Distributed Shortest Paths and Related Problems

Danupon Nanongkai
University of Vienna, Austria

About this talk

A survey of problems and techniques related to recent progress on computing shortest paths on distributed networks

1. Problems: **s-t-distance**, Single-source shortest paths (SSSP), All-pairs shortest paths (APSP), Distance Labeling, Diameter, Routing, etc.

2. Algorihtmic Techniques: Weight approximation, Skeleton, Spanner/Emulator, Hopset

<u>Note</u>

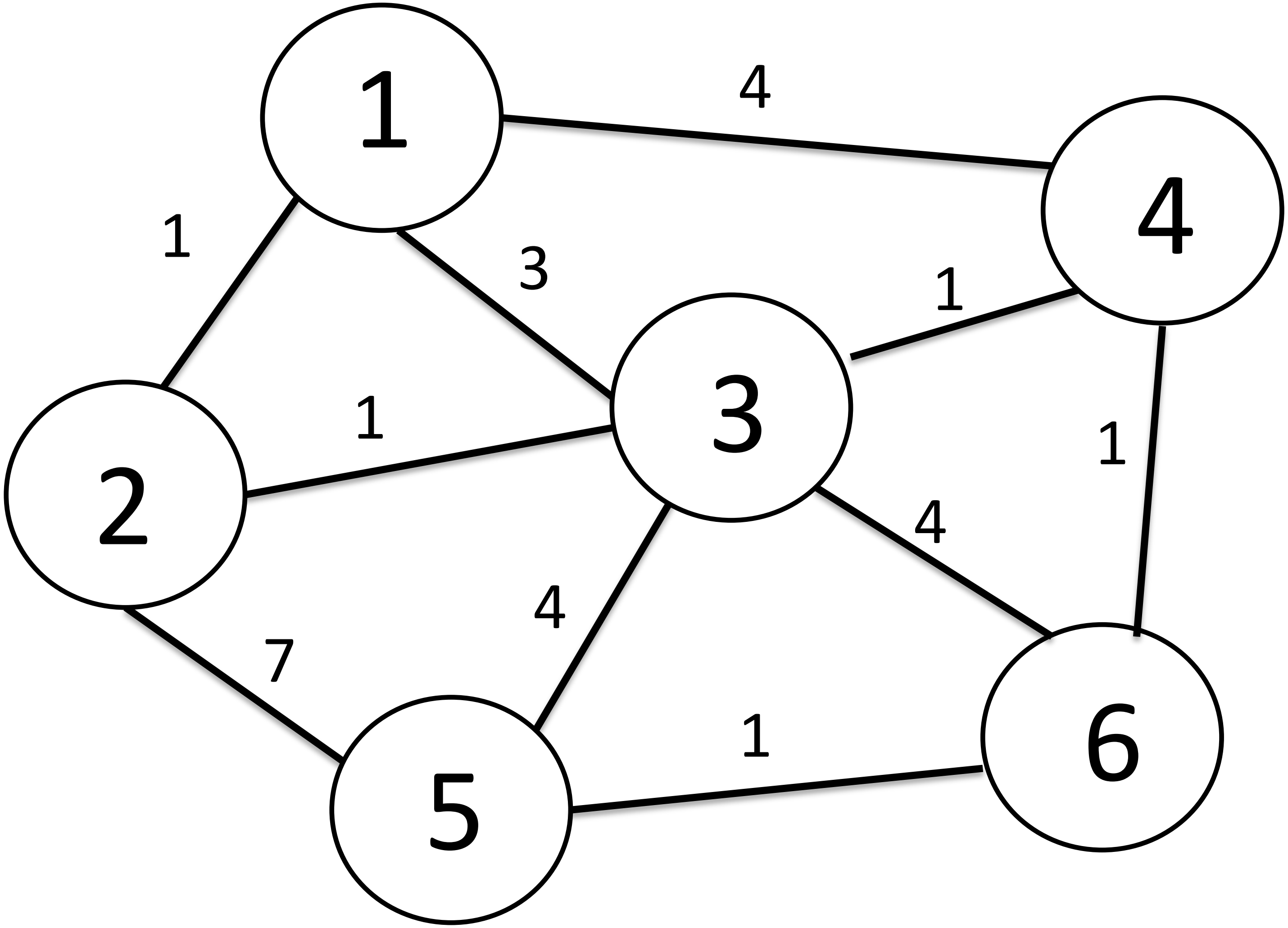
polylog terms will be
hidden most of the time

