINT Views)



Application 2: Spatiotemporal Query Processing

- **"Distributed Spatio-Temporal Similarity Search"** by *D. Zeinalipour-Yazti, S. Lin, D. Gunopulos, ACM 15th Conference on Information and Knowledge Management, (ACM CIKM 2006), November 6-11, Arlington, VA, USA, pp.14-23, August 2006.*
- **Similarity Search:** Given a query Q , find the degree of similarity (Euclidean distance, DTW, LCSS) between Q and a set of m trajectories $\{A_1, A_2, \dots, A_m\}$.



- Each A_i ($i \leq m$) is segmented into a number of non-overlapping cells $\{C_1, C_2, \dots, C_n\}$ that maintain the local subsequences.
- **Challenge:** How can we find the K most similar trajectories to Q without pulling together all subsequences

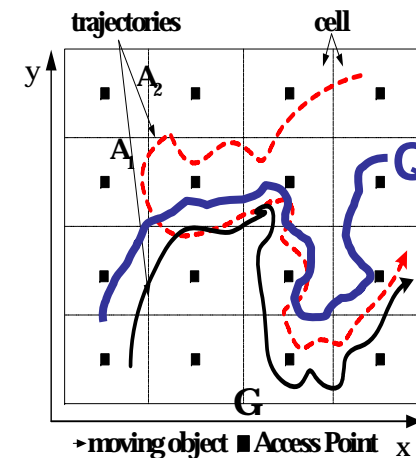
Application 2: Spatiotemporal Query Processing

Solution Outline

- Each cell computes a lower bound and an upper bound on the distance of Q to its local subsequences.
- The distributed **scoring table** now contains **score bounds (lower,upper) rather than exact scores.**

	v1	v2	v3	METADATA
	id,lb,ub	id,lb,ub	id,lb,ub	id,lb,ub
m	A2,3,6	A4,4,5	A4,1,3	A4,10,18
	A0,4,8	A2,5,6	A0,6,10	A2,13,19
	A4,5,10	A0,5,7	A2,5,7	A0,15,25
	A7,7,9	A3,5,6	A9,6,7	A3,20,27
	A3,8,11	A9,8,10	A3,7,10	A9,22,26
	A9,8,9	A7,12,13	A7,11,13	A7,30,35

n				



- We have proposed two iterative algorithms: **UB-K** and **UBLB-K**, which combine these score bounds.

- UB-K and UBLB-K find the K most similar trajectories to Q without pulling together the distributed subsequences.²⁹



