Chapter 3

Lower Bounds from SETH

Danupon Nanongkai

KTH, Sweden

ADFOCS 2018
Last edited: Aug. 16, 2018

Summary

OMv conjecture

- Usually refutes $\boldsymbol{n^{1-\epsilon}}$ update time on **dense** graph, or $\boldsymbol{m^{\frac{1}{2}-\epsilon}}$ in general.
- This helps refute polylogarithmic time, but might not be tight.

SETH

- **SETH** refutes $\boldsymbol{n^{1-\epsilon}}$ time for **sparse** graph, refuting $\boldsymbol{m^{1-\epsilon}}$ in general.
- To start from SETH, reduce from **dynamic OV,** a.k.a. **dynamic client-server.**
- For some problem (e.g. diameter), SETH even refutes $\boldsymbol{n^{2-\epsilon}}$ bound (but need **3OV**)!

n = # of nodes, m=# of edges
* There are some exceptions.

Part 1

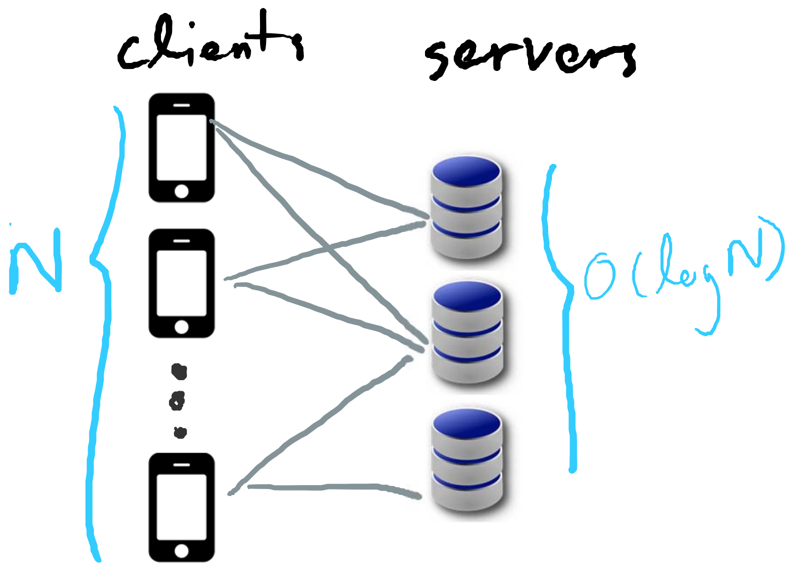
SETH and Dynamic OV

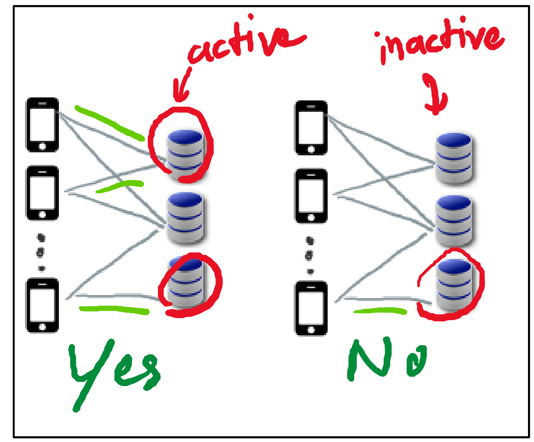
(a.k.a. Dynamic Client-Server Problem)

Starting point: Client-Server Problem

Preprocess:

$poly(N)$ time





Updates: A server becomes **active/inactive**

Output: All clients are **connected** to active servers?

Naïve algorithm takes **O(N)** update time

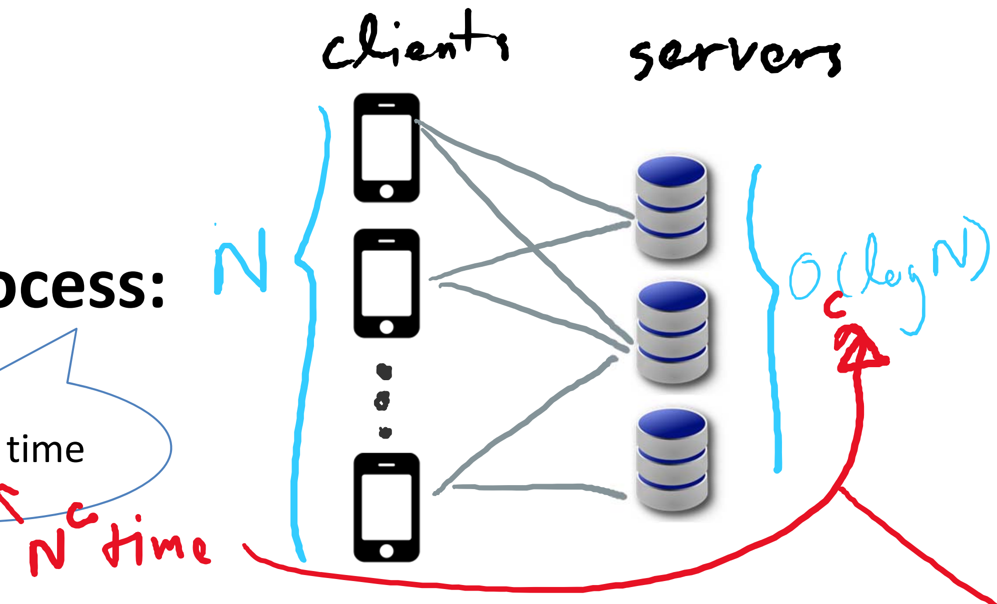
SETH→
No $N^{1-\epsilon}$ amortized per server update