Part 1

Introuction

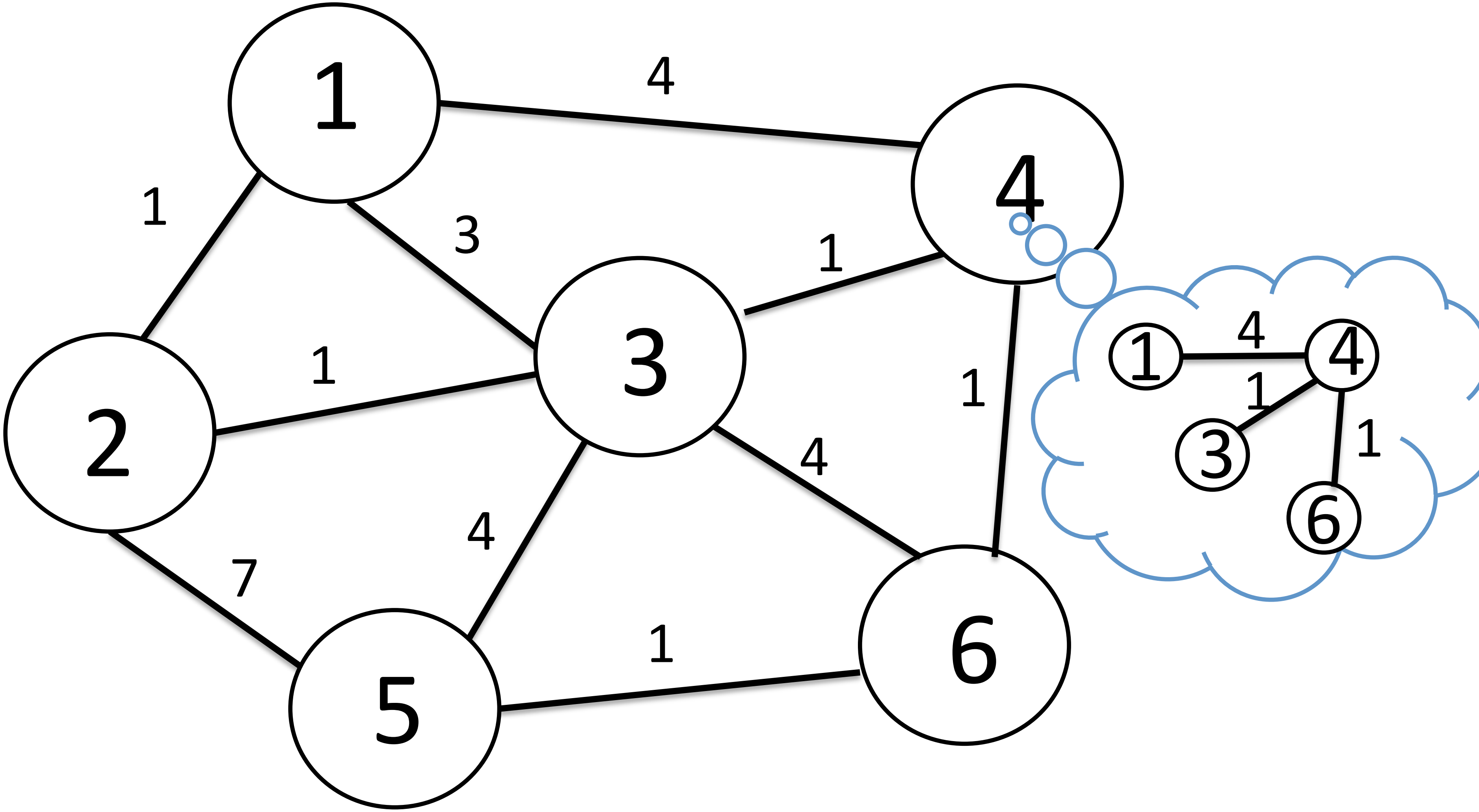
Part 1.1

CONGEST Model

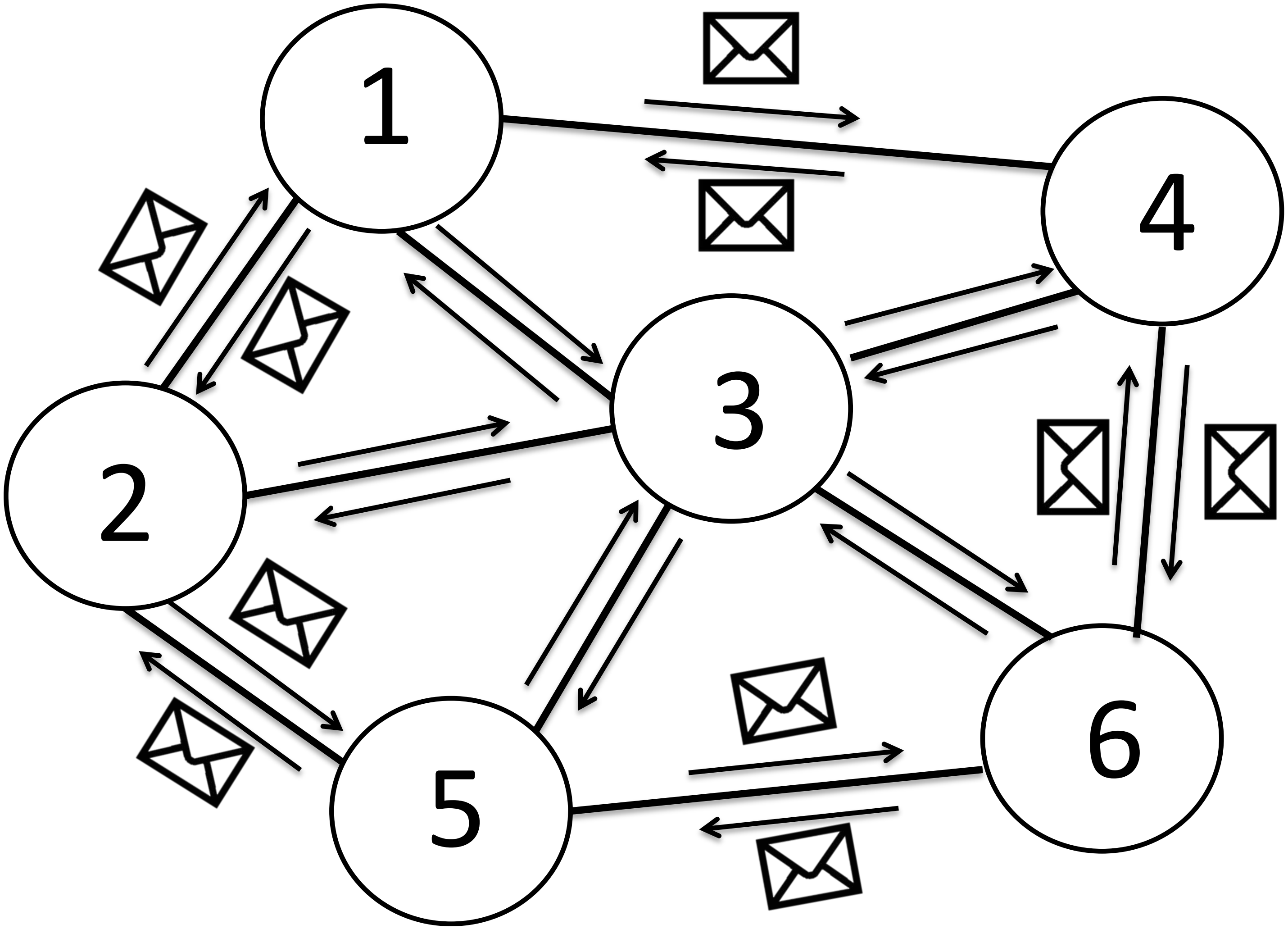


n=6
D=2

Network represented by a weighted graph **G** with **n** nodes and diameter **D**.