Application 3: MINT Views

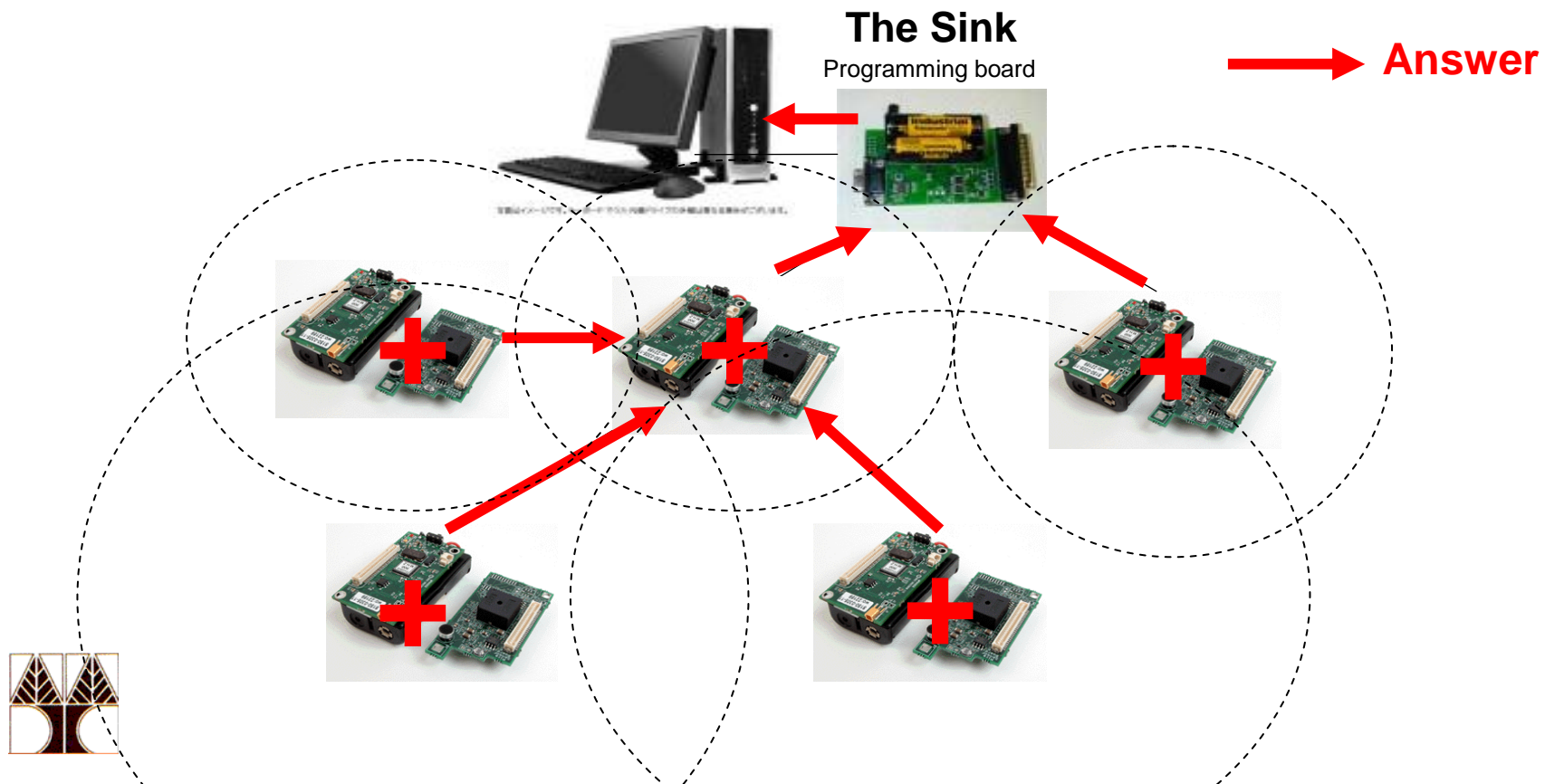
- **"MINT Views: Materialized In-Network Top-k Views in Sensor Networks"**
D. Zeinalipour-Yazti, P. Andreou, P. Chrysanthis and G. Samaras, In IEEE 8th International Conference on Mobile Data Management, Mannheim, Germany, May 7 - 11, accepted, 2007
- **Views (in databases)** are virtual tables that contain the results from an arbitrary query.
- ... therefore they speedup query execution.
- **MINT Views:** a novel framework for optimizing the execution of continuous monitoring queries in sensor networks.