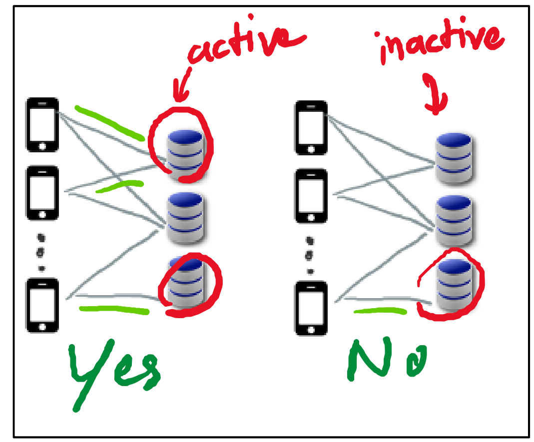
Sparsity:
#edges $= O(N \log N)$

Starting point: Client-Server Problem

Preprocess:

$poly(N)$ time





Details: #Servers depends on preprocessing time

Updates: A server becomes **active/inactive**

Output: All clients are **connected** to active servers?

Naïve algorithm takes **O(N)** update time

SETH→

No $N^{1-\epsilon}$ amortized per server update

Sparsity:

#edges $= O(N \log N)$

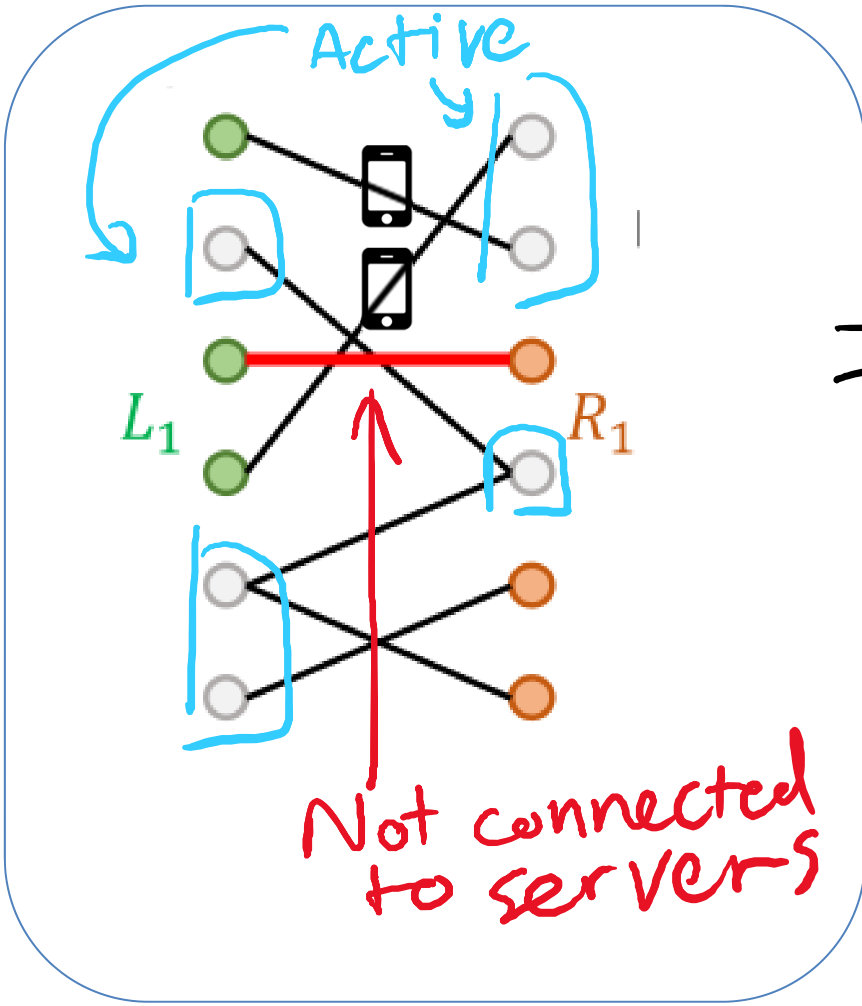
Details: #servers depends on preprocessing time