Nodes know only local information

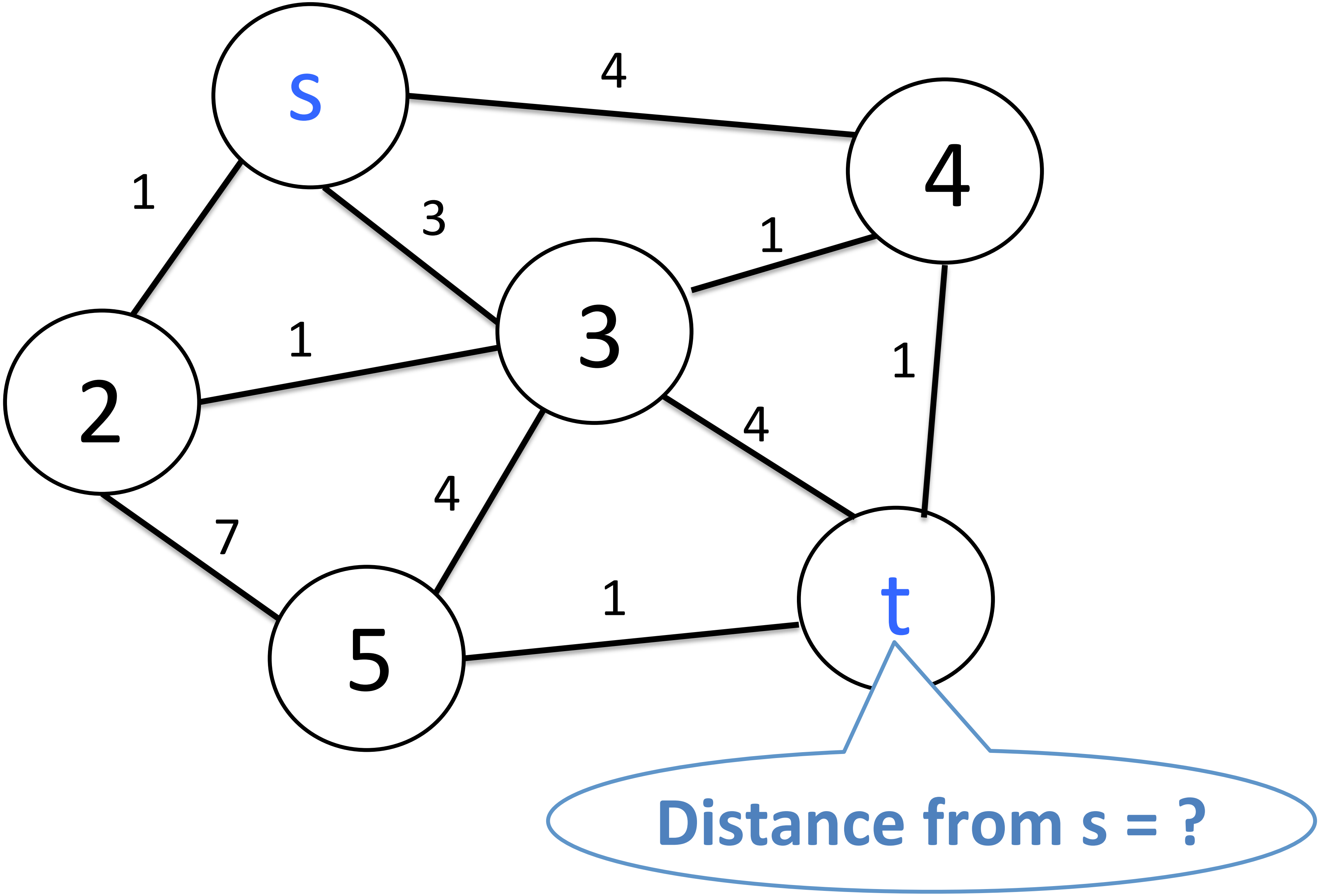


Nodes exchange O(log n) bits per round

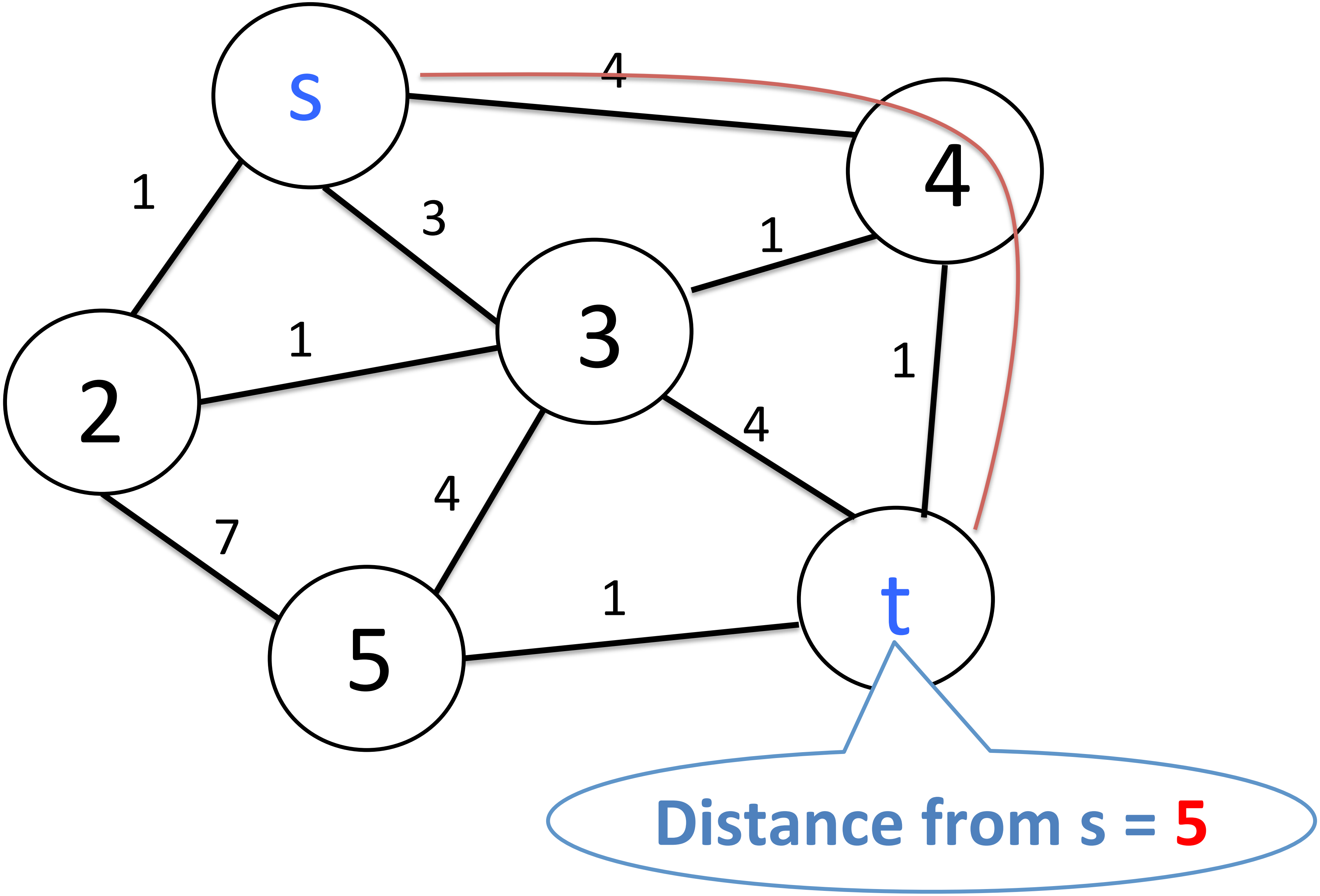
<u>Part 1.2</u>

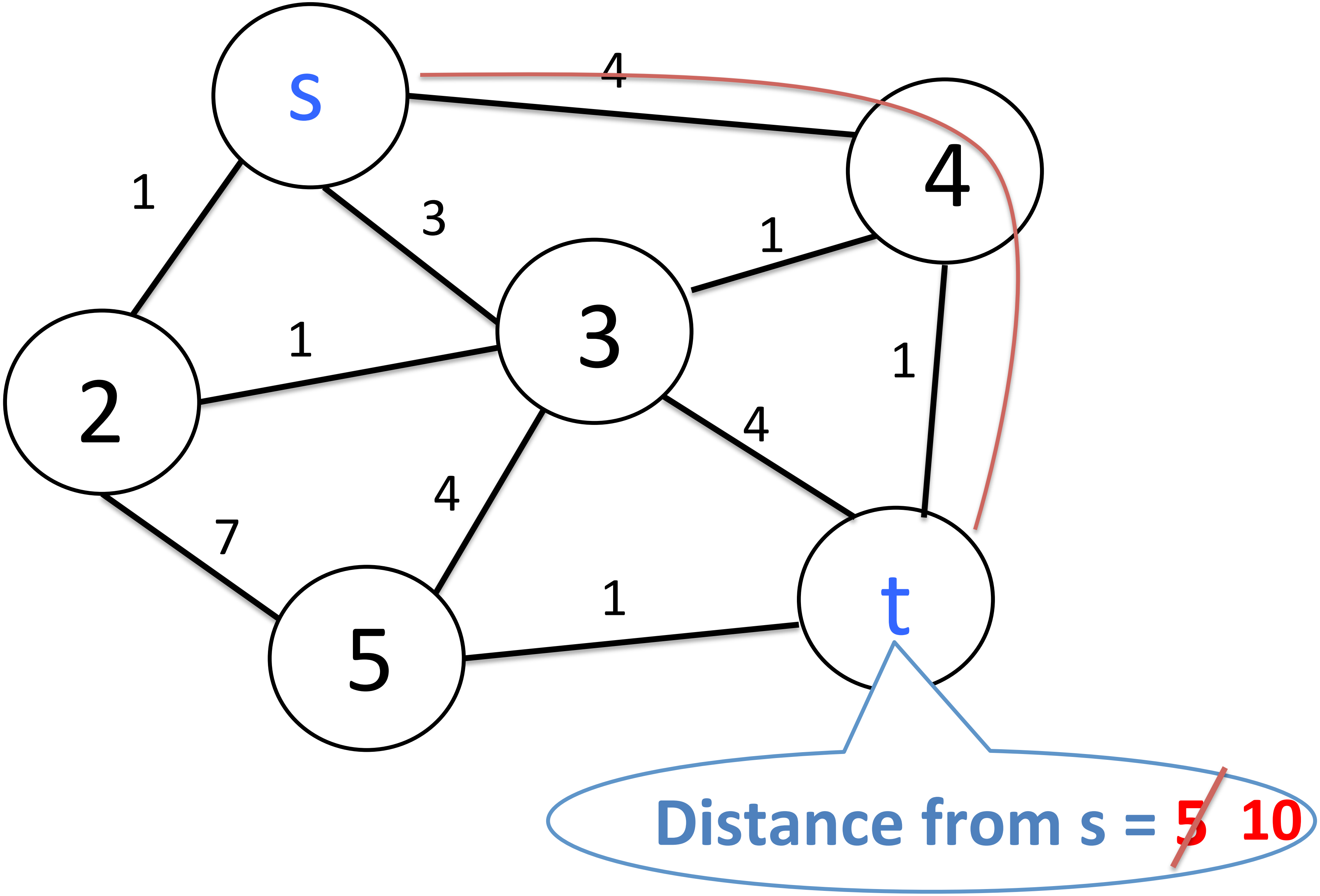
Example of problem:

