Efficient TopK-Query Algorithm for Minerva

Christian Langner

Supervisors:
Sebastian Michel, Thomas Neumann

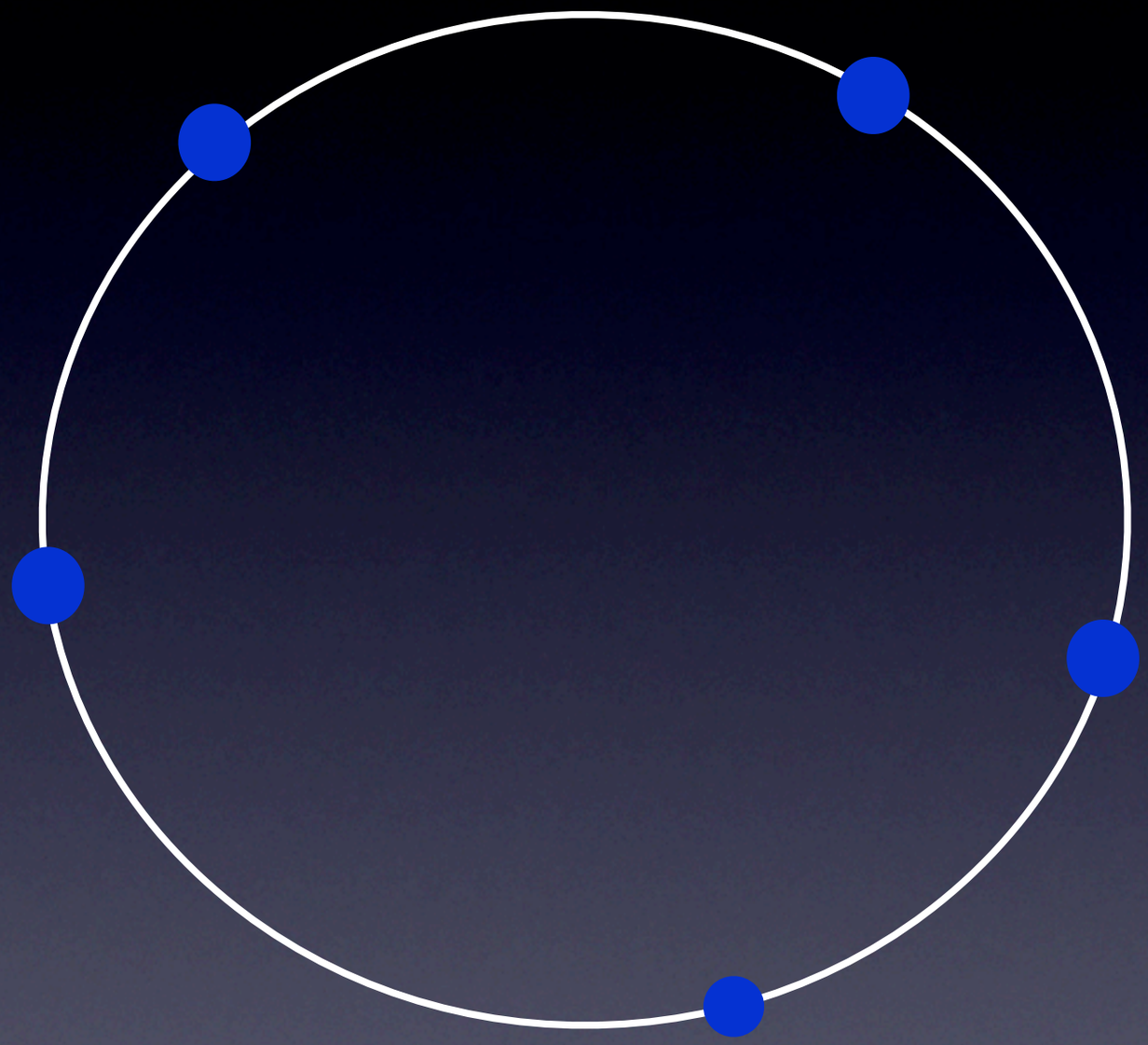
Oberseminar, 27.02.2007

Department 5: Databases and Information Systems
MPI Informatik, Saarbrücken, Germany

Motivation

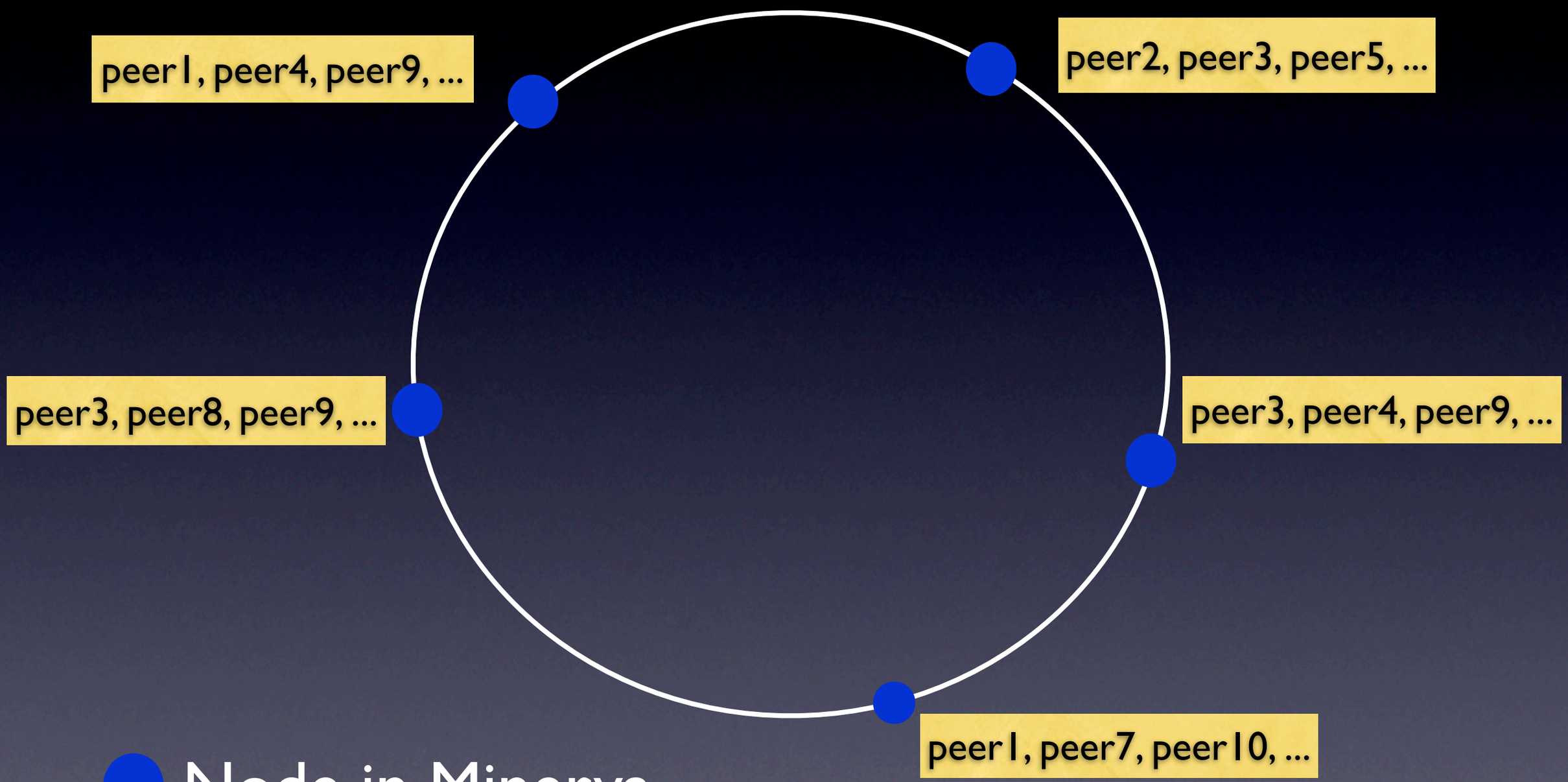
- New search technique: P2P Web Search
- Implementation of technique in Minerva
- Allows to search for multiple terms
- Problem of overlap in selection of peers
- Efficient selection of peers