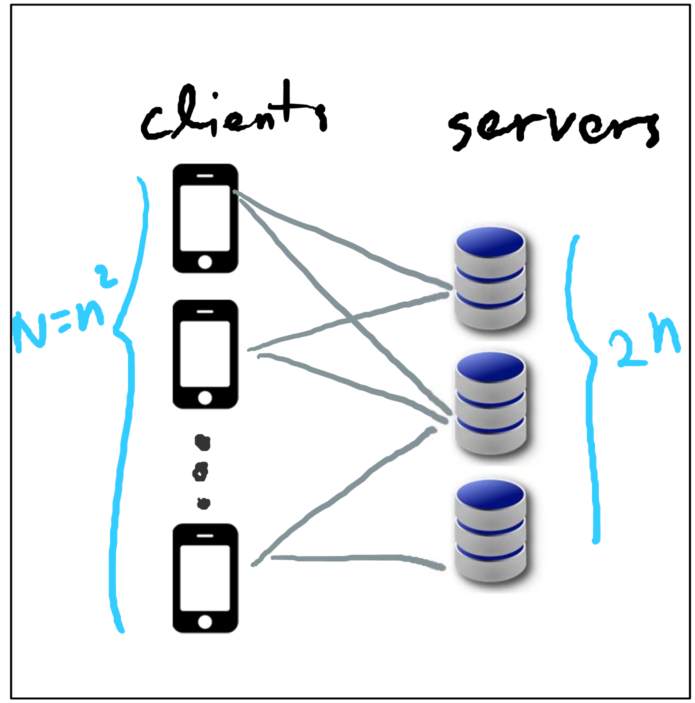
- Otherwise, you can prepare for all $2^{\#servers} = n^{O(1)}$ possible sets of active servers during the preprocessing time.

- The claim should be interpreted as: If someone claims to have an algorithm with $\boldsymbol{N^c}$ preprocessing time, then we can pick the number of servers to be $\boldsymbol{f(c)\log N}$. Then, SETH implies that there is no algorithm with $\boldsymbol{N^c}$ preprocessing time and $\boldsymbol{N^{1-\epsilon}}$ update time.

Motivation: OuMv can be viewed as Client-Server with **#clients=#server²**


Optional





OMv →
No $N^{1/2-\epsilon}$ amortized time per node recolor with polynomial preprocess

Claim: SETH implies no $N^{1-\epsilon}$ amortized per server update

Proof (sketched):

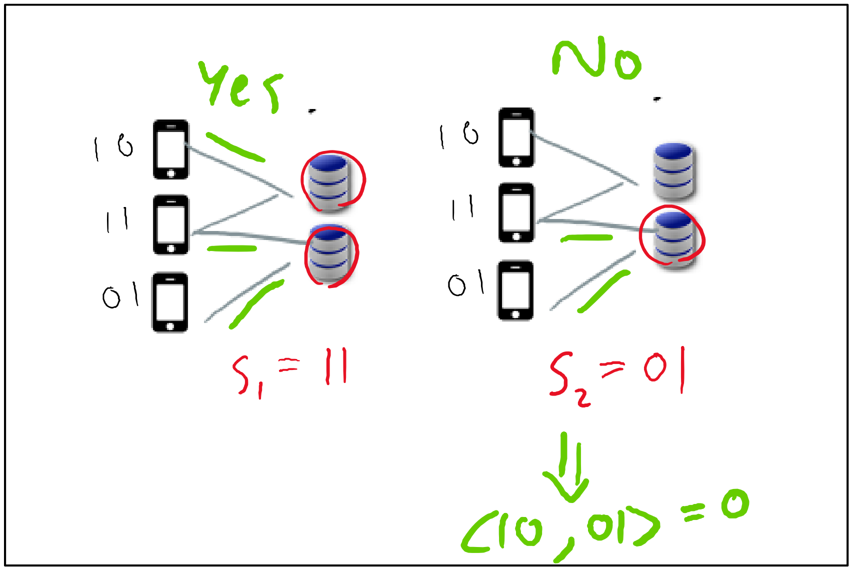
OV: Given sets **A** and **B** of vectors, exists $u \in A, v \in B$ s.t. $\langle u, v \rangle = 0$?

- **SETH** implies **no** $(|A||B|)^{1-\epsilon}$ time.
- Hold for: $|A| = N$, $|B| = poly(N)$ and **dimension=n=O(log N)**

Reduction:

1. Vectors in A → **Clients.**
2. Coordinates→ **Servers.**
3. Each vector in B → Each set of active servers

Example: A={10,11,01}, B={11, 01}



Claim: SETH implies no $N^{1-\epsilon}$ amortized per server update

Proof (sketched):

OV: Given sets **A** and **B** of vectors, exists $u \in A, v \in B$ s.t. $\langle u, v \rangle = 0$?