s-t distance



Goal: Node t knows distance from s


s
4
4
1
3
1
3
2
1
1
4
4
7
1
5
t
Distance from s = 5



2-approximate solution

Computing s-t distance on **unweighted** graphs can be done in **O(D)** time by using the **Breadth-First Search (BFS)** algorithm.

There is an **Ω(D)** lower bound.

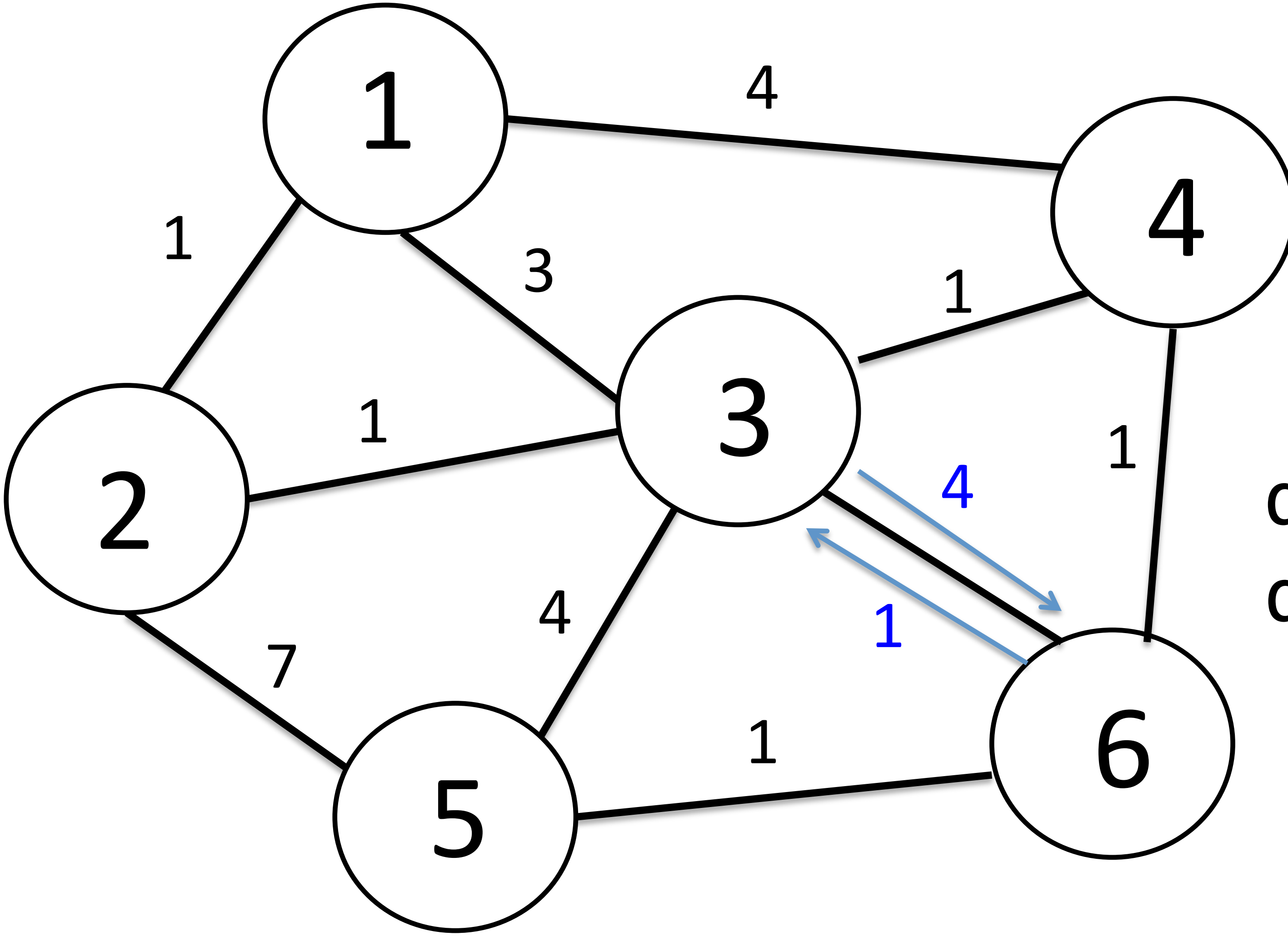
How about the weighted case?

Reference	Time	Approximation
Folklore	$\Omega(D)$	any
Bellman&Ford [1950s]	$O(n)$	exact
Elkin [STOC 2006]	$\Omega((n/\alpha)^{1/2} + D)$	any α
Das Sarma et al [STOC 2011]	$\Omega(n^{1/2} + D)$	any α
Lenzen,Patt-Shamir [STOC 2013]	$O(n^{1/2+1/2\alpha} + D)$	$O(\alpha)$
N [STOC 2014]	$O(n^{1/2}D^{1/4}+ D)$	$1+\varepsilon$
Henzinger,Krinninger,N	$O(n^{1/2+o(1)} + D^{1+o(1)})$	$1+\varepsilon$

- Polylog n factors are hidden
- Lenzen&Patt-Shamir actually achieve more than computing distances

Open Problems

- Sublinear-time **exact** algorithm?
  - Current: Nothing.
- $O(n^{1/2}\text{polylog } n+D)$-time $(1+\varepsilon)$-approx. algorithm?
  - Current: $O(n^{1/2+o(1)}+D^{1+o(1)})$-time
- Deterministic sublinear-time algorithm?
  - Current: Nothing
- Algorithm when weights are **asymmetric**?
  - Current: $O(n^{1/2}D^{1/2}+D)$-time