Minerva Design

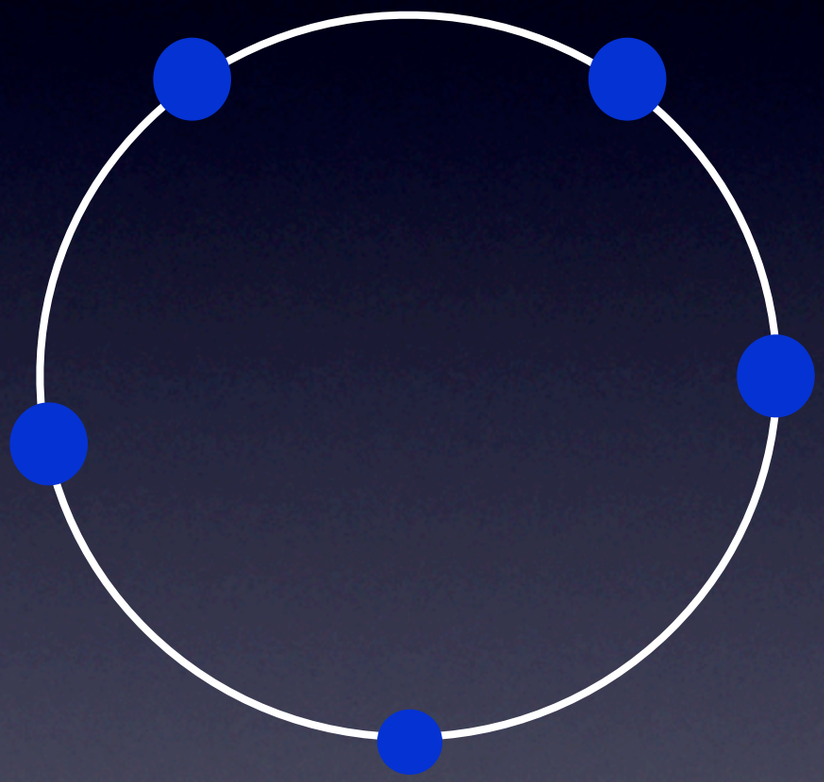


Node in Minerva

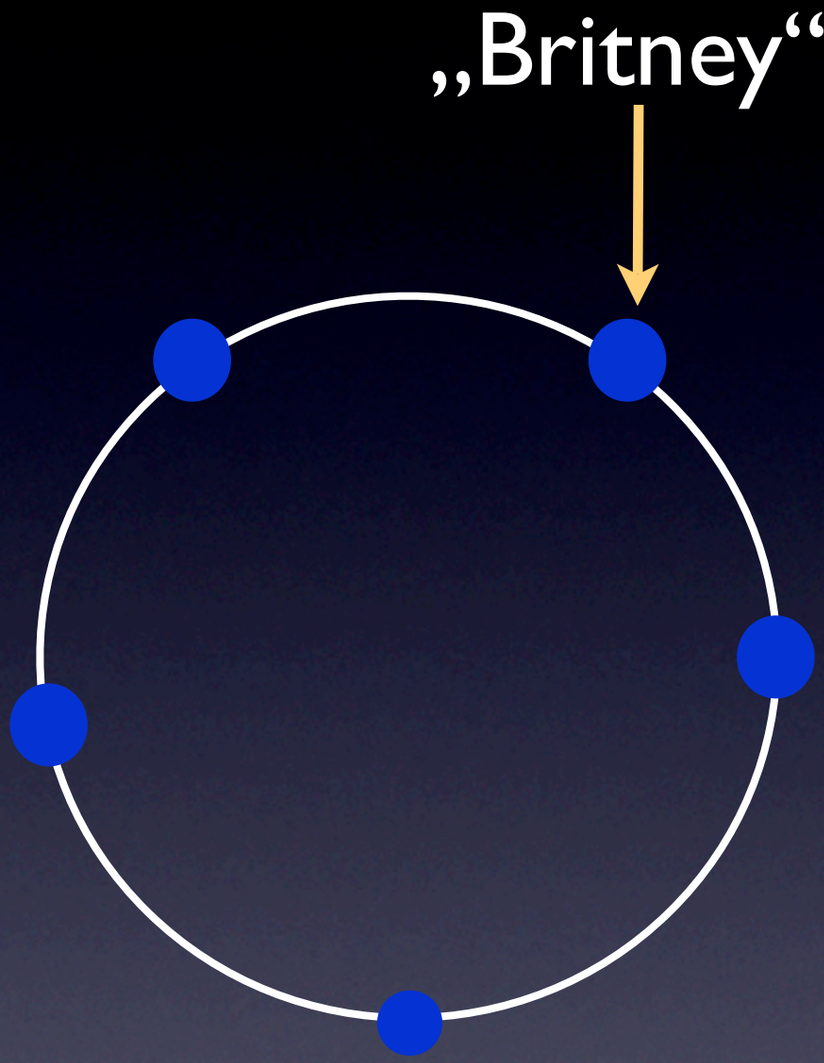
Minerva Design



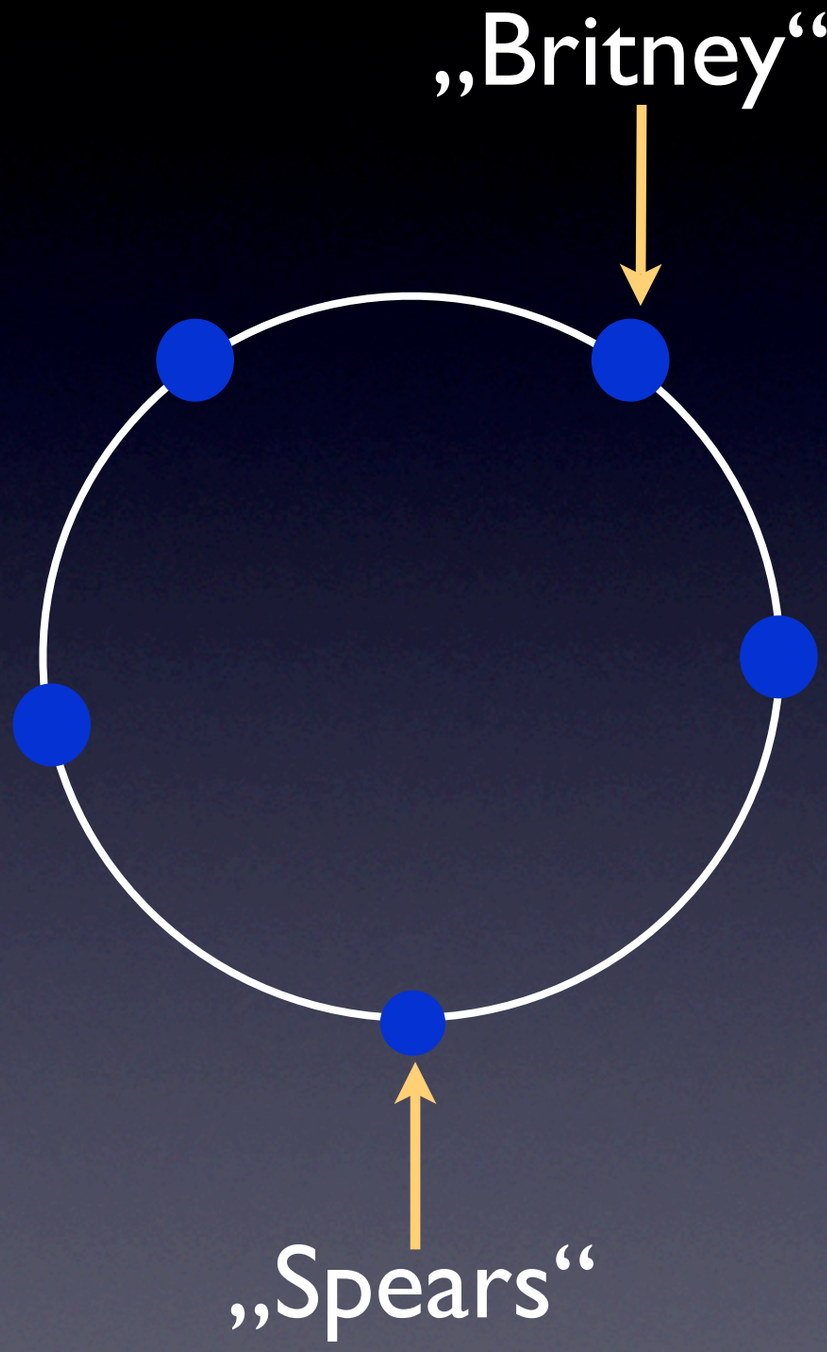
Problem ...



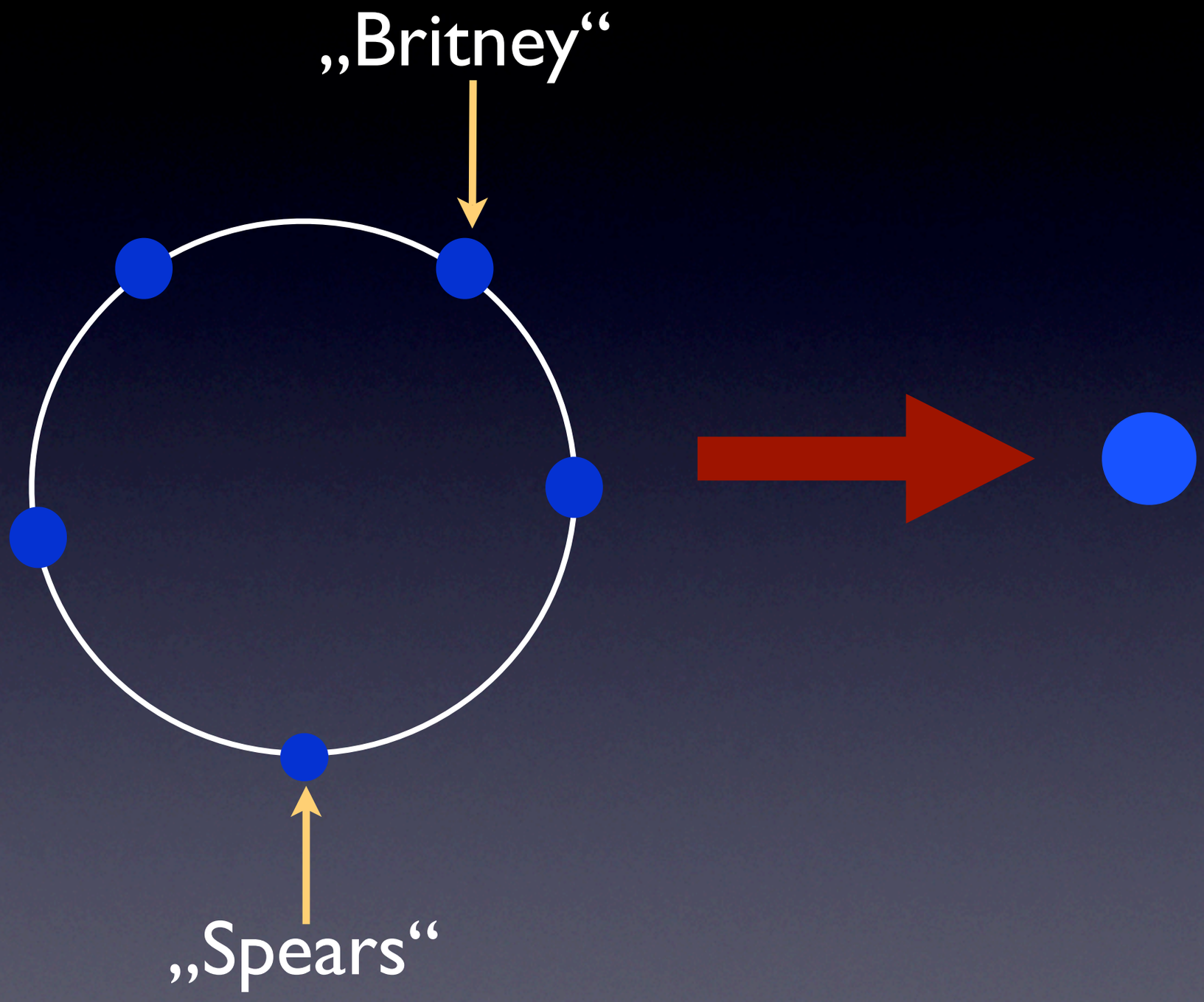
Problem ...



Problem ...



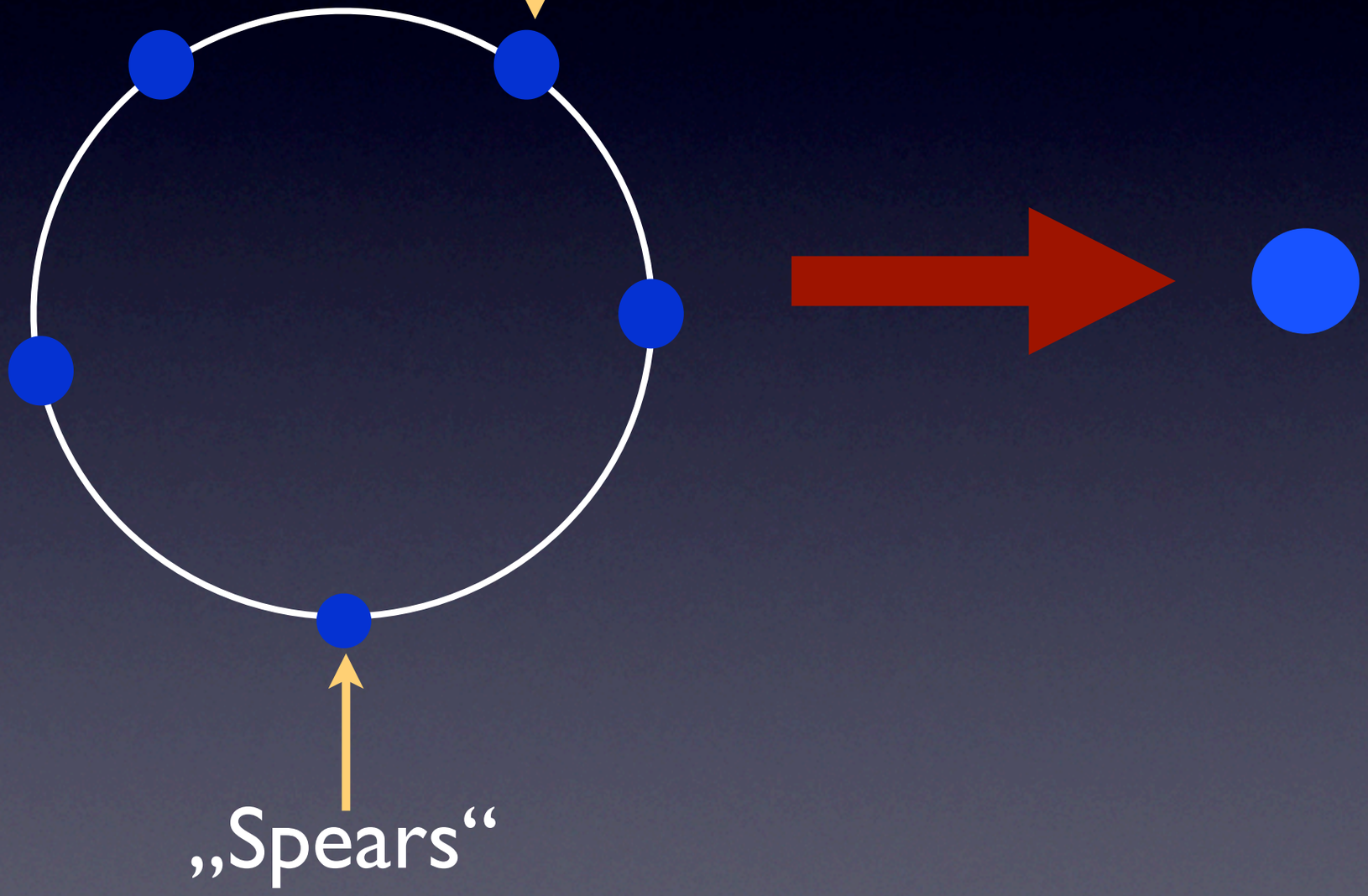
Problem ...



Problem ...

„Britney“

„Britney Spears“



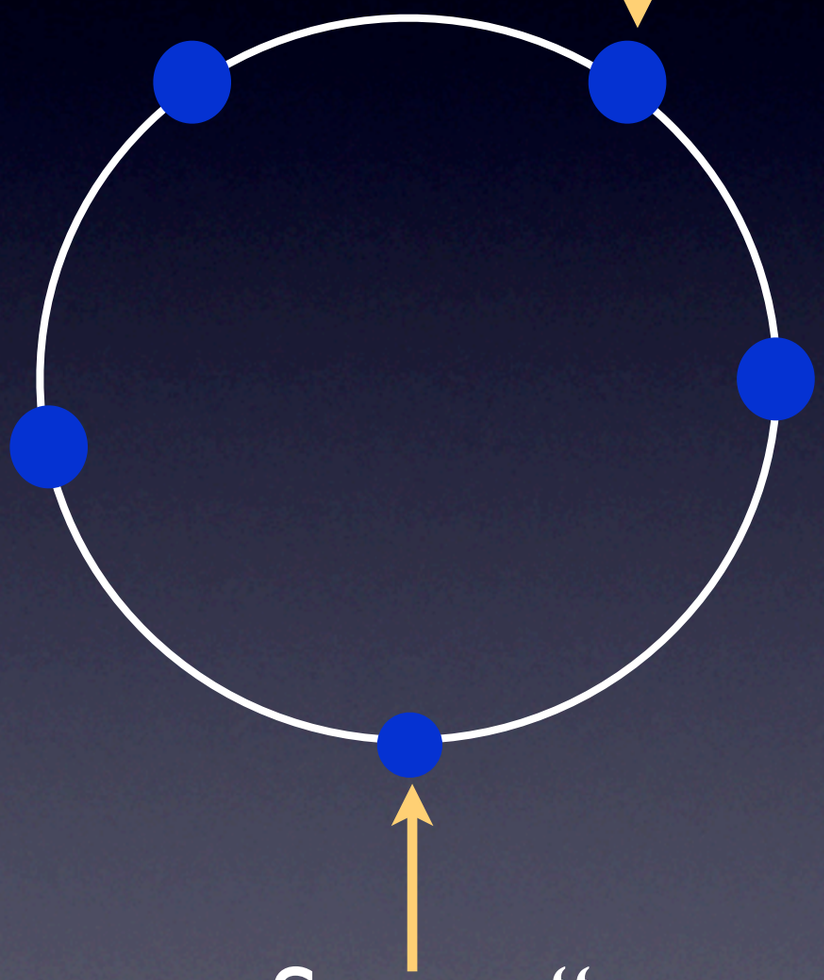
„Spears“

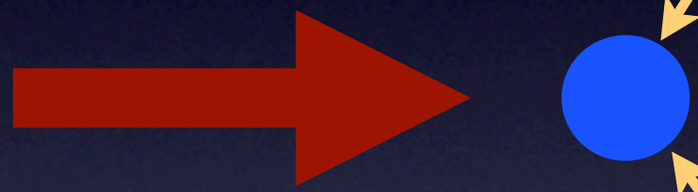
Problem ...