- **SETH** implies **no** $(|A||B|)^{1-\epsilon}$ time.
- Hold for: $|A| = N$, $|B| = poly(N)$ and **dimension=n=O(log N)**

Reduction:

1. Vectors in A → **Clients.**
2. Coordinates→ **Servers.**
3. Each vector in B → Each set of active servers

Example: A={10,11,01}, B={11, 01}

Analysis:

- Preprocessing time = $poly(|A|) < (|A||B|)^{1-\epsilon}$ if |B| is big enough compared to |A|.
- Assume time per server update is $N^{1-\epsilon}$. Then, time per vector $v \in B$ is $(N)^{1-\epsilon}n$.
- So total time is $|B|(Nn)^{1-\epsilon} = |A|^{1-\epsilon}|B| = (|A||B|)^{1-\epsilon}$

Another form: Dynamic OV

- Preprocess: Set **A** of Boolean vectors
  - Let **N**=|A|. Vectors have dimension **O(log N).**
- Update: A Boolean vector v
- Output: Exists $u \in A$ s.t. $\langle u, v \rangle = 0$?

Naïve algorithm takes O(N log N) time per v.

Claim: SETH→No $N^{1-\epsilon}$-time algorithm with polynomial preprocessing time.

Part 2

Some Reductions from Dynamic OV (Client-Server)

Plan

- **Single-Source Reachability Count (#SSR):** Counting number of nodes reachable from **s**
- **Strongly-Connected Component Count (#SCC):** Counting number of strongly connected components

<u>Lesson</u>: SETH may give higher lower bounds (than OMv) in **m** for **counting** versions.

Intuition: *The client-server problem is about the* ***number*** *of connected clients.*

Example 1

Reachability - #SSR

st-Reachability (recall)

- Exists directed path from s to t?
- **No $n^{1-\epsilon}$** update time (on dense graph) assuming **OMv**
  - Hold against randomized and amortized algorithms
  - Implies $m^{1/2-\epsilon}$ lower bound
- **Open:** Higher lower bound*?

Single-Source Reachability Count (#SSR)

- How many nodes are reachable from s?
- No $m^{1-\epsilon}$ update time assuming **SETH**
  - Hold against randomized and amortized algorithms

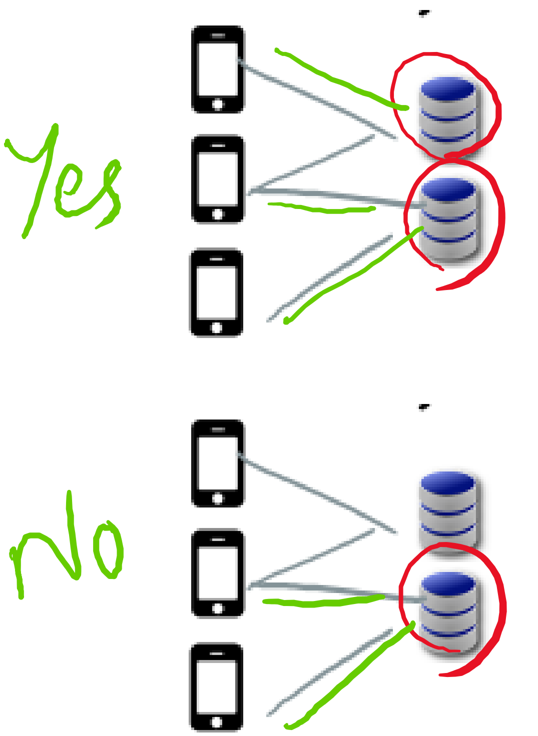
Later: **SSR** with **$n^{o(1)}$** query time also has **no $m^{1-\epsilon}$** update time assuming **OMv.**

n = # of nodes, m=# of edges
* We do have higher bound for worst-case update time

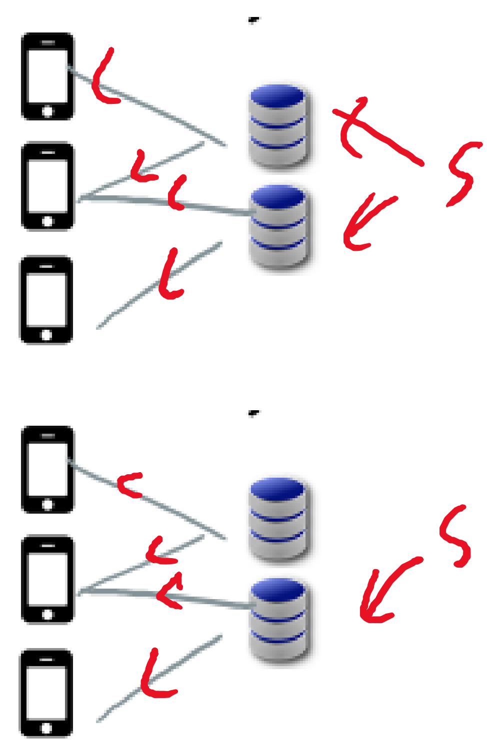
Claim: No $m^{1-\epsilon}$ update time assuming SETH