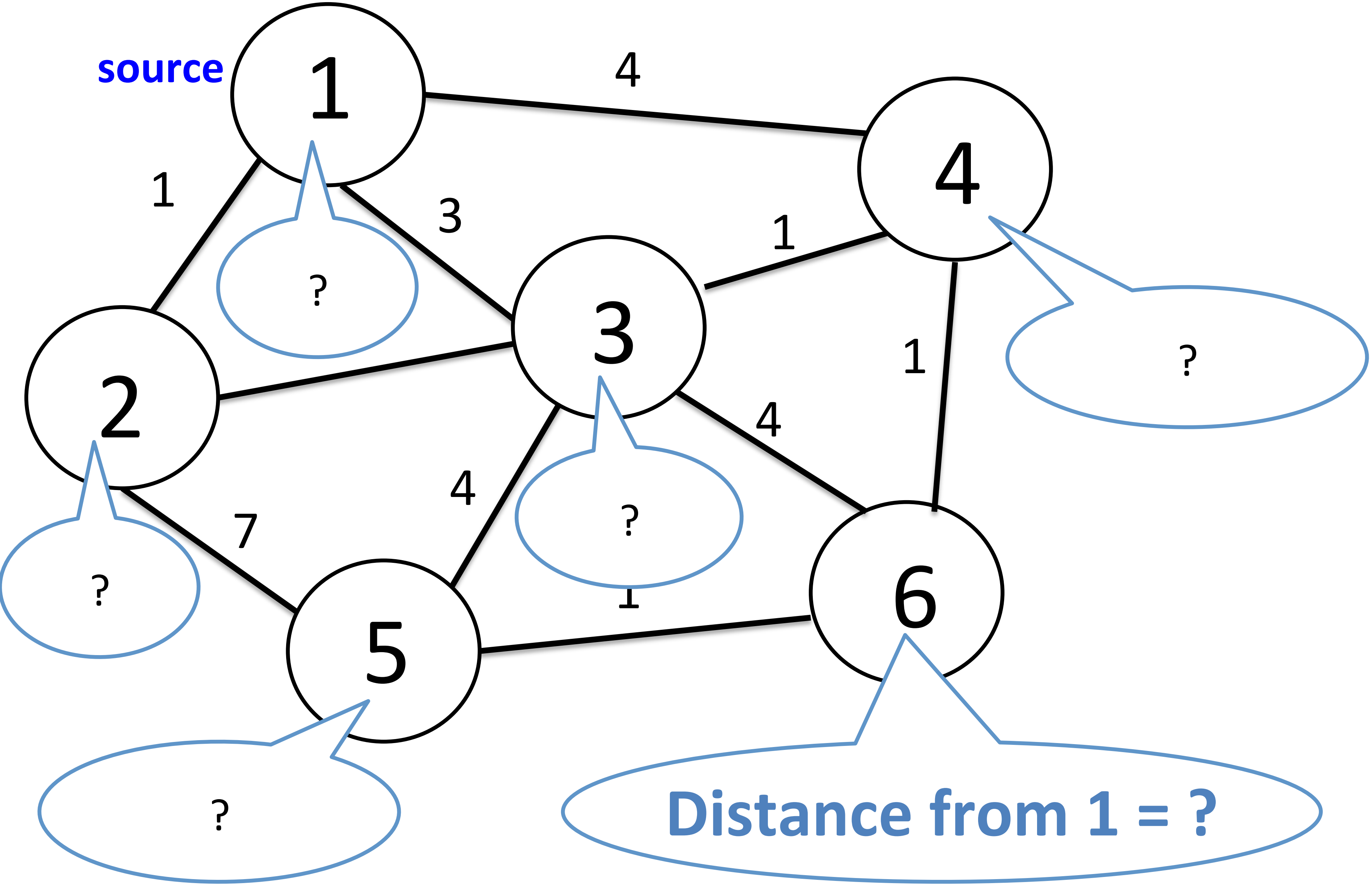
dist(6, 3)=1
dist(3, 6)=2

Asymmetric weight

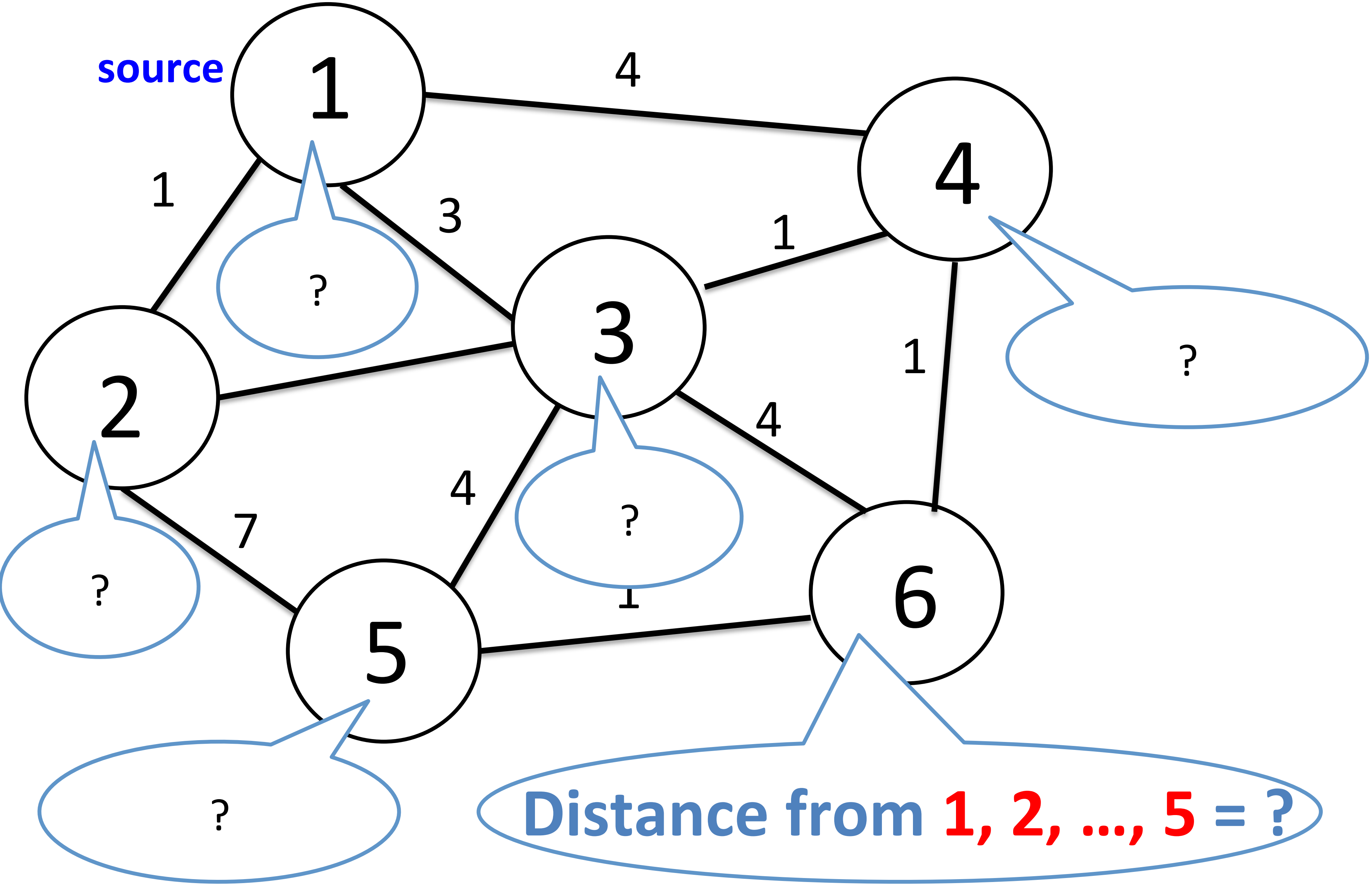
Note: Weights do not affect communication