Application 3: MINT Views

A sensor network at a glance

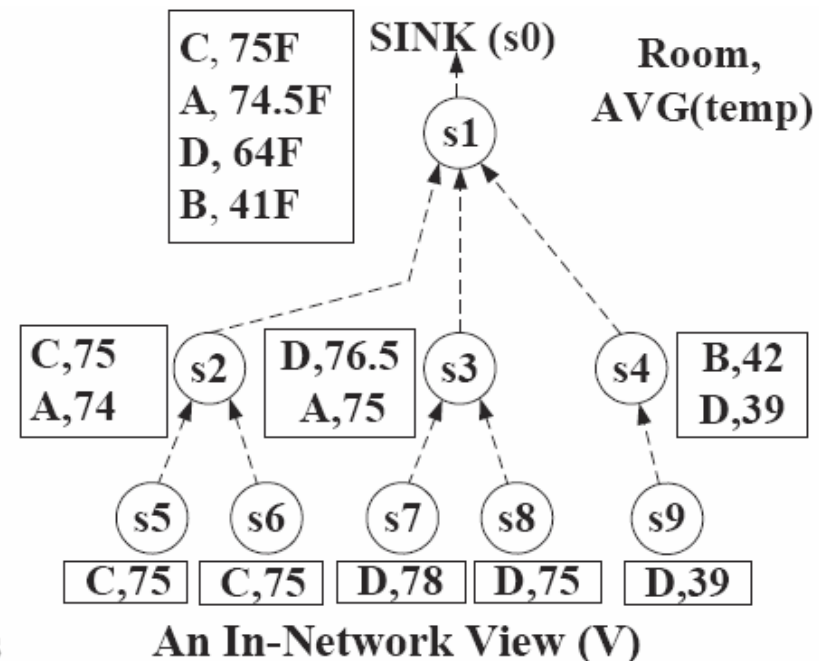
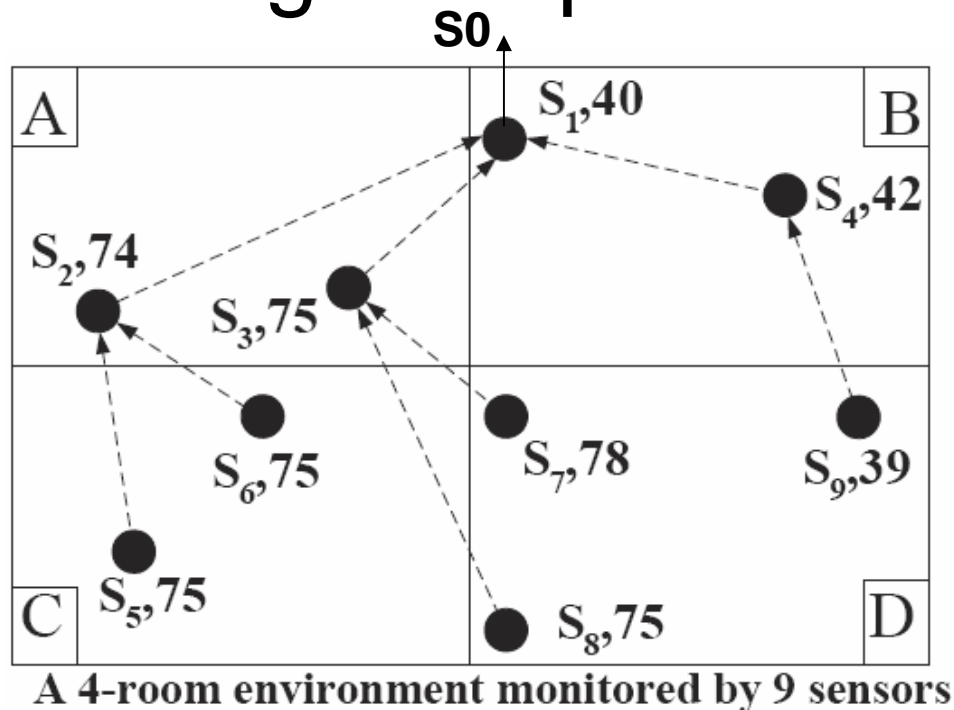
While parameters are **sensed** from the physical environment, these are **aggregated** (with the results of the children) and are then **transferred** towards the sink for storage and analysis