Reduction: Add edges from s to all active servers

Servers-Clients



#SSR



s reaches all clients
#SSR=#clients+#(active servers)

s **doesn't** reaches all clients
#SSR<#clients+#(active servers)

SETH→No $n^{1-\epsilon}$ time per **server** update

→No $n^{1-\epsilon}$ time per **edge** update for #SSR

→No $m^{1-\epsilon}$ time since graph is **sparse**!

n = # of nodes, m=# of edges

Example 2

Strong Connectivity- #SCC

Strong Connectivity (recall)

- Exists directed path from **every** s to **every** t?
- **No $\boldsymbol{n^{1-\epsilon}}$** update time assuming **OMv**
  - Hold against randomized and amortized algorithms
  - Implies $\boldsymbol{m^{1/2-\epsilon}}$ lower bound
- **Open:** Higher lower bound?

Strongly Connected Components Count (#SCC)

- How many strongly connected components are there?
- **No $\boldsymbol{m^{1-\epsilon}}$** update time assuming **SETH**

n = # of nodes, m=# of edges

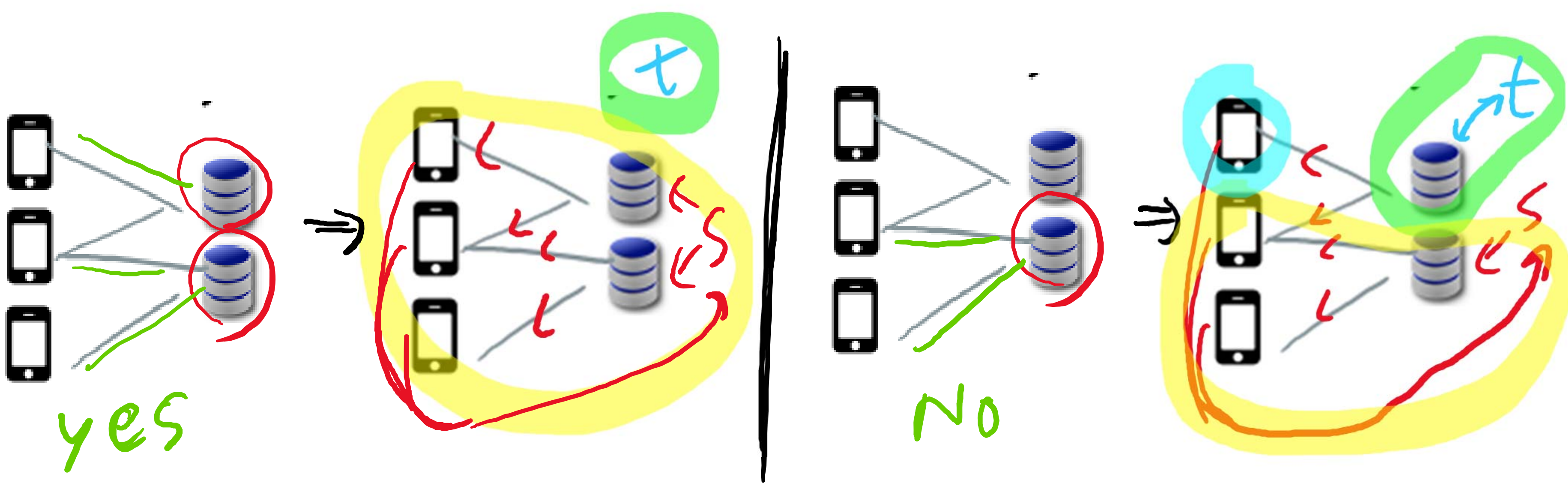
Claim: No $m^{1-\epsilon}$ update time assuming SETH

Reduction:

- Add edges from all clients to **s** to all **active** servers
- Add edges between **t** and all **inactive** servers

Observe: Yes for clients-servers ←→ #SCC≤2.

- All active servers and adjacent clients form one component with s.
- Other clients are not in any connected components.
- Inactive servers form another component with y.