Part 1.3

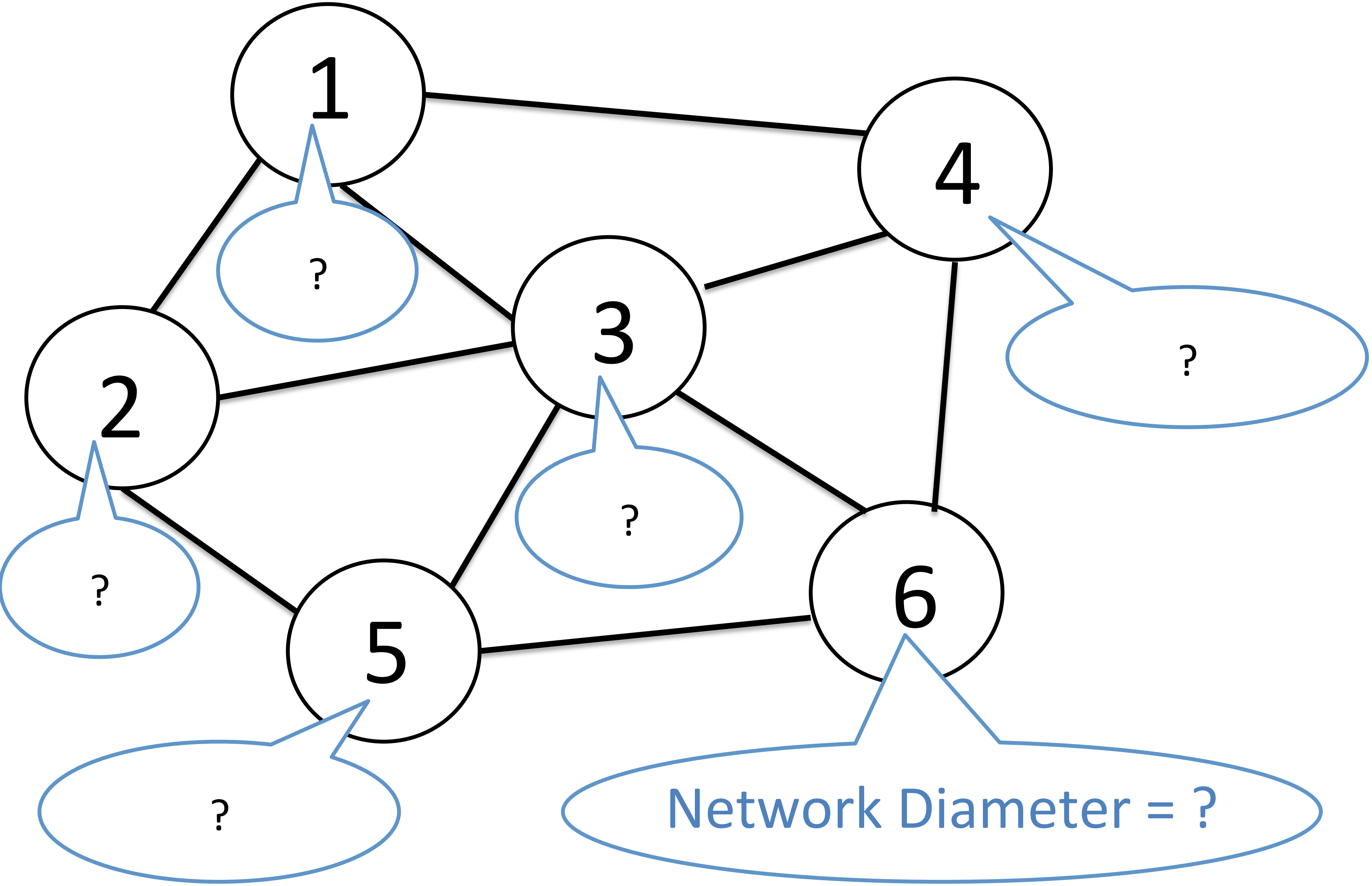
Related Problems



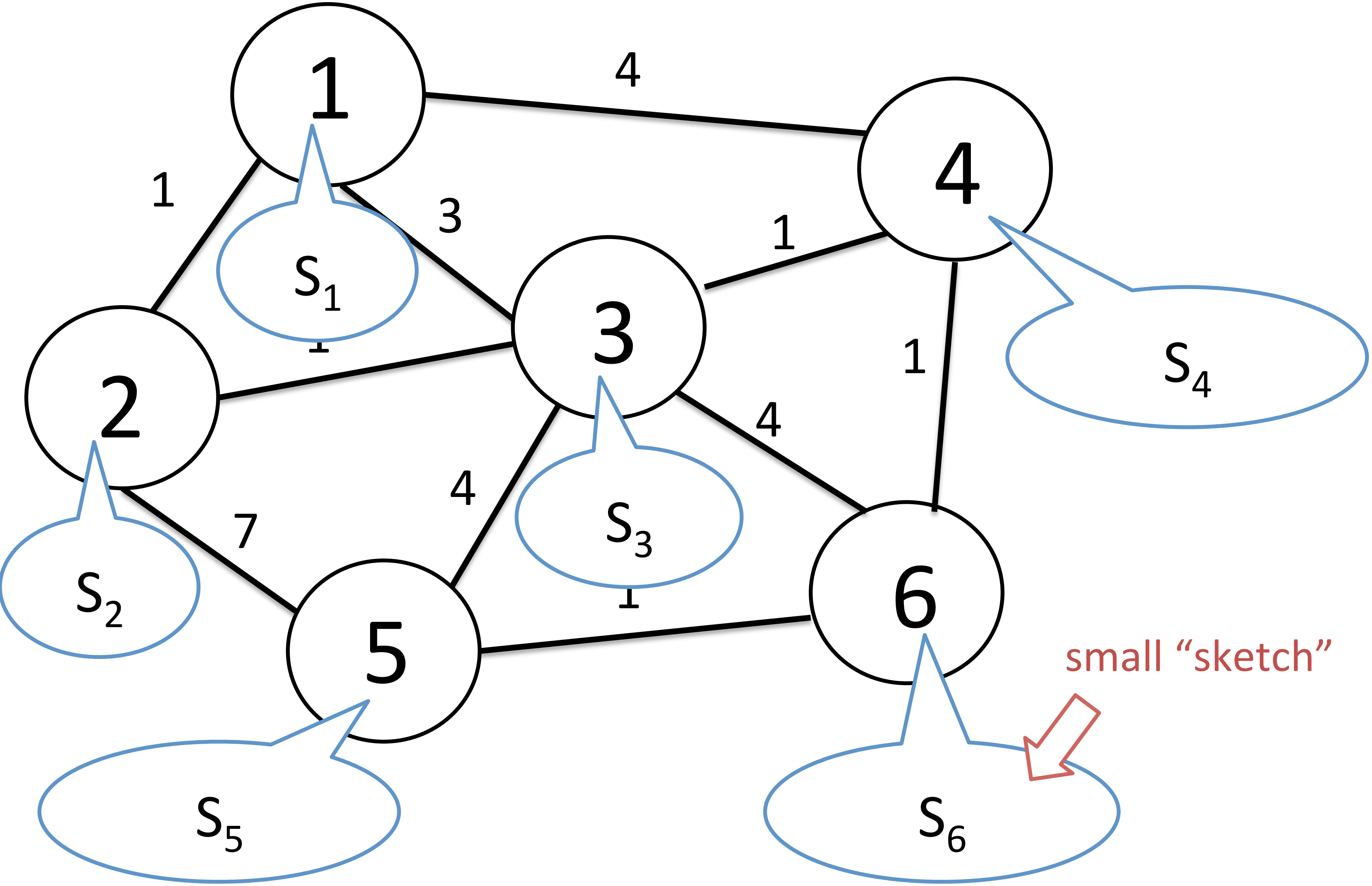
Single-source shortest paths



All-Pairs Shortest Paths



Diameter

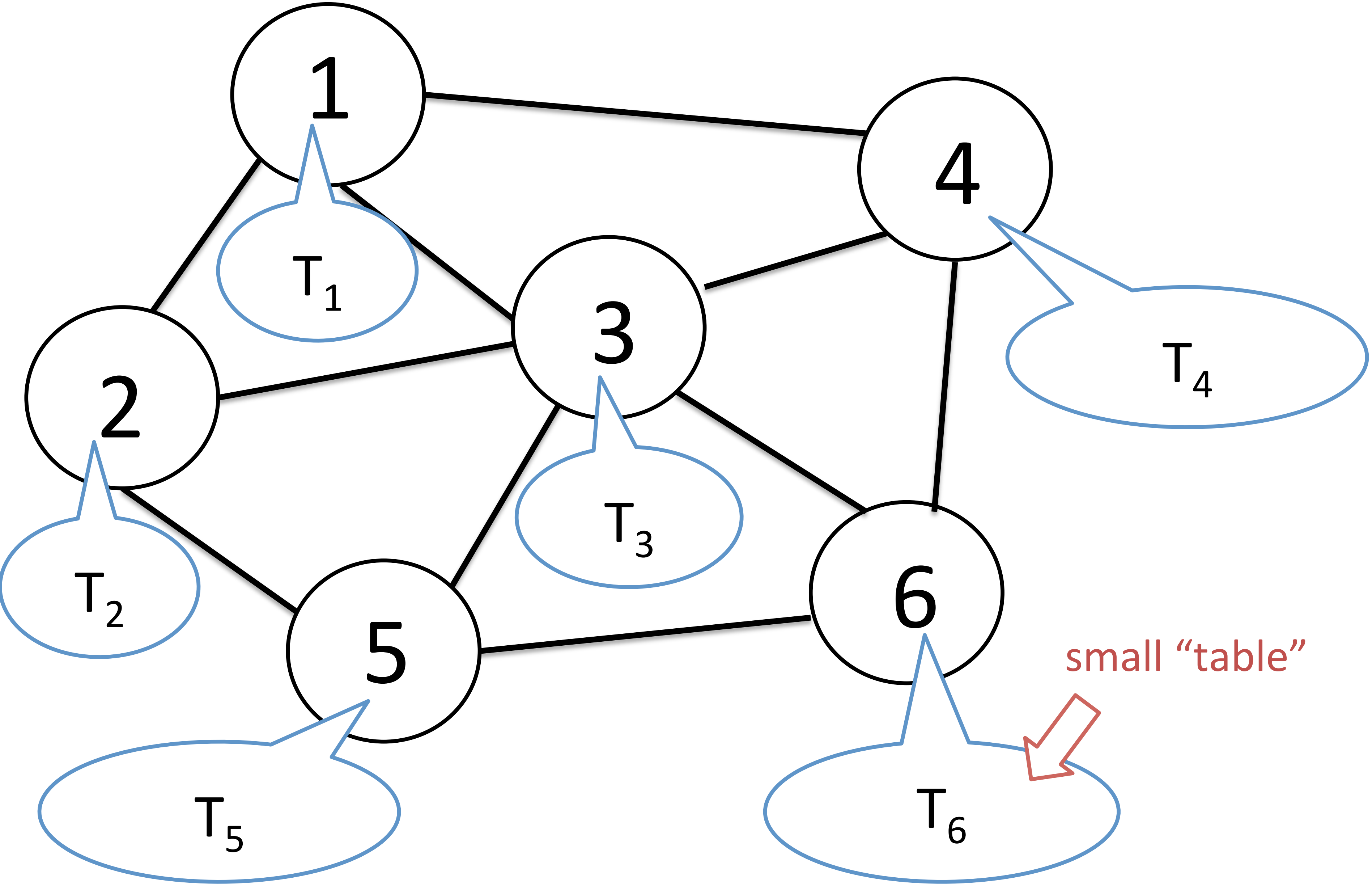


Distance labeling/sketching

Distance Labeling/Sketching

distance(u, v) can be approximated by "looking" at labels/sketch S_u and S_v

i.e., there is algorithm A that takes L_u and L_v and outputs an approximate value of dist(u, v)



Routing Table

Some Other Related Problems

SESSION V (Graph Distances and Routing)	
Session Chair: Dahlia Malkhi	
8:30 – 8:55 am	Merav Parter *Vertex Fault Tolerant Additive Spanners* **(best student paper award)**
8:55 – 9:20 am	Liam Roditty and Roei Tov *Close to Linear Space Routing Schemes*
9:20 – 9:45 am	Mohsen Ghaffari and Christoph Lenzen *Near-Optimal Distributed Tree Embedding*
9:45 – 9:52 am	Brief Announcement: Noy Rotbart, Søren Dahlgaard and Mathias Bæk Tejs Knudsen *On Dynamic and Multi-Functional Labeling Schemes*
9:52 – 9:59 am	Brief Announcement: Matthieu Perrin, Achour Mostéfaoui and Claude Jard *Update Consistency in Partitionable Systems*

Also Session XI: Diameter [Holzer et al.]

Some Results

Single-source shortest paths

-- same as s-t distance --

Reference	Time	Approximation
Das Sarma et al [STOC 2011]	$\Omega(n^{1/2} + D)$	any α
Henzinger,Krinninger,N	$O(n^{1/2+o(1)} + D^{1+o(1)})$	$1+\varepsilon$
open	$O(n^{1/2} + D)$	$1+\varepsilon$