„Britney“

„Britney Spears“

peer2, peer3, peer5, ...

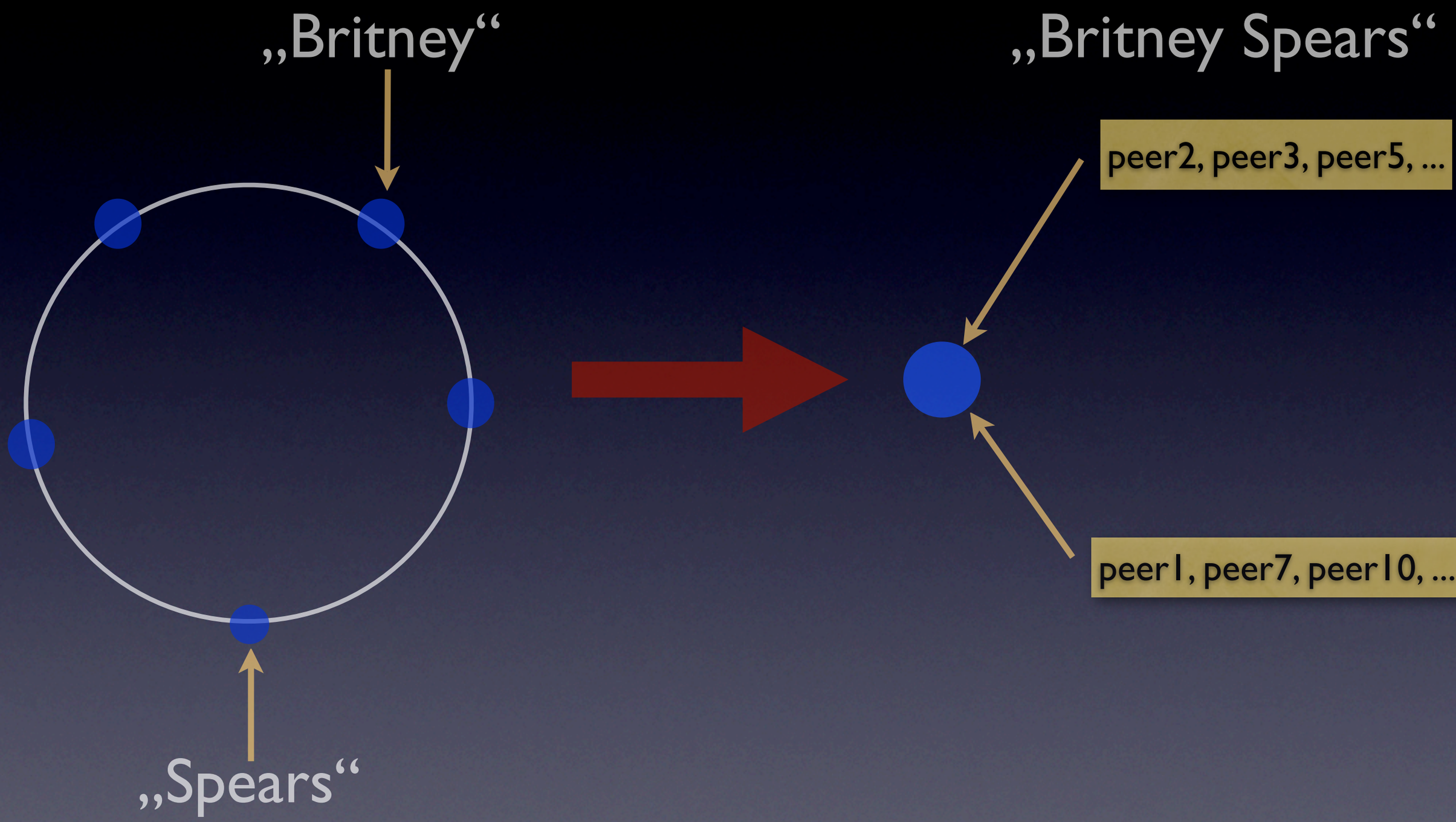




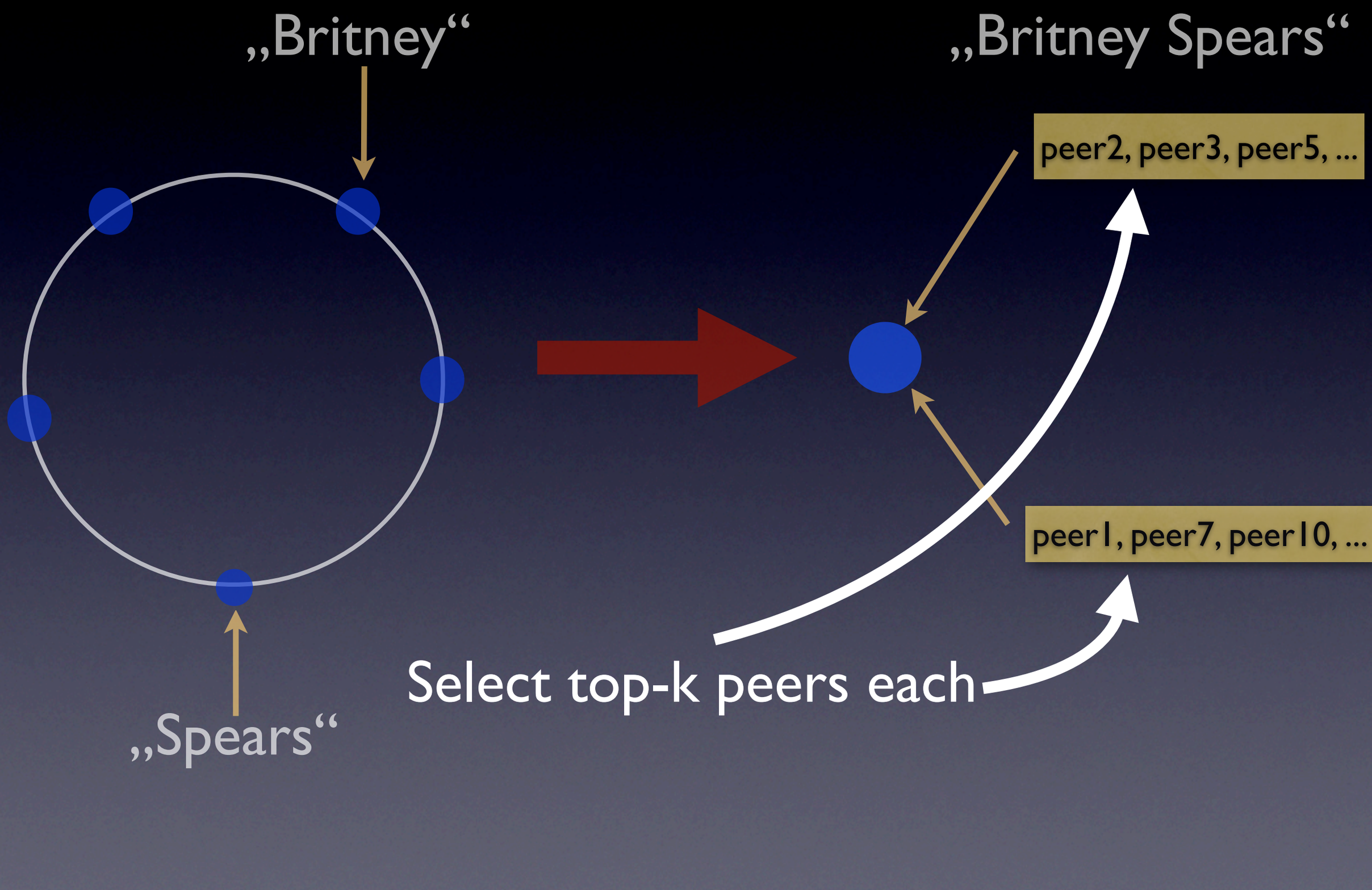
peer1, peer7, peer10, ...

„Spears“

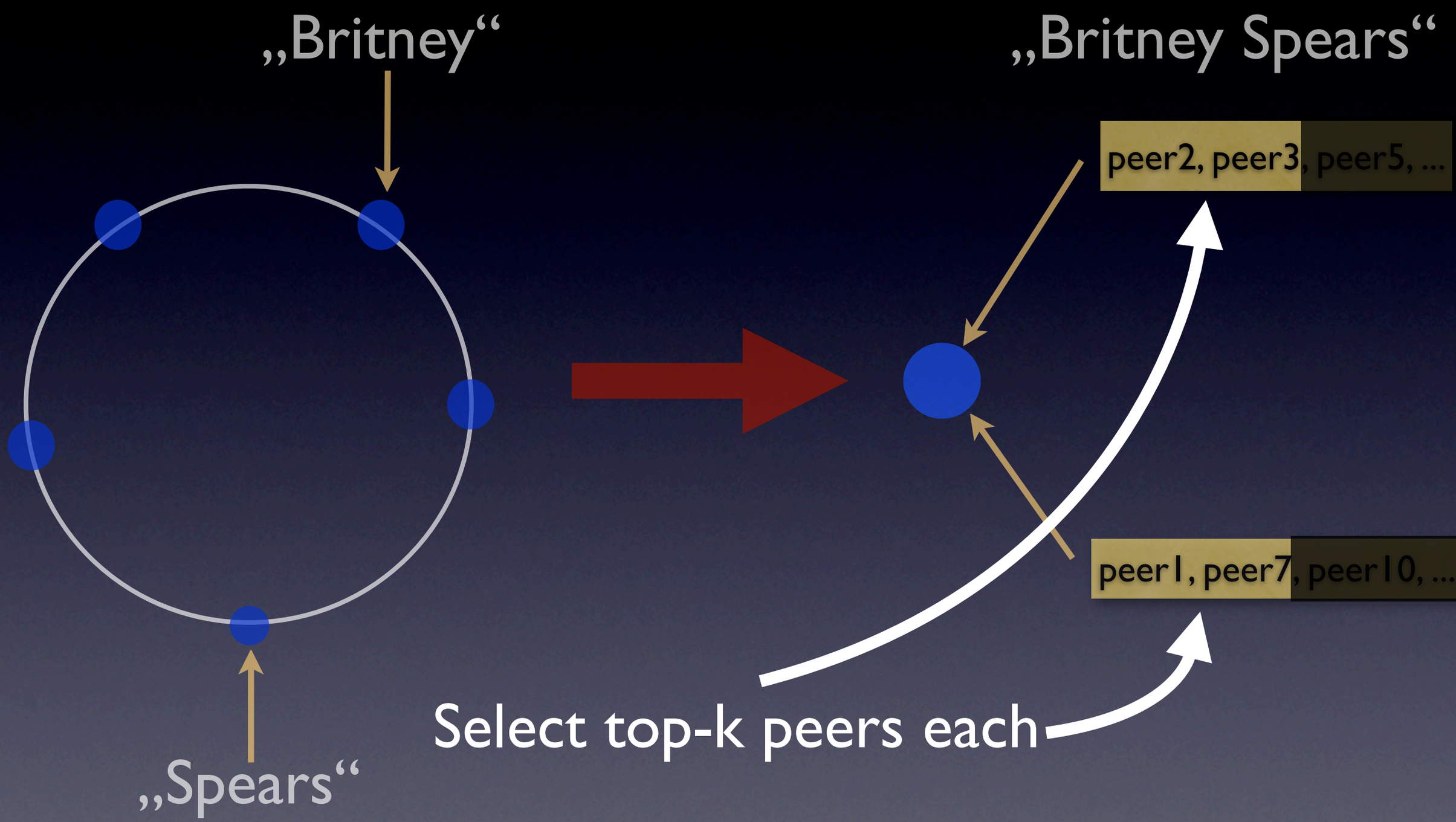
Problem ...