Part 3

Diameter from dynamic 3OV

Dynamic Diameter: Output the diameter of an **undirected** graph

Algorithms

- **Naïve algorithm:** O(mn) per update.
- **Best** **(via APSP):** $O(n^2)$ amortized update time and $O\left(n^{2\frac{2}{3}}\right)$ worst-case.

Lower Bounds

- **No $\boldsymbol{n^{1-\epsilon}}$** update time assuming **OMv** [Thanks to a participant!]
  - Implies $\boldsymbol{m^{1/2-\epsilon}}$ lower bound
- No $\boldsymbol{n^{2-\epsilon}}$ update time assuming **SETH**
  - **Not** known how to prove this from **dynamic OV (client-server)**
  - Instead, reduce from **dynamic 3-OV**

Both hold against randomized and amortized algorithms

Lesson: Dynamic 3-OV might be useful for problems that involve **many pairs of nodes**.

n = # of nodes, m=# of edges

Dynamic 3-OV

- Preprocess: Set **A, B** of Boolean vectors
  - Let **N**=|A|=|B|. Vectors have dimension **O(log N).**
- Update: A Boolean vector w
- Output: Exists $u \in A, v \in B$ s.t. entry-wise multiplication of $u \circ v \circ w = \vec{0}$?

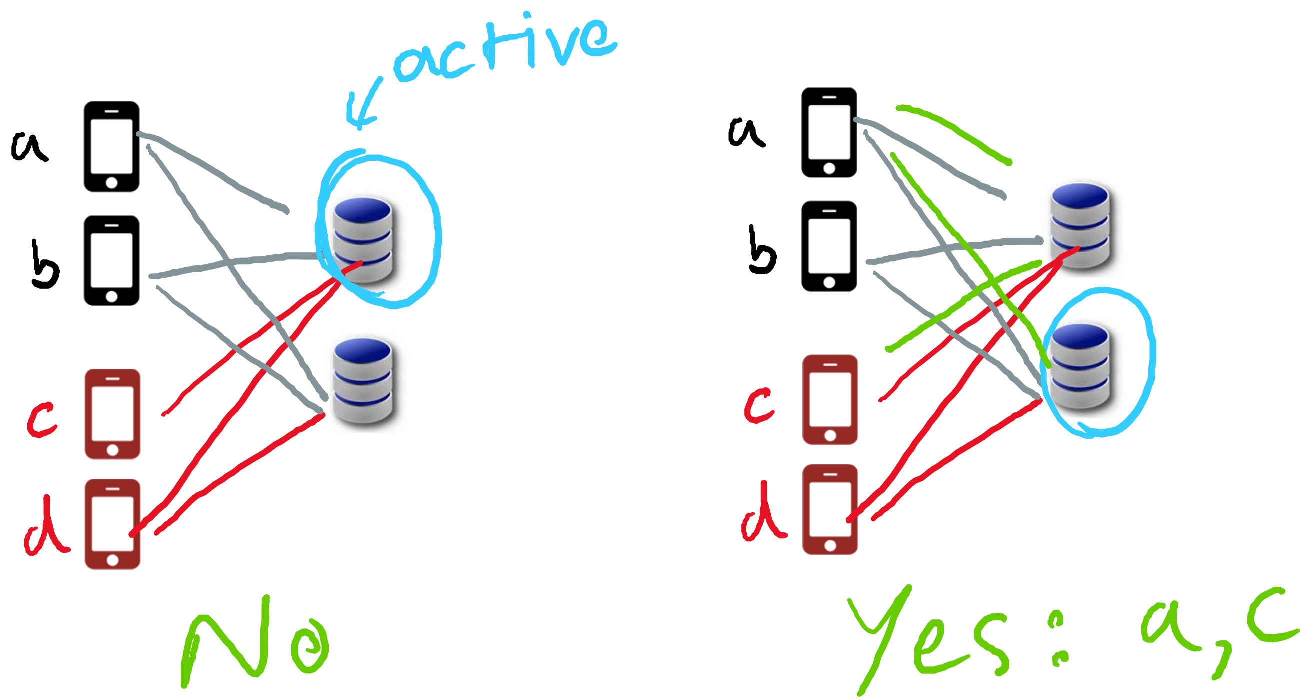
$u = (1,0,\mathbf{1})$
$v = (0,1,\mathbf{1})$
$w = (1,1,\mathbf{1})$
$u \circ v \circ w = (0,0,\mathbf{1})$

Naïve algorithm takes O(N^2 log N) time per v
(Keep track of all pairs)

Claim: SETH → No $N^{2-\epsilon}$-time algorithm with polynomial preprocessing time
Proof: Omitted.