open	sublinear	exact

Also open: Deterministic algorithm and Asymmetric case.

Open Problems

- Sublinear-time **exact** algorithm?
  - Current: Nothing.
- $O(n^{1/2}$polylog $n+D)$-time $(1+\varepsilon)$-approx. algorithm?
  - Current: $O(n^{1/2+o(1)}+D^{1+o(1)})$-time
- Deterministic sublinear-time algorithm?
  - Current: Nothing
- Algorithm when weights are **asymmetric**?
  - Current: $O(n^{1/2}D^{1/2}+D)$-time

All-pairs shortest paths

Algorithm	**Time**	**Approximation**
Lower bound (Lenzen,Patt-Shamir [STOC'13] &N [STOC'14])	$\Omega(n)$	any
Lenzen,Patt-Shamir [STOC'13]	$O(n)$	$O(1)$
N [STOC 2014]	$O(n)$	$1+o(1)$
N	$O(n^{3/2})$	exact
Open	$O(n)$	exact

Distance labeling/sketching

Algorithm	**Time**	**Approximation**	**Size**
Das Sarma et al. [SPAA'12]	$O(n^{1/k}\mathit{SPD})$	2k-1	$kn^{1/k}$
Lenzen&Patt-Shamir [STOC'13]	$O(n^{1/2+1/2k} + D)$	$O(k^2)$	$n^{1/2k}$
Open	$O(n^{1/2+1/k} + D)$	2k-1	$kn^{1/k}$

SPD = shortest path diameter (could be as large as n)

*Definition used by Lenzen and Patt-Shamir is slightly different

Diameter (unweighted)

Algorithm	**Time**	**Approximation**
BFS	D	2
Holzer et al. [PODC'12]	$\Omega(n)$	$3/2-\varepsilon$
Holzer et al. [PODC'12] Peleg et al. [ICALP'12]	$O(n)$	exact
Frischknecht et al. [SODA'12]	$\Omega((n/D)^{1/2}+D)$	$3/2-\varepsilon$
Lenzen-Peleg [PODC'13]	$O(n^{1/2}+D)$	$3/2$
Holzer et al. [DISC'14]	$O((n/D)^{1/2}+D)$	$3/2+\varepsilon$
Open	?	?