MINT Views: Example

Example: Four rooms {A,B,C,D}, 9 sensors {s1,...,s9}

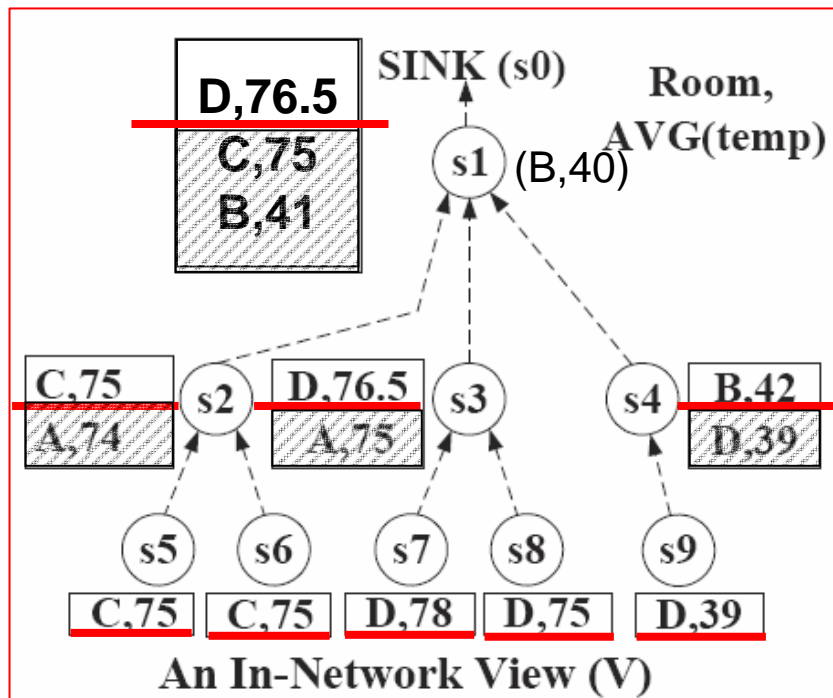
Query: Find the room with the highest average temperature (**TOP-1** result)



MINT Views: Example

Assume that we only need the $K=1$ highest-ranked answers, rather than all of them.

Naïve Solution: Each node eliminates any tuple with a score lower than its top-1 result.



Problem:

We received an incorrect answer i.e. $(D, 76.5)$ instead of $(C, 75)$.



MINT Views: Top-K Pruning Concept

- **Objective**

Find the **correct** Top-K answer at the sink.

- **Problem:**

If a node X prunes object O then X's parent might need O. **What should we prune-away?**

- **Solution:**

- To determine which objects will be needed at the higher-levels of the hierarchy by **bounding** them with their maximum possible value.

- Then pruning becomes straightforward!.

- **We can guarantee that the pruned objects will not be among the K highest ranked answers at the sink (therefore we always find the correct answer)!**



MINT Views: Example

- X is an arbitrary node in the tree hierarchy.
- X maintains a list of (room,sum) objects.
- X knows some meta-information about the network, e.g.,
 - $y1 = \text{«max possible temperature»} = 120$, and
 - $y2 = \text{«sensors in each room»} = 5$.
- X now bounds the final value of every object it has locally:
 $\text{sum}' = \text{sum} + (y2 - \text{count}) * y1$
- sum' is an upper bound of sum (maximum possible value for sum at the sink).

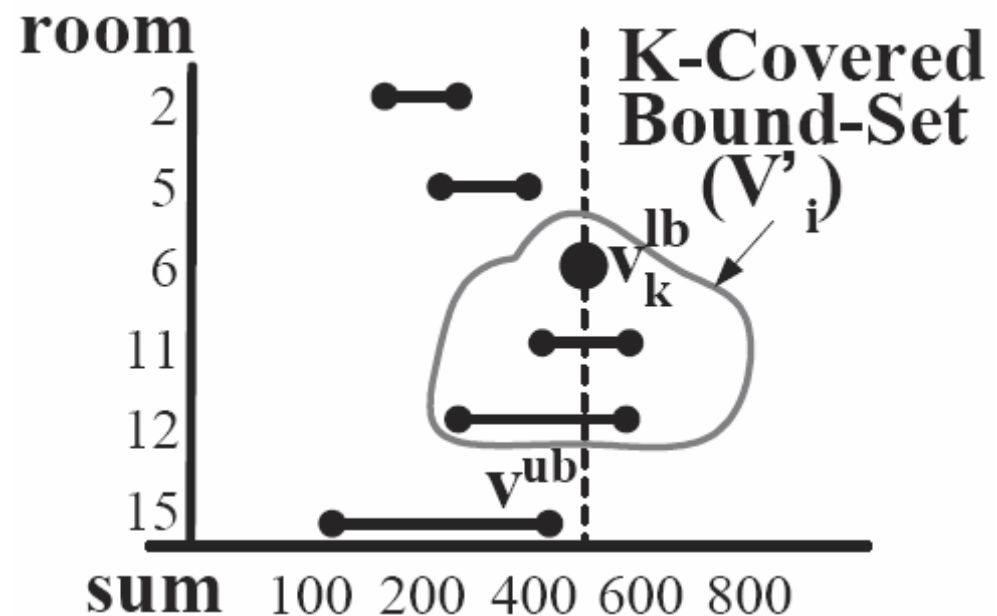
room	sum	count	sum'
2	200	4	320
5	270	4	390
6	500	5	500
11	460	4	580
12	290	3	530
15	130	2	490