Client-Server Form of dynamic 3OV

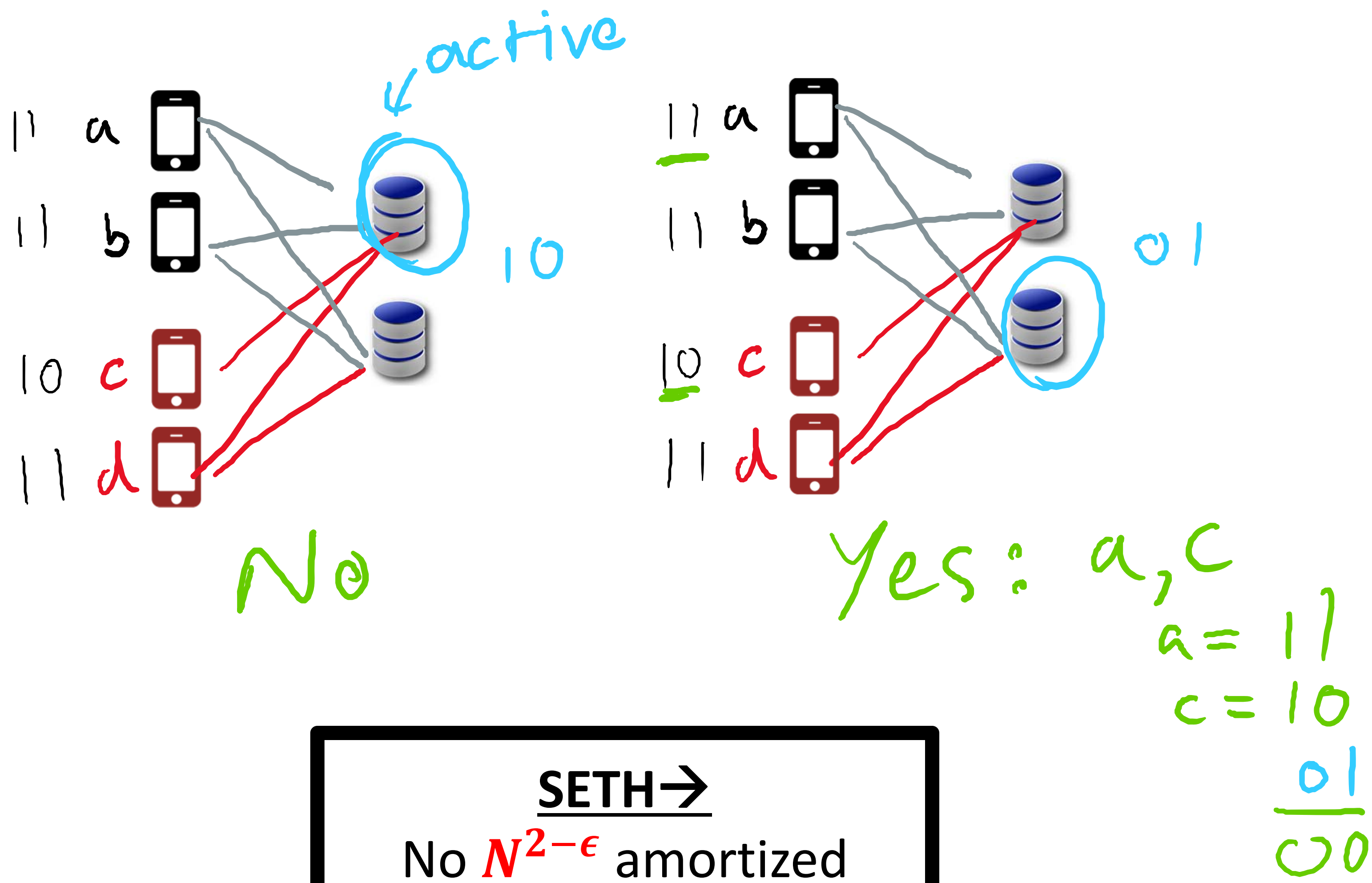
Exists pair of red-black client that doesn't share active server?