Diameter (weighted)

Algorithm	**Time**	**Approximation**
Holzer et al. [PODC'12]	$\Omega(n)$	$2-\varepsilon$
Henzinger,Krinninger,N	$O(n^{1/2+o(1)} + D^{1+o(1)})$	$2+\varepsilon$
Open	$O(n^{1/2} + D)$	$2+\varepsilon$
Open	sublinear	2

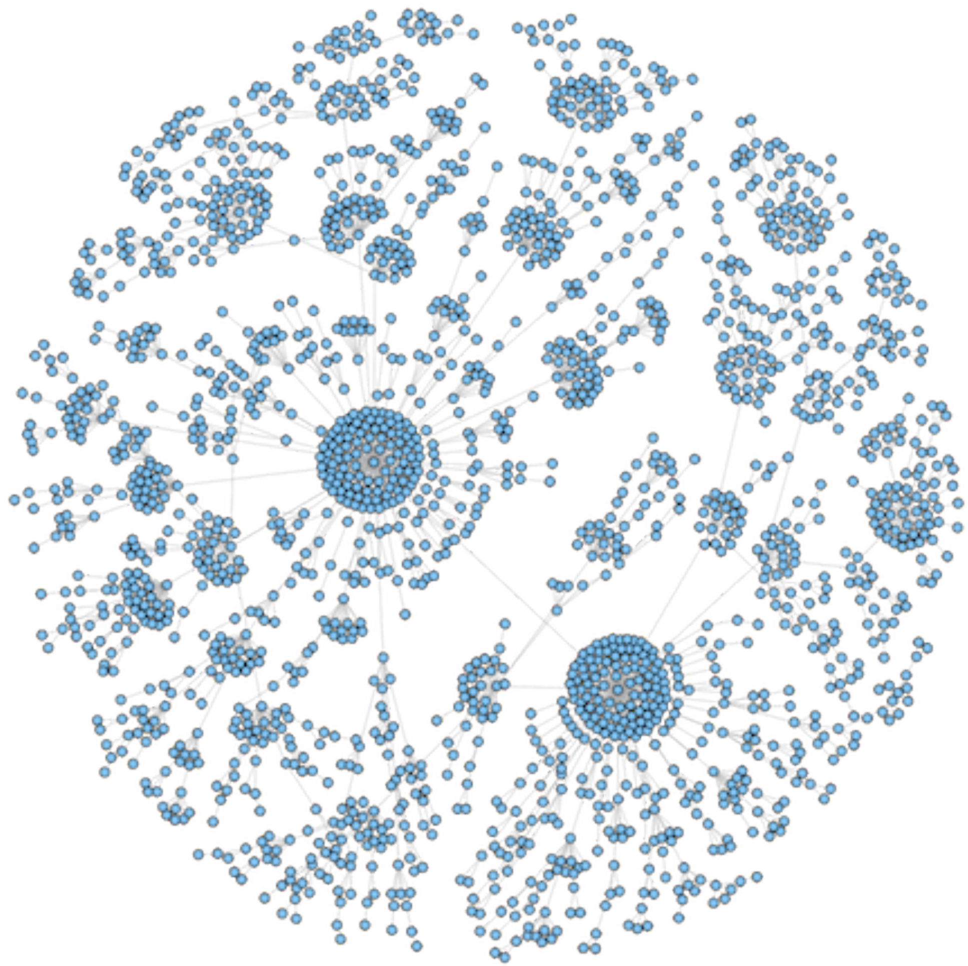
(Getting a sublinear-time exact algorithm for SSSP will resolve this)

Part 2

Algorithmic Techniques

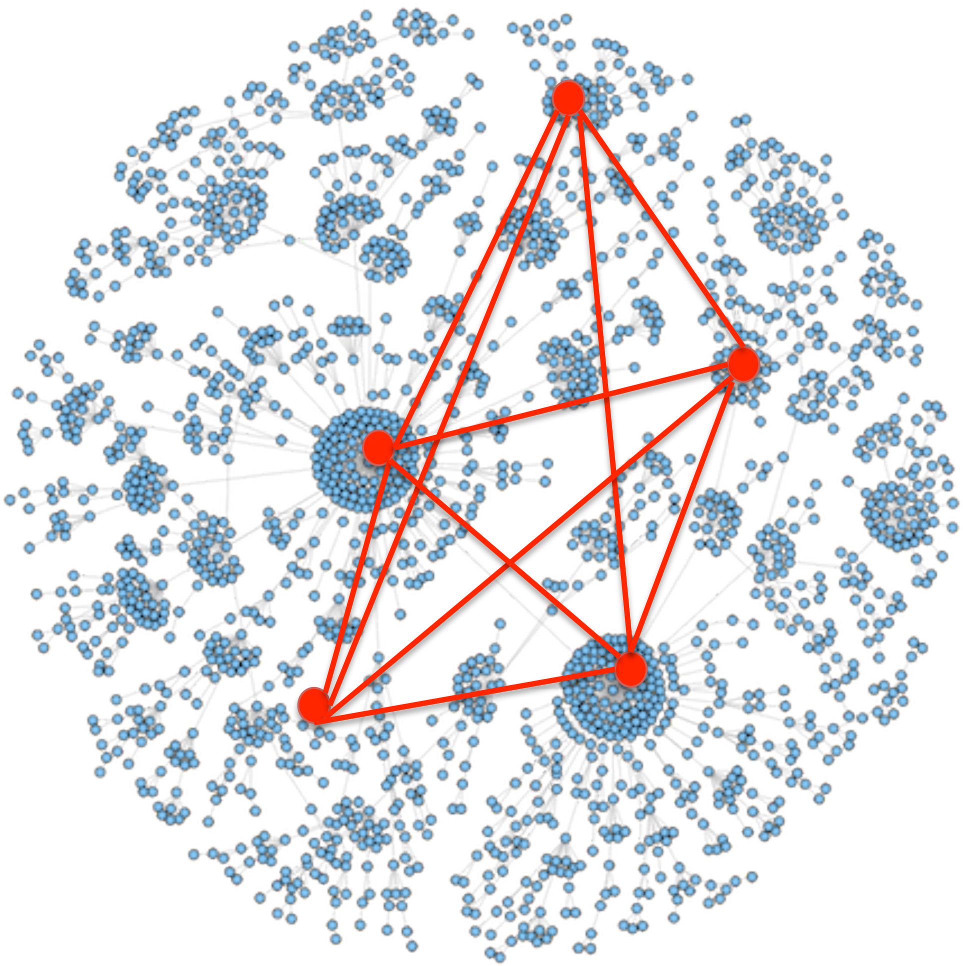
A Common Framework

A Common Framework