„Britney“
„Britney Spears“
peer2, peer3, peer5, ...
peer1, peer7, peer10, ...
„Spears“

Problem ...


„Britney“
„Britney Spears“
peer2, peer3, peer5, ...
peer1, peer7, peer10, ...
Select top-k peers each
„Spears“

Problem ...

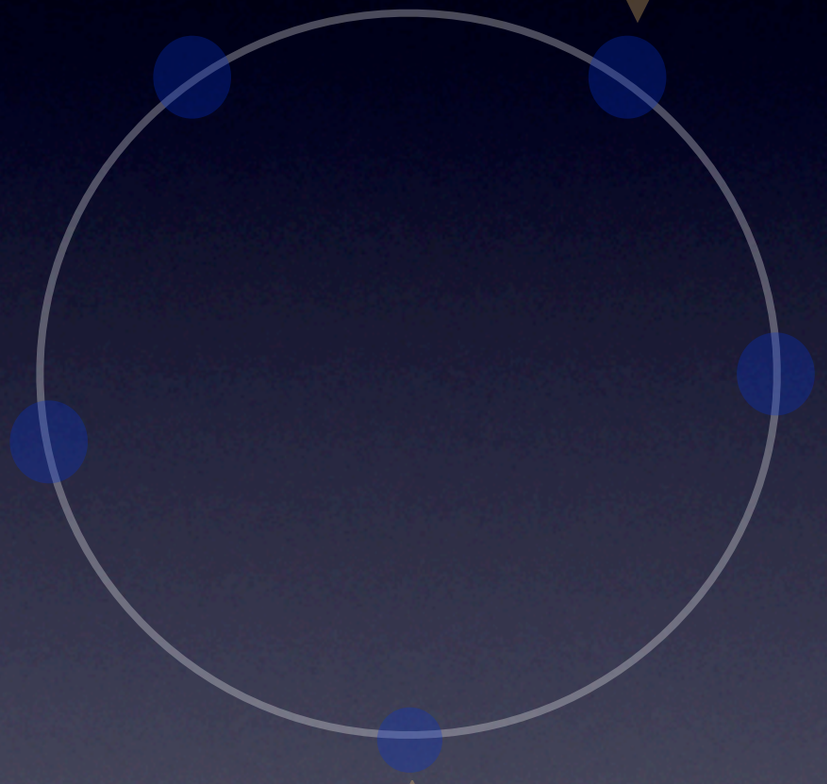

„Britney“
„Britney Spears“
peer2, peer3, peer5, ...
peer1, peer7, peer10, ...
Select top-k peers each
„Spears“

Problem ...

„Britney“

„Britney Spears“

peer2, peer3, peer5, ...



peer1, peer7, peer10, ...

„Spears“

Problem ...

„Britney“

„Britney Spears“

r3, peer5, ...

Fetching all peers from nodes is too expensive !!

7, peer10, ...

„Spears“

Problem ...

„Britney“

„Britney Spears“

r3, peer5, ...

7, peer10, ...

„Spears“

Idea:

Usage of Distributed Top-K algorithm (TPUT, Klee, Fagin)

TPUT

- Peerlists ranked in descending order
- Phase 1:
  - Select from all peerlists top-k peers
  - Compute aggregated value for every peer
  - k'th top value is τ_1 (phase-1 bottom)
- Phase 2:
  - Set threshold $T = (\tau_1 / m)$, m is number of nodes
  - Select k highest peers $\geq T$
  - k'th highest is τ_2 (phase-2 bottom, where $\tau_1 \leq \tau_2 \leq \tau$)
  - Peers $< \tau_2$ are eliminated, remaining ones in set S

TPUT

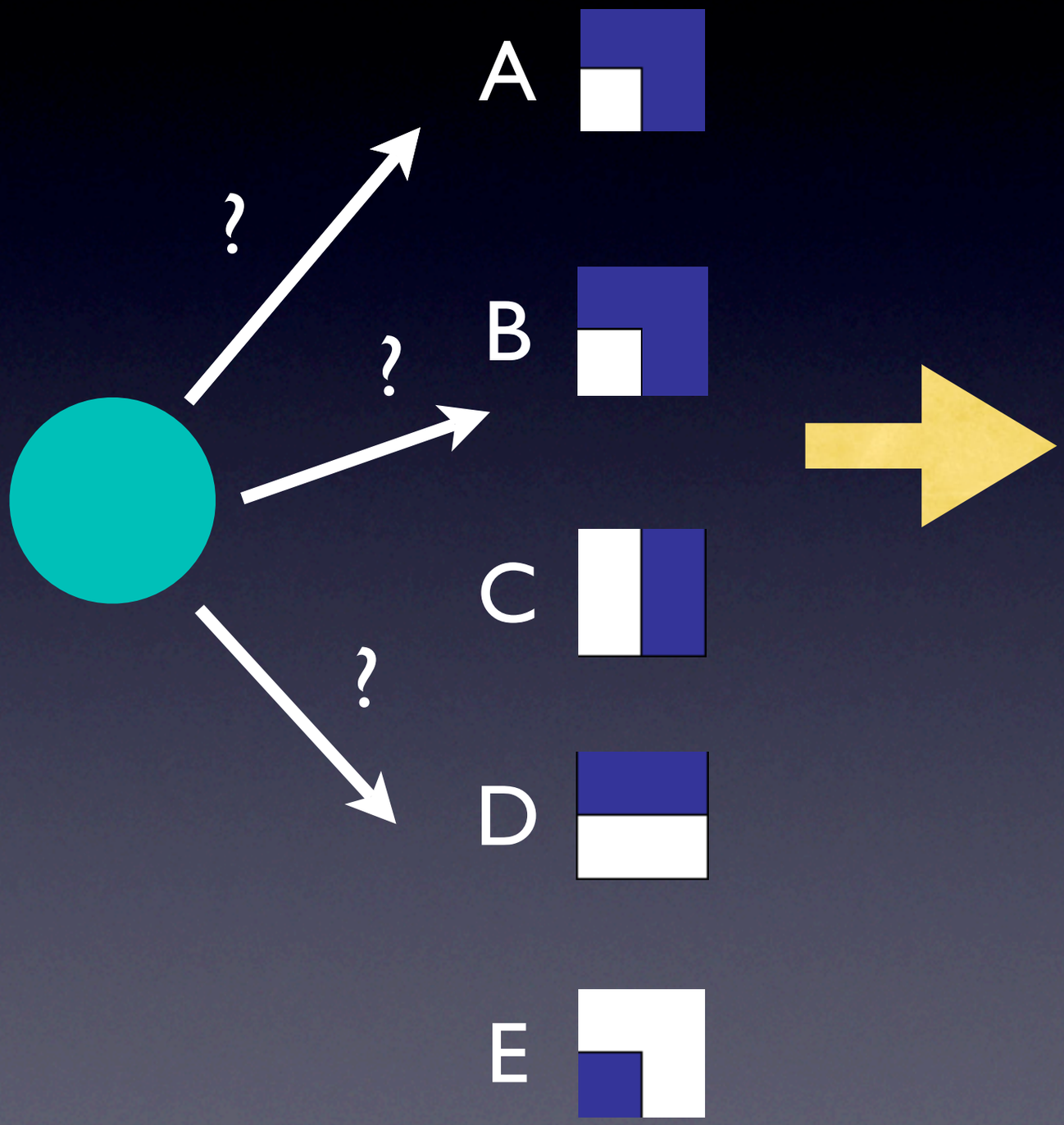
- Phase 3:
  - Set S sended to all nodes
  - Each node sends values of peers back
  - Calculate exact sum of peers in S
  - Choose top-k peers
  - ➡ True top-k peers

TPUT

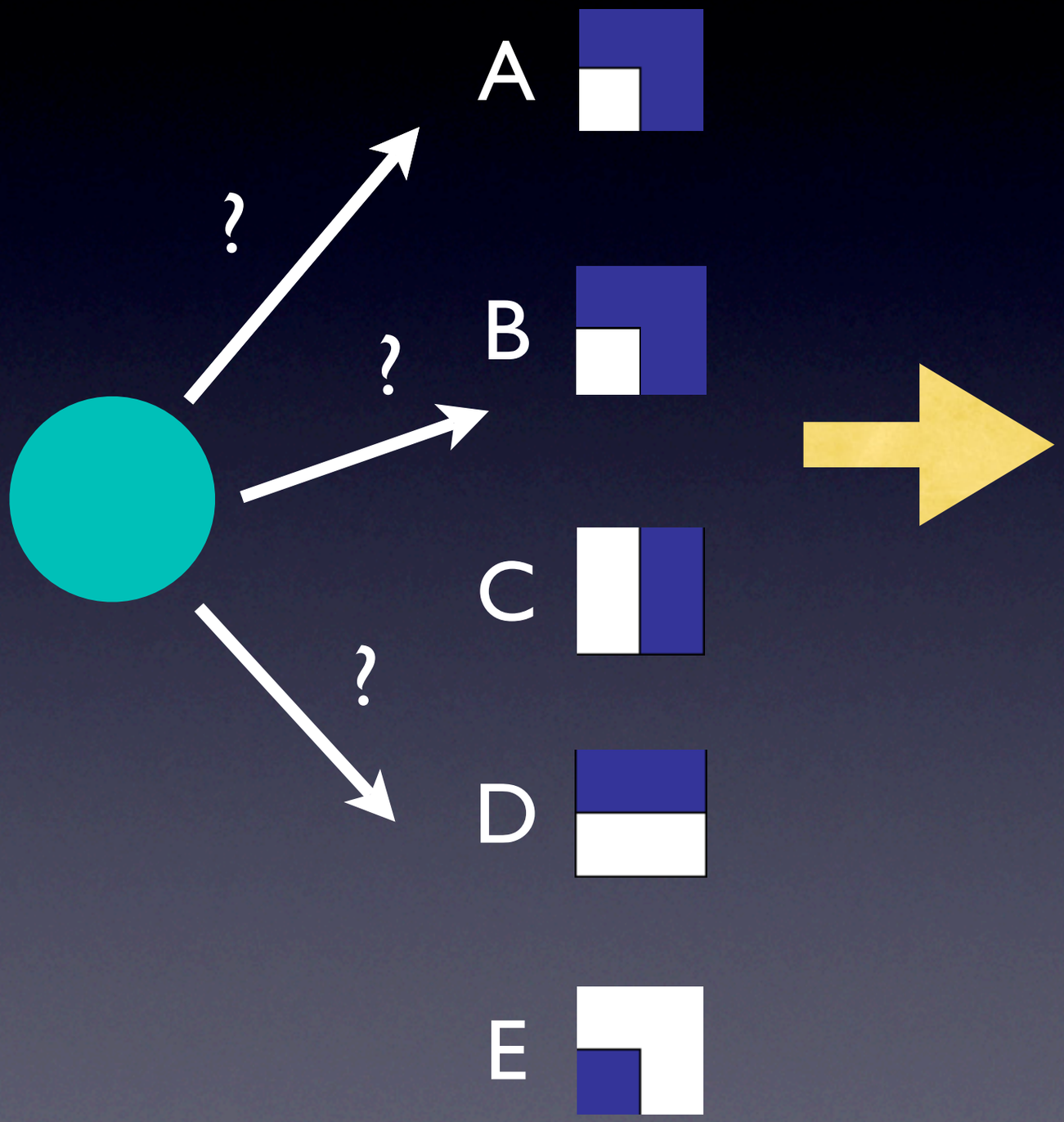
- Algorithm computes best k peers (for constant ranking /scores)
- Now: Adapt algorithm for Minerva peer selection
- Problem of Overlap Aware

➡ Goal: Analyze how peer selection in Minerva influences the result of Top-K

Overlap Awareness

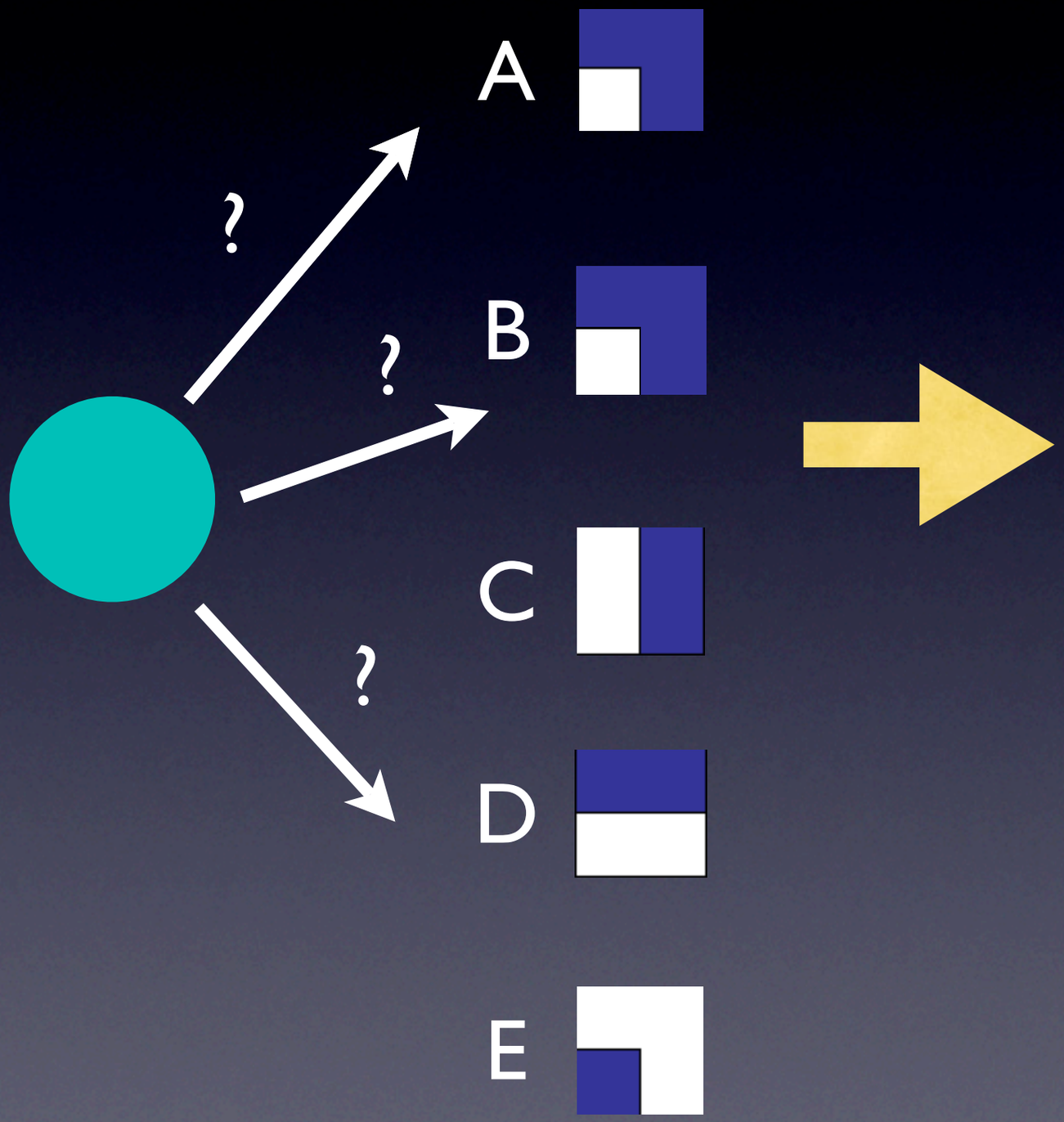


Overlap Awareness



Naive Strategy:

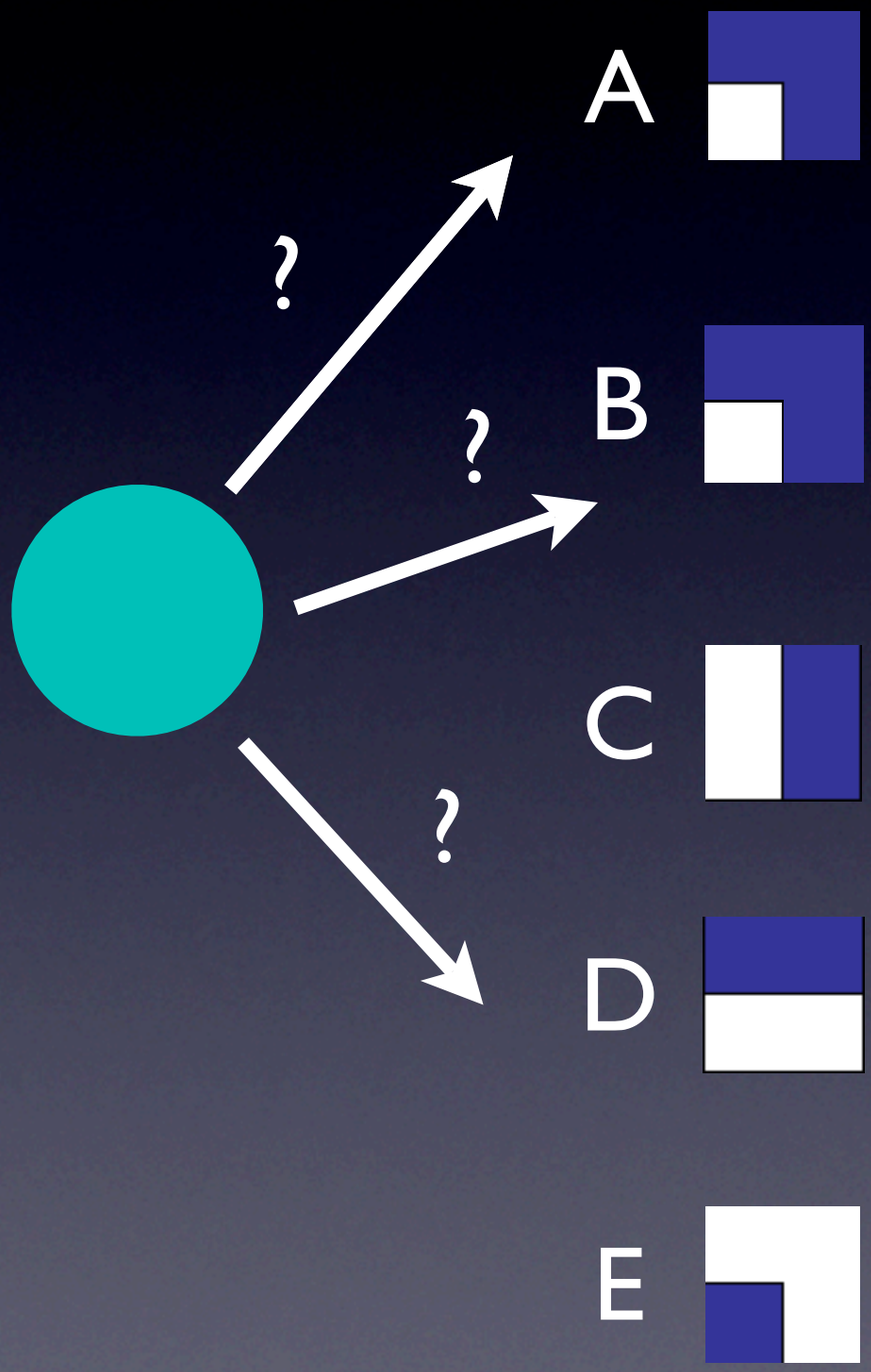
Overlap Awareness



Naive Strategy:

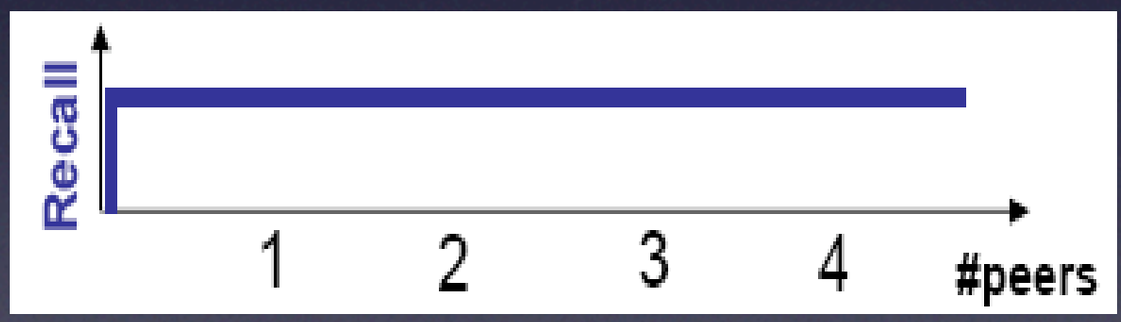
Select peers A to D
step by step

Overlap Awareness

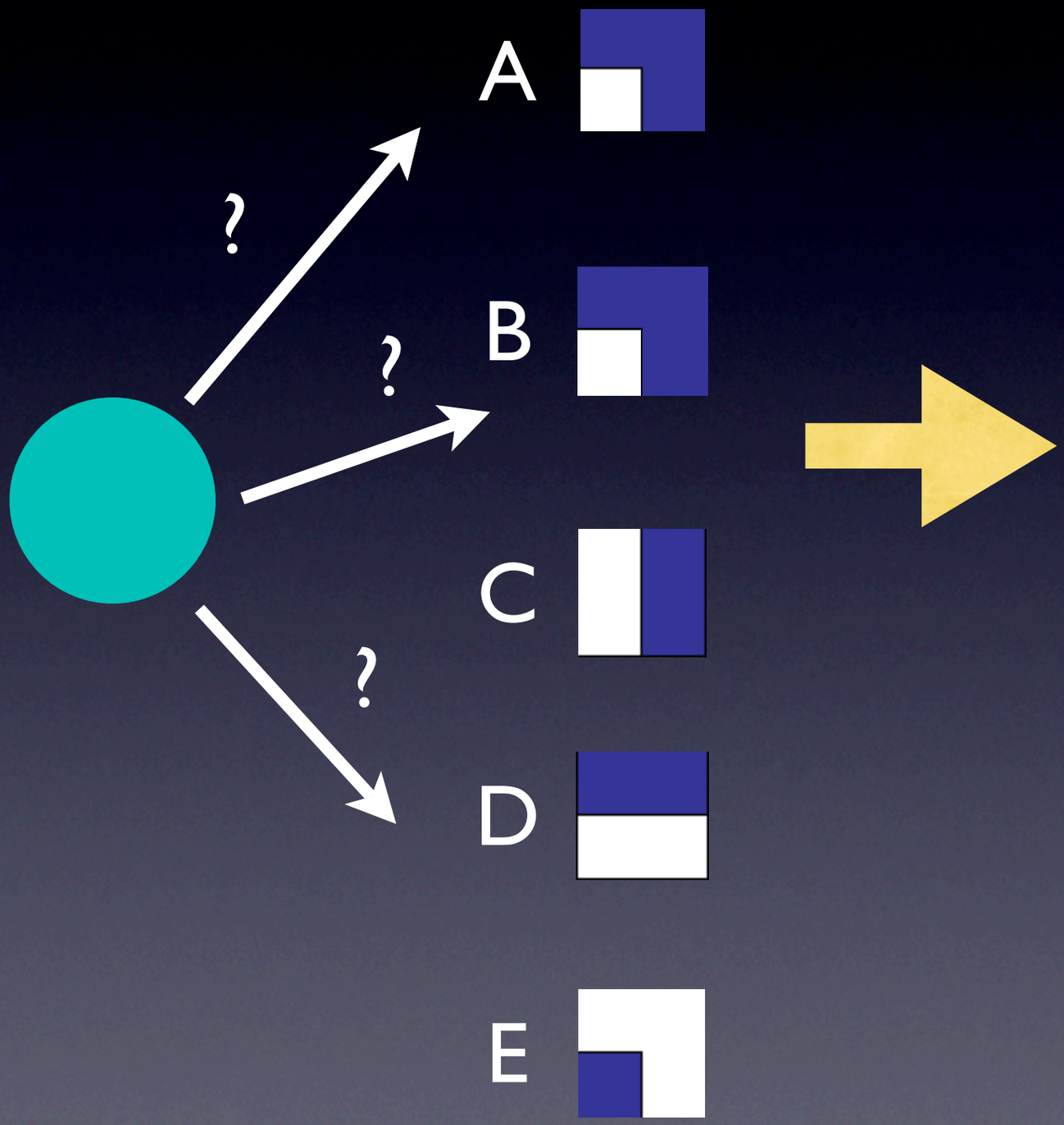


Naive Strategy:

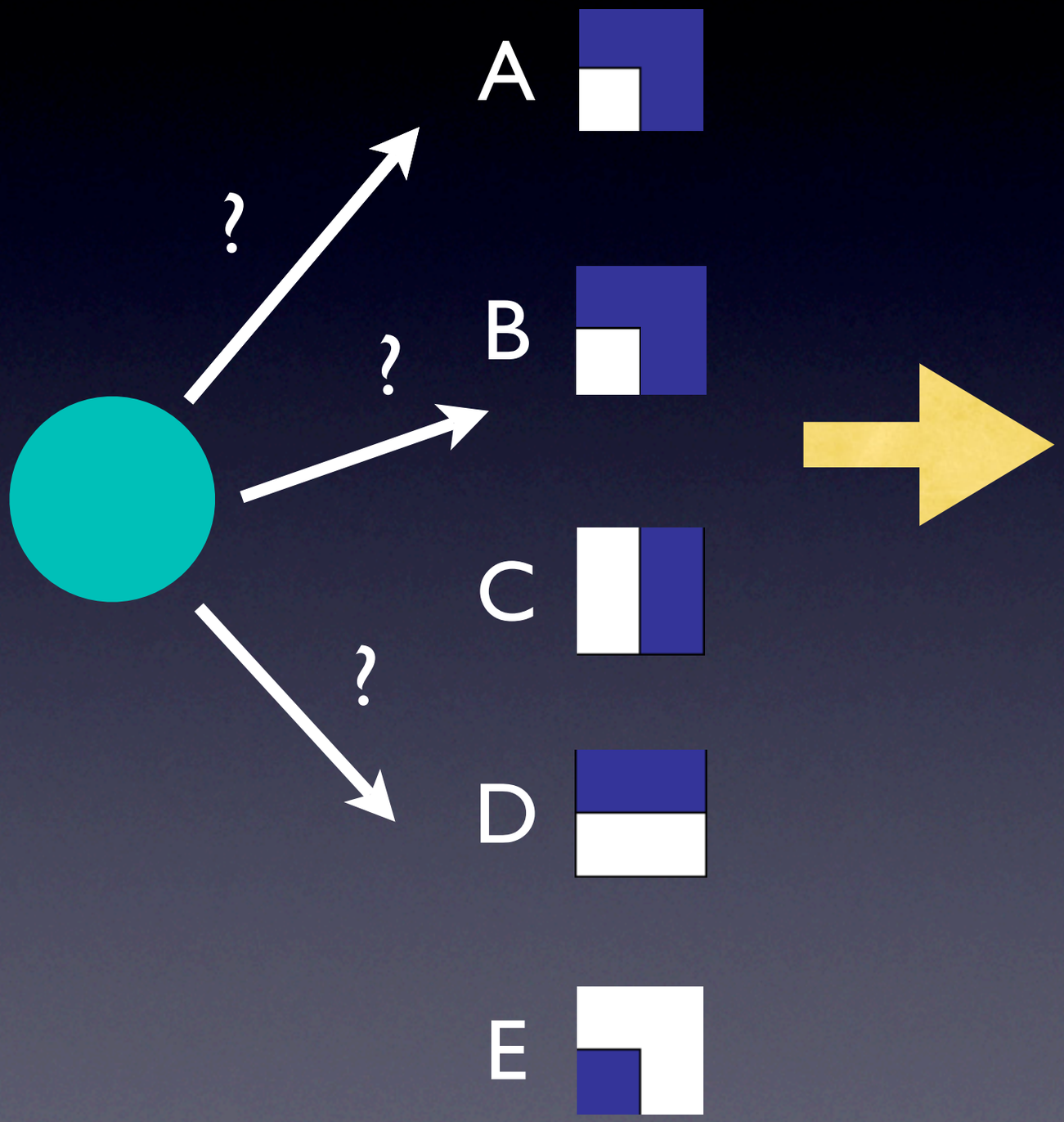
Select peers A to D step by step



Overlap Awareness

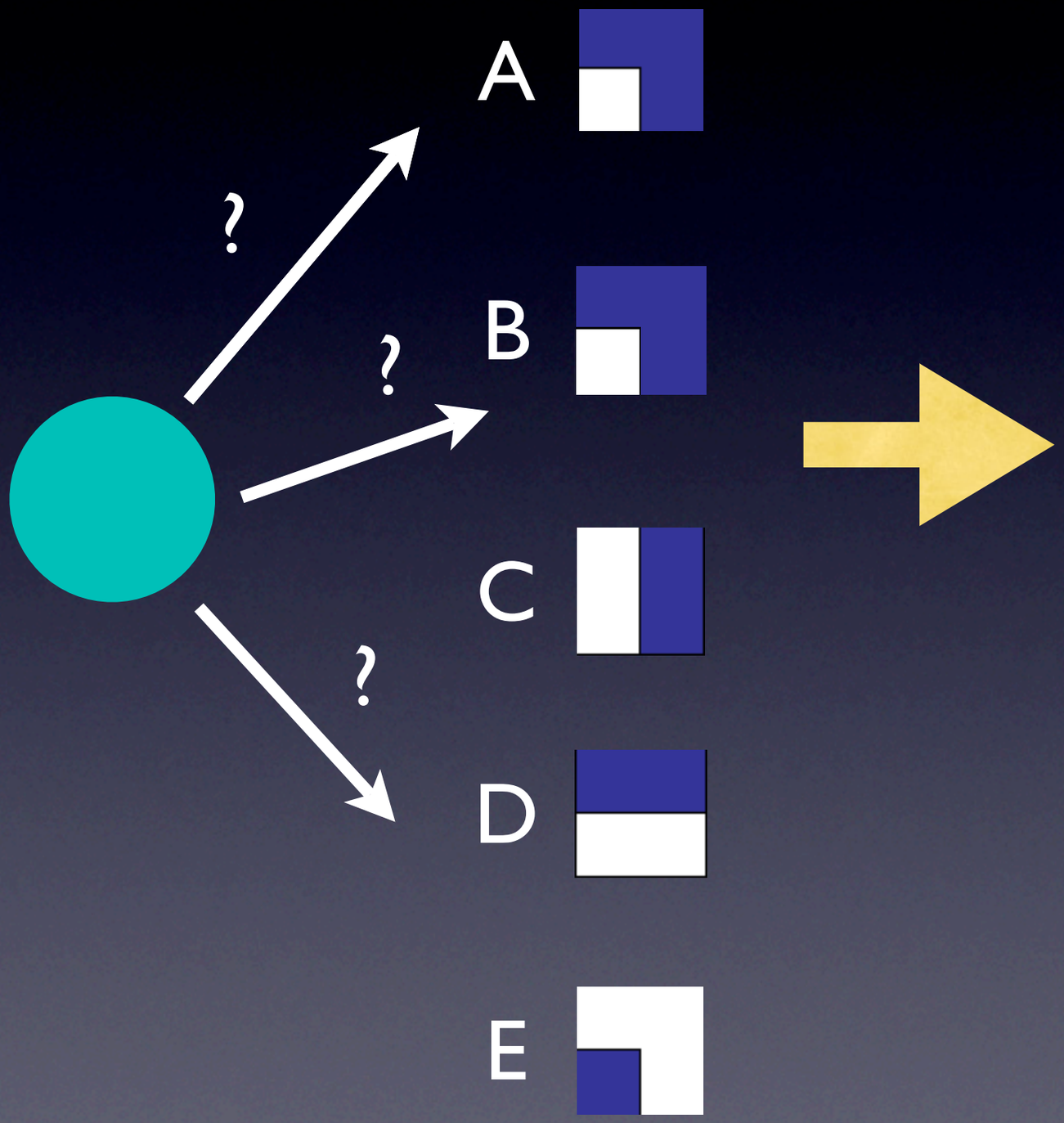


Overlap Awareness



Overlap Aware Strategy:

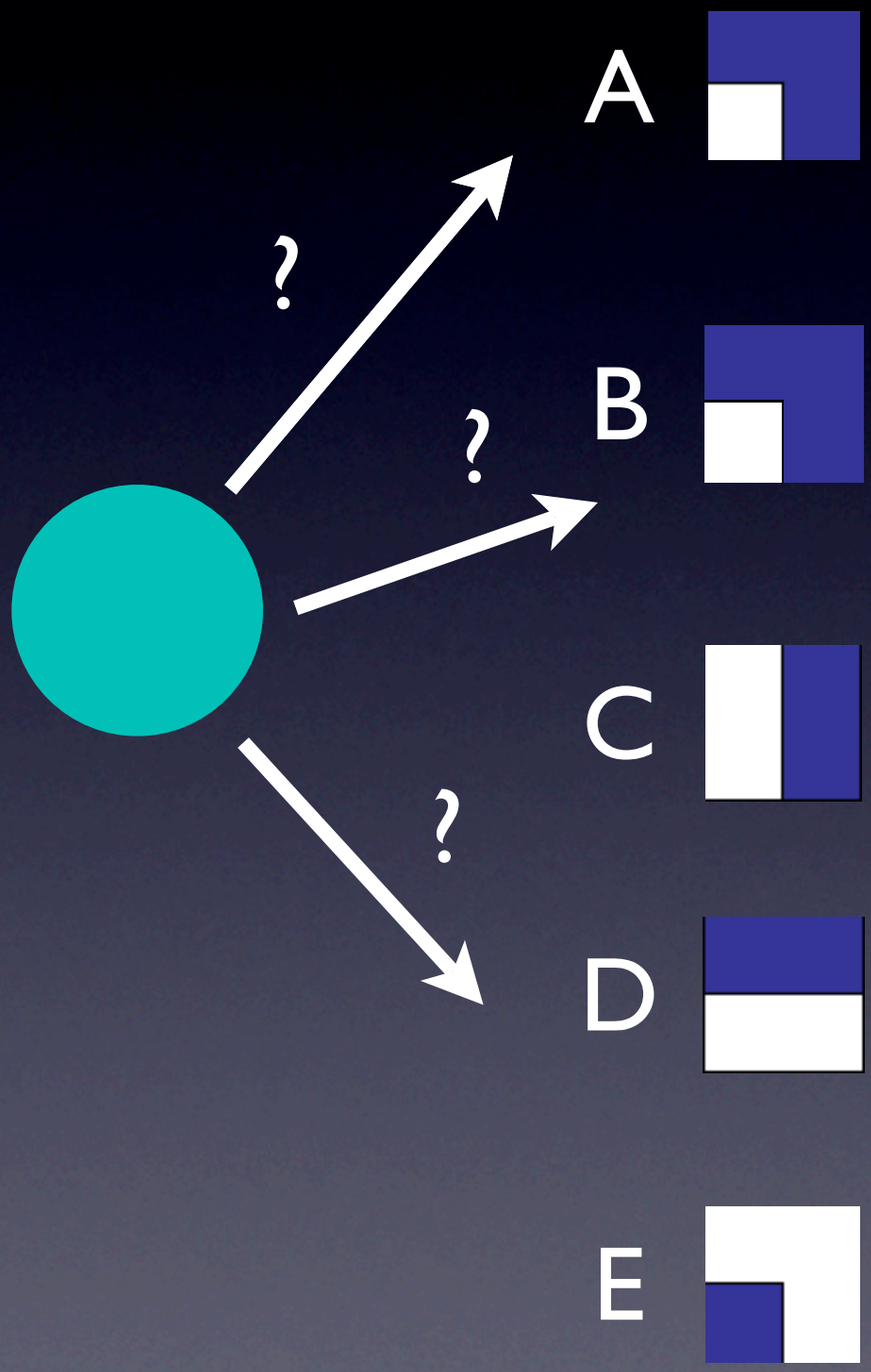
Overlap Awareness



<u>Overlap Aware Strategy:</u>

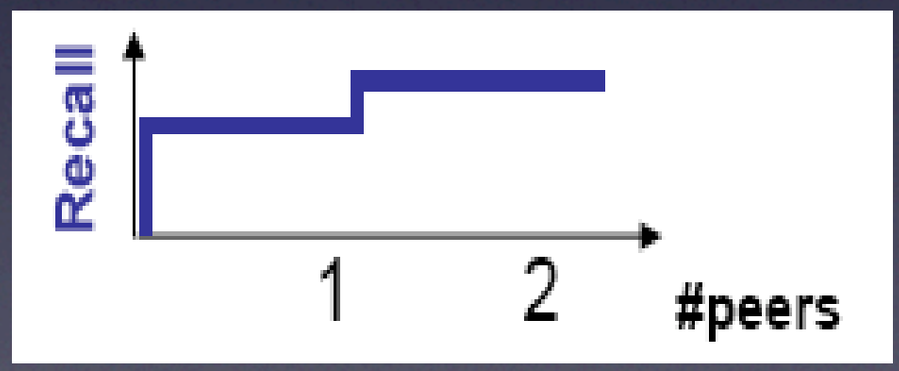
Combine A and E for max. Recall by min. of peers

Overlap Awareness



<u>Overlap Aware Strategy:</u>

Combine A and E for max. Recall by min. of peers



Overlap Aware Routing

peer2
peer3
peer5
peer8
peer9
peer7

peer1
peer2
peer5
peer7
peer10
peer15

Overlap:

peer	rank
2	2*20
5	2*18
1	23
3	20
10	15
7	2*7
8	13
9	12
15	6

Overlap Aware Routing

peer2
peer3
peer5
peer8
peer9
peer7

peer1
peer2
peer5
peer7
peer10
peer15

Step 1: Select Peer2

peer	rank
2	2*20
5	2*18
1	23
3	20
10	15
7	2*7
8	13
9	12
15	6

Overlap:

Overlap Aware Routing

peer2
peer3
peer5
peer8
peer9
peer7

peer1
peer2
peer5
peer7
peer10
peer15

Step 1: Select Peer2

peer	rank
2	---
3	17
5	16
1	16
7	15
9	12
8	11
15	6
10	7

Overlap:

Overlap Aware Routing

Step 1: Select Peer2

peer	rank
2	---
3	17
5	16
1	16
7	15
9	12
8	11
15	6
10	7

Overlap:

Peer2

Overlap Aware Routing

peer3
peer5
peer7
peer9
peer8

peer5
peer1
peer7
peer15
peer10

Step 1: Select Peer2

peer	rank
2	---
3	17
5	16
1	16
7	15
9	12
8	11
15	6
10	7

Overlap:

Peer2

Overlap Aware Routing

peer3
peer5
peer7
peer9
peer8

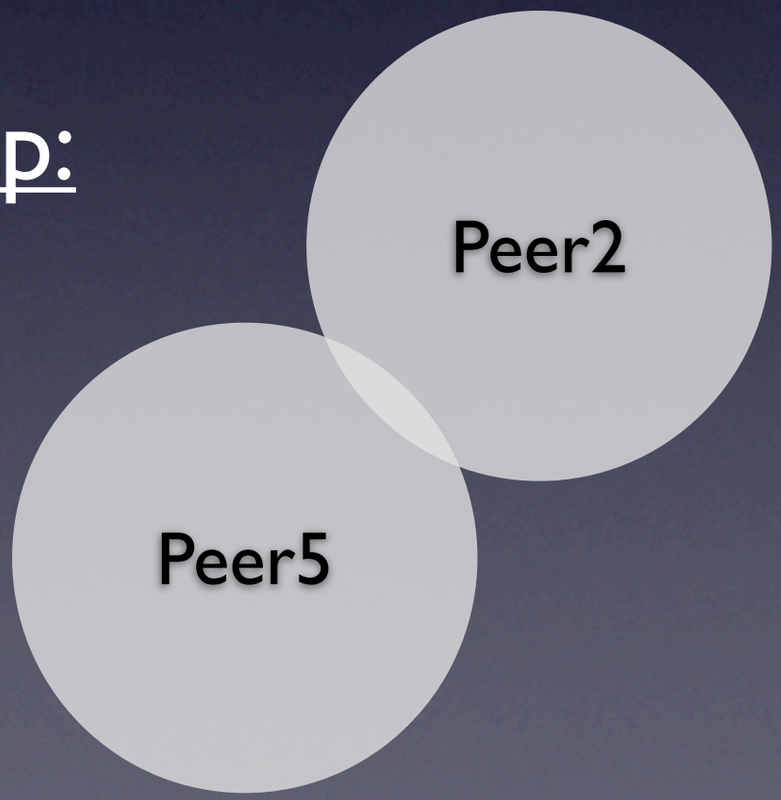
peer5
peer1
peer7
peer15
peer10

Step1: Select Peer2

Step2: Select Peer5

peer	rank
2	---
3	17
5	16
1	16
7	15
9	12
8	11
15	6
10	7

Overlap:



Overlap Aware Routing

peer3
peer5
peer7
peer9
peer8

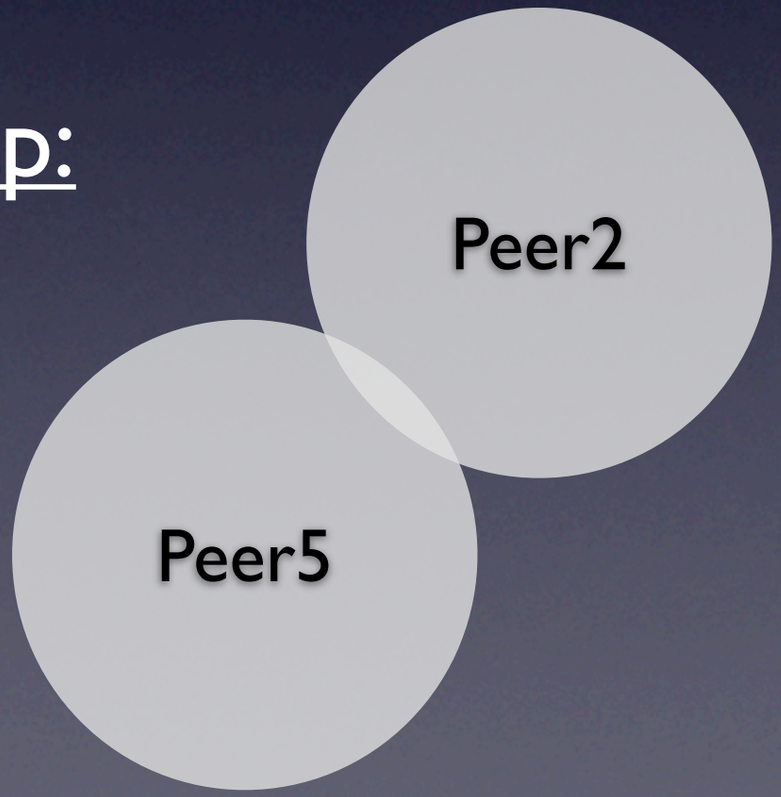
peer5
peer1
peer7
peer15
peer10

Step1: Select Peer2

Step2: Select Peer5

peer	rank
2	---
5	---
10	14
7	12
1	12
9	12
3	11
8	6
15	7

Overlap:



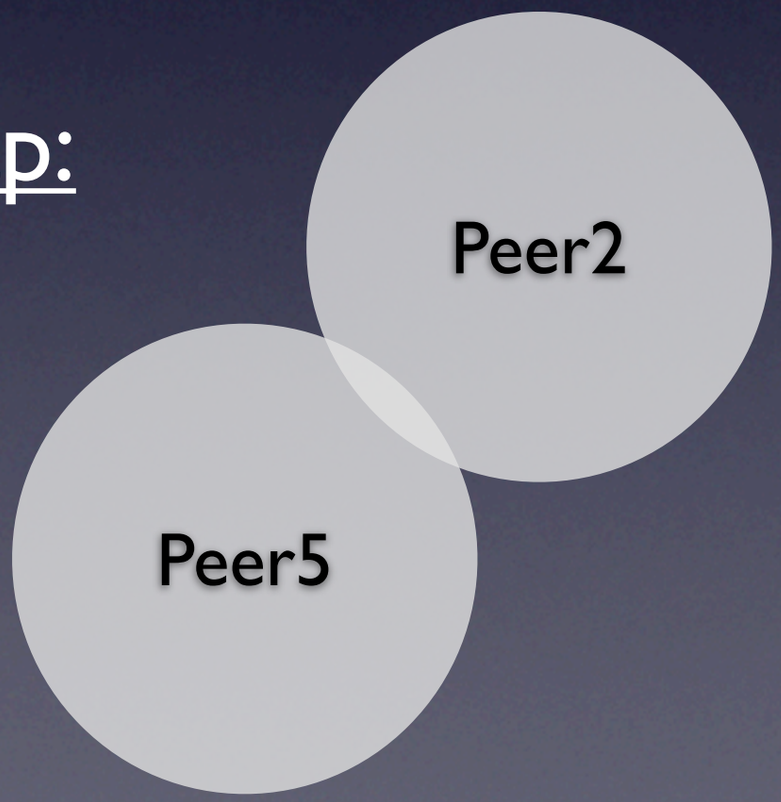
Overlap Aware Routing

Step 1: Select Peer2

Step2: Select Peer5

peer	rank
2	---
5	---
10	14
7	12
1	12
9	12
3	11
8	6
15	7

Overlap:



Overlap Aware Routing

peer7
peer9
peer3
peer8

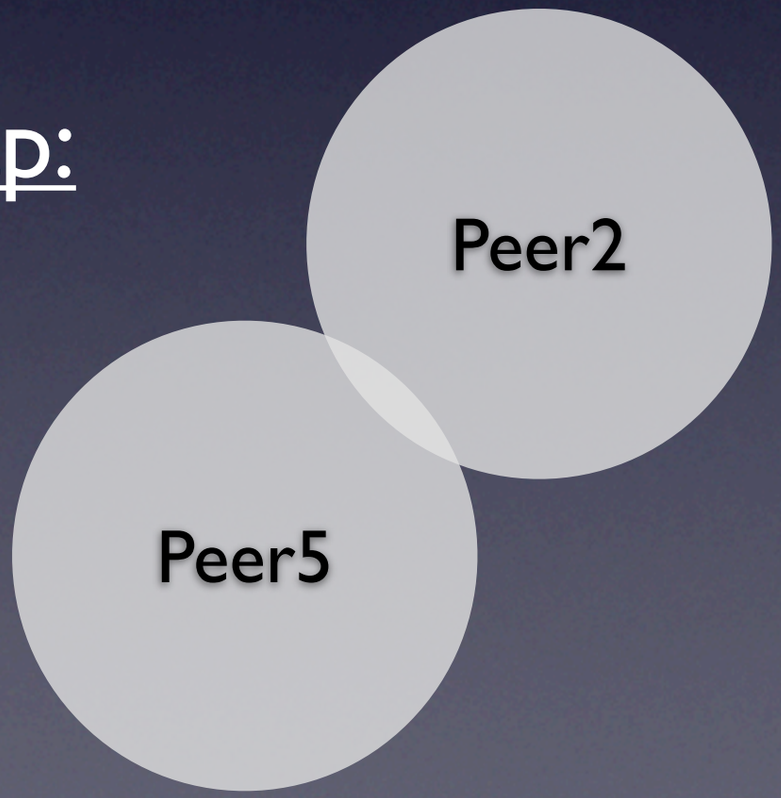
peer10
peer7
peer1
peer15

Step1: Select Peer2

Step2: Select Peer5

peer	rank
2	---
5	---
10	14
7	12
1	12
9	12
3	11
8	6
15	7

Overlap:



Overlap Aware Routing

peer7
peer9
peer3
peer8

peer10
peer7
peer1
peer15

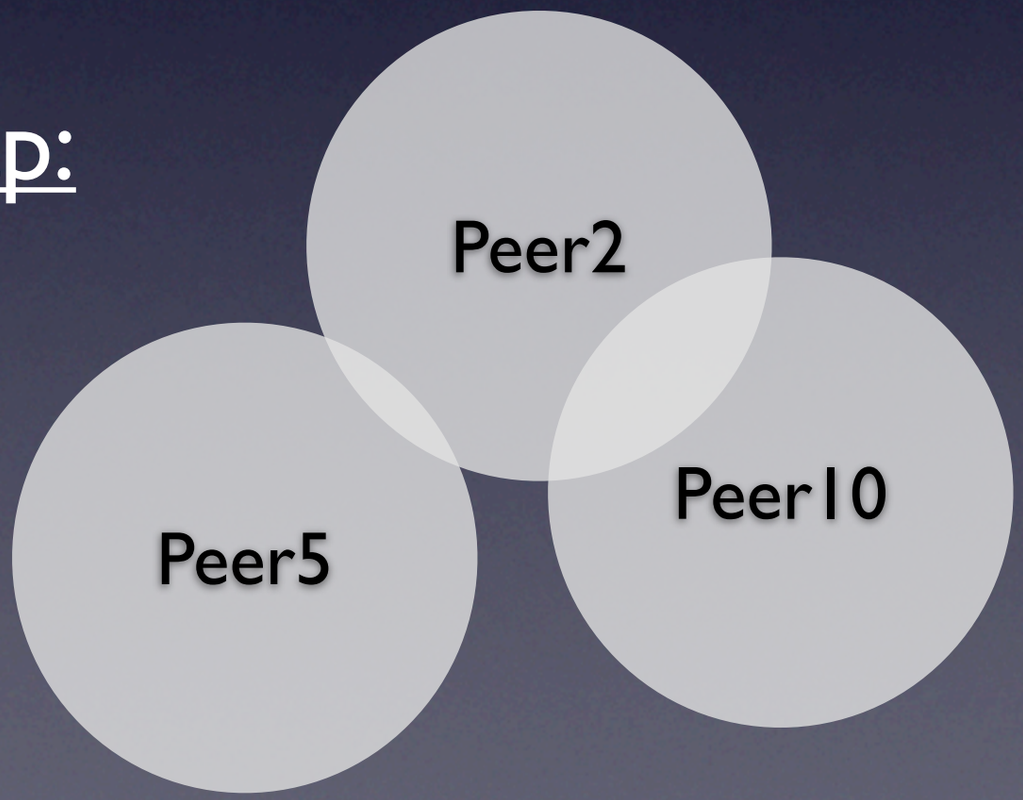
Step1: Select Peer2

Step2: Select Peer5

Step3: Select Peer10

peer	rank
2	---
5	---
10	14
7	12
1	12
9	12
3	11
8	6
15	7

Overlap:



Overlap Aware Routing

peer7
peer9
peer3
peer8

peer10
peer7
peer1
peer15

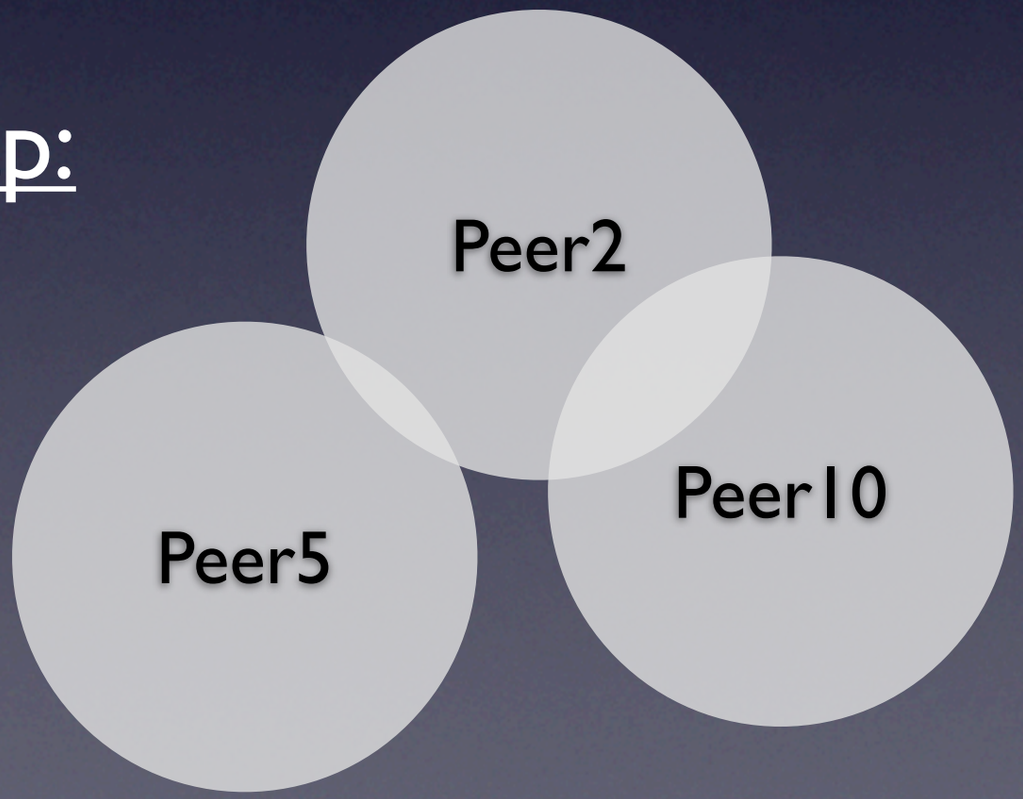
Step1: Select Peer2

Step2: Select Peer5

Step3: Select Peer10

peer	rank
2	---
5	---
10	---
9	10
3	12
1	11
7	11
15	6
8	7

Overlap:



Overlap Aware Routing

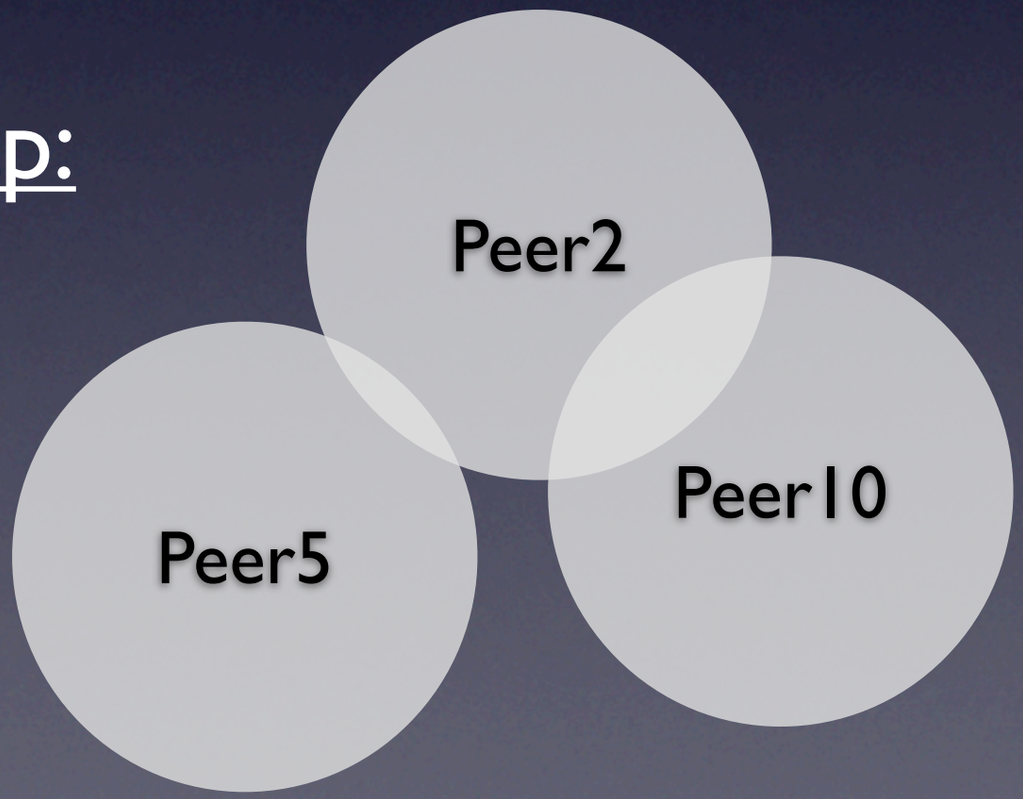
Step1: Select Peer2

Step2: Select Peer5

Step3: Select Peer10

peer	rank
2	---
5	---
10	---
9	10
3	12
1	11
7	11
15	6
8	7

Overlap:



Overlap Aware Routing

peer9
peer3
peer7
peer8

peer1
peer7
peer15

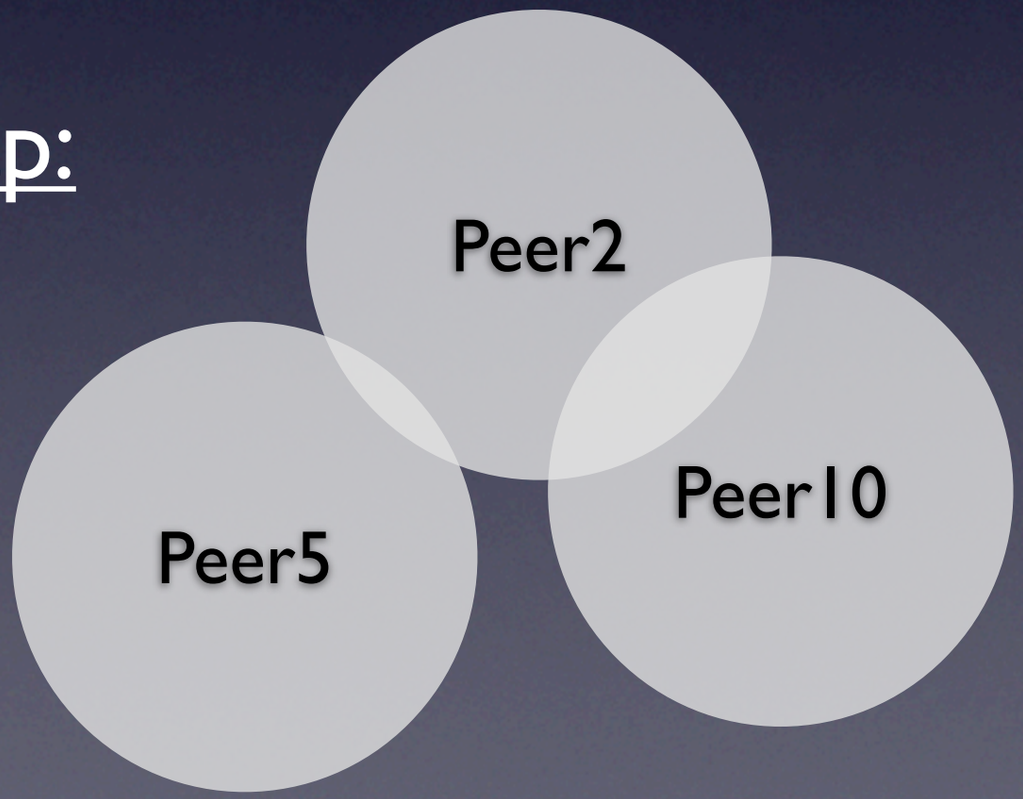
Step1: Select Peer2

Step2: Select Peer5

Step3: Select Peer10

peer	rank
2	---
5	---
10	---
9	10
3	12
1	11
7	11
15	6
8	7

Overlap:



Overlap Aware Routing

peer9
peer3
peer7
peer8

rank
10
12
11
11
6
7

Problem of efficient peer selection !!

Ove

Peer5

Peer10

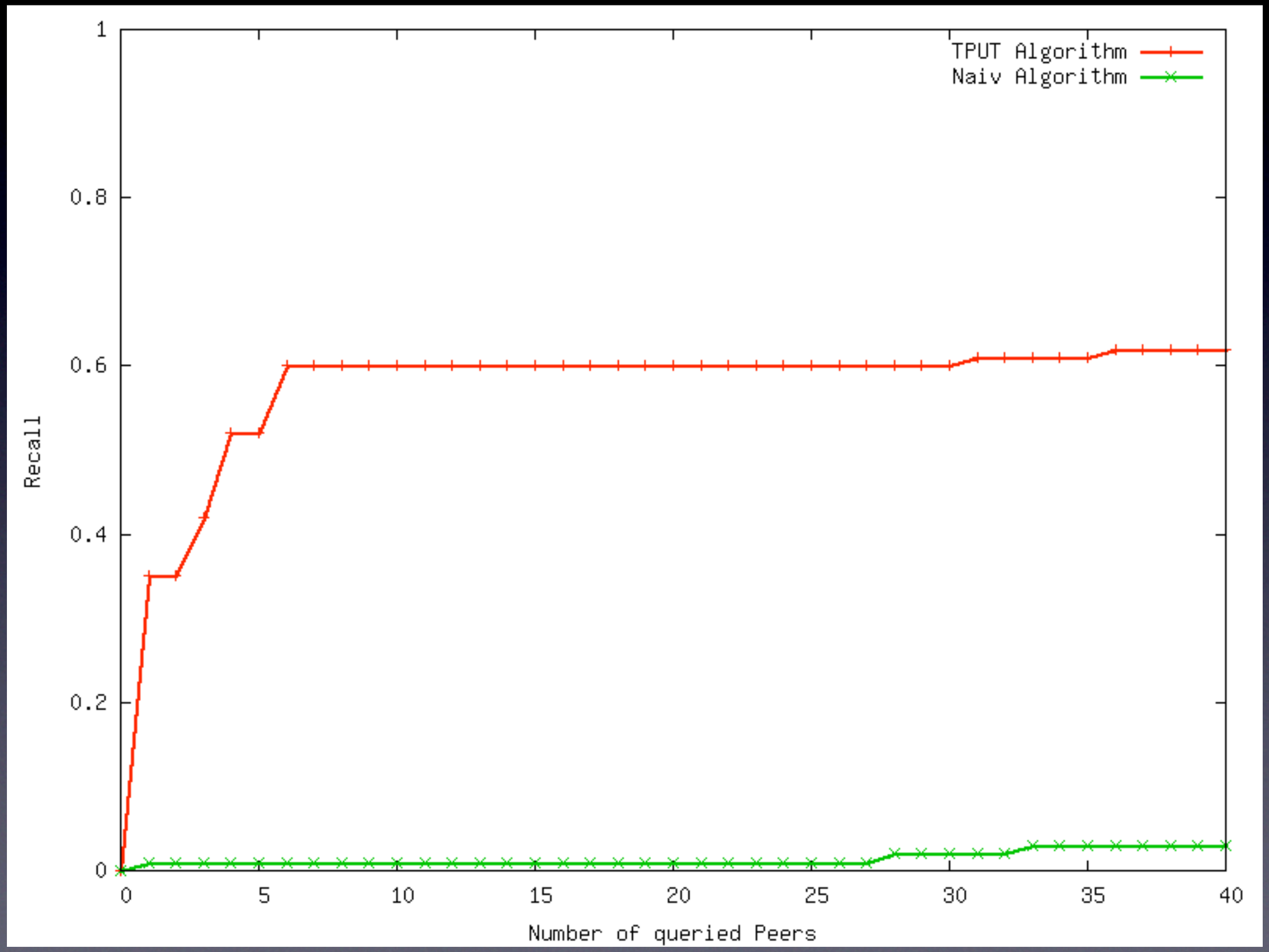
Problems

- TPUT not applicable for Overlap Aware Routing, needs a stable rank
- Naiv Algorithm could be to iterate over all peers, but too expensive
- „Batching“: Select parts of documentlists from every peer, but what exactly and from which peer ?

Experimental Setup

- Wikipedia dump (approx. 5.1GB raw data) as test data in PostgreSQL Database
- Linux on Intel 3GHz and 1GB memory
- Data clustered in 1.000 different peers (with overlaps)
- Data Tables:
  - Table for statistics, includes: peerid, term
  - Main table with all informations (term, score, termfrequency, docid, peerid, doclength, ...)
  - A lot of indices

Results


TPUT Algorithm
Naiv Algorithm
Recall
Number of queried Peers
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
25
30
35
40

Conclusion / Ongoing Work

- Problem in peer selection if rank / score changes in every iteration step
- Framework running, queries working, TPUT fully implemented
- Now: Find algorithm that computes peer selection better and efficient for dynamical ranking condition

End

Thank you