SETH→

No $N^{2-\epsilon}$ amortized per server update

Example of how it's related to 3OV

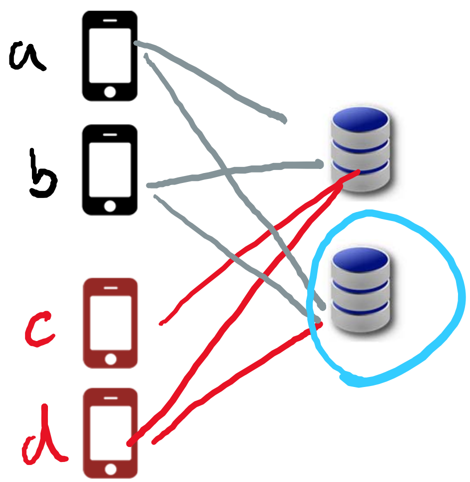


SETH→
No $N^{2-\epsilon}$ amortized
per server update

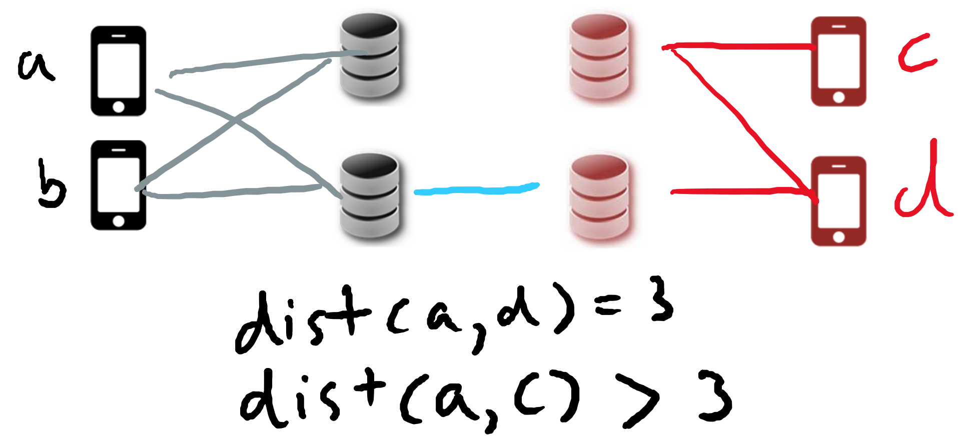
Reduction to Diameter (partial)

1. Create copies of servers.
2. Connect black and red clients to different copies.
3. If a server is active, connect its two copies.

3OV: Exists pair without shared server?



Diameter



Intuition: Red-Black clients that share active servers has distance 3. (Otherwise distance will be more.)

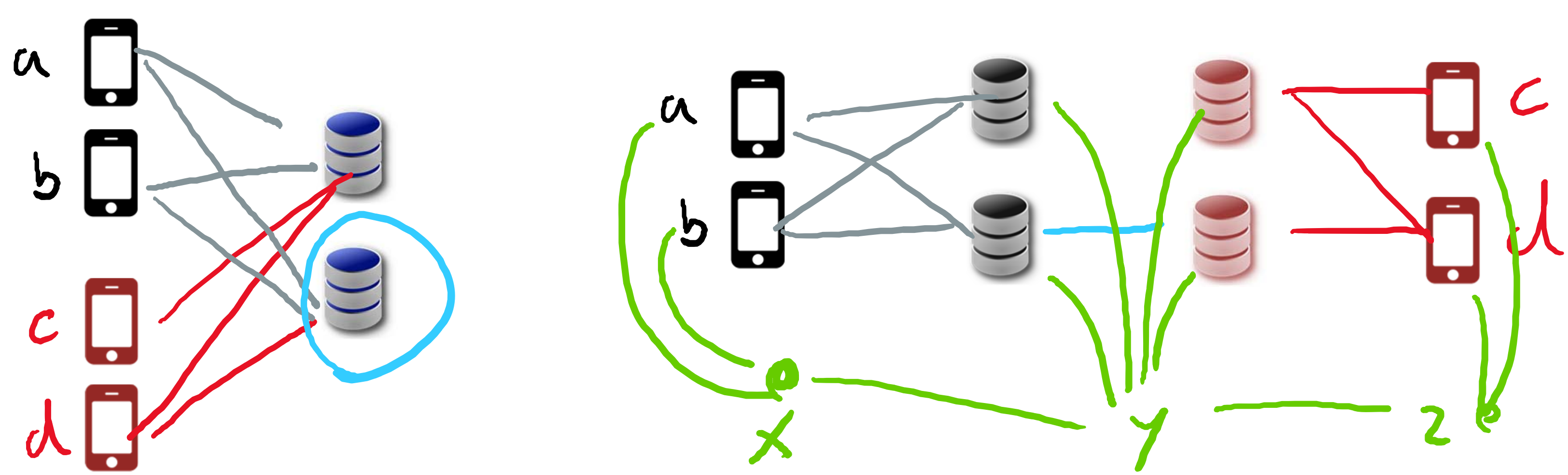
Problem: How about black-black clients, etc?

Reduction to Diameter (full)

1. Create copies of servers.
2. Connect black and red clients to different copies.
3. If a server is active, connect its two copies.
4. **Add edges between black clients and x, servers and y, red clients and z.**

3OV: Exists pair without shared server?

Diameter



Claim (Tedious to check): Diameter > 3 **iff** exists pair without shared server

Questions?

Thanks to co-authors:

Sayan Bhattacharya, Jan van den Brand, Deeparnab Chakraborty, Sebastian Forster, Monika Henzinger, Christian Wulff-Nilsen, Thatchaphol Saranurak

This project has received funding from the *European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme* under grant agreement No 715672