MINT Views: Example

- We can now locally rank these ranges and prune-away any object outside the K-covered-bound set.
- K-covered Bound-set** : Includes all the objects which have an **upper bound** (v^{ub}) greater or equal to the **kth highest lower bound** (v_k^{lb}), **i.e., $v^{ub} \geq v_k^{lb}$**

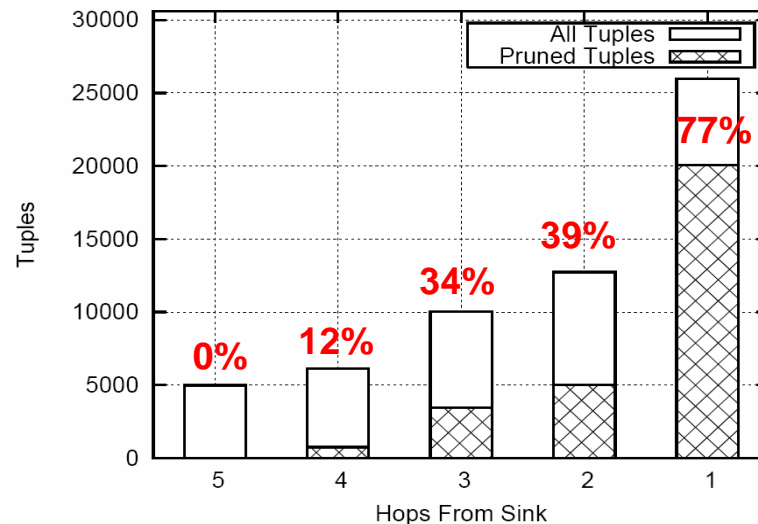
room	sum	sum'
2	200	320
5	270	390
6	500	500
11	460	580
12	290	530
15	130	490



MINT Views: Experimentation

- We obtained a real trace of atmospheric data collected by UC-Berkeley on the Great Duck Island (Maine) in 2002.
- We then performed a trace-driven experimentation using XBows TELOSB sensor.
- Our query was as follows:
 - SELECT TOP-K area, Avg(temp)
 - FROM sensors
 - GROUP BY area

AtmoMon Dataset - MINT Pruning Per Tree Depth
(Random Graph, $n=32$, network=250Kbps)



Conclusions

- I have presented the **Concepts** behind popular Top-k query processing algorithms and an array of **Applications** utilizing these algorithms.
- I have also presented, at a high level, a variety of **Algorithms** that we have developed in order to support this era of distributed databases.
- Top-K Query Processing is a new area with many **new challenges** and **opportunities**!
- We are working on applying this technology in new application areas, e.g.:

“FailRank: Towards a Unified Grid Failure Monitoring and Ranking System”, Demetrios Zeinalipour-Yazti, Kyriacos Neocleous, Chryssis Georgiou, Marios D. Dikaiakos, submitted for publication, 2007.



Top-K Algorithms: Concepts and Applications

by

Demetris Zeinalipour

Thank you!

This presentation is available at:

<http://www2.cs.ucy.ac.cy/~dzeina/talks.html>

Related Publications available at:



<http://www2.cs.ucy.ac.cy/~dzeina/publications.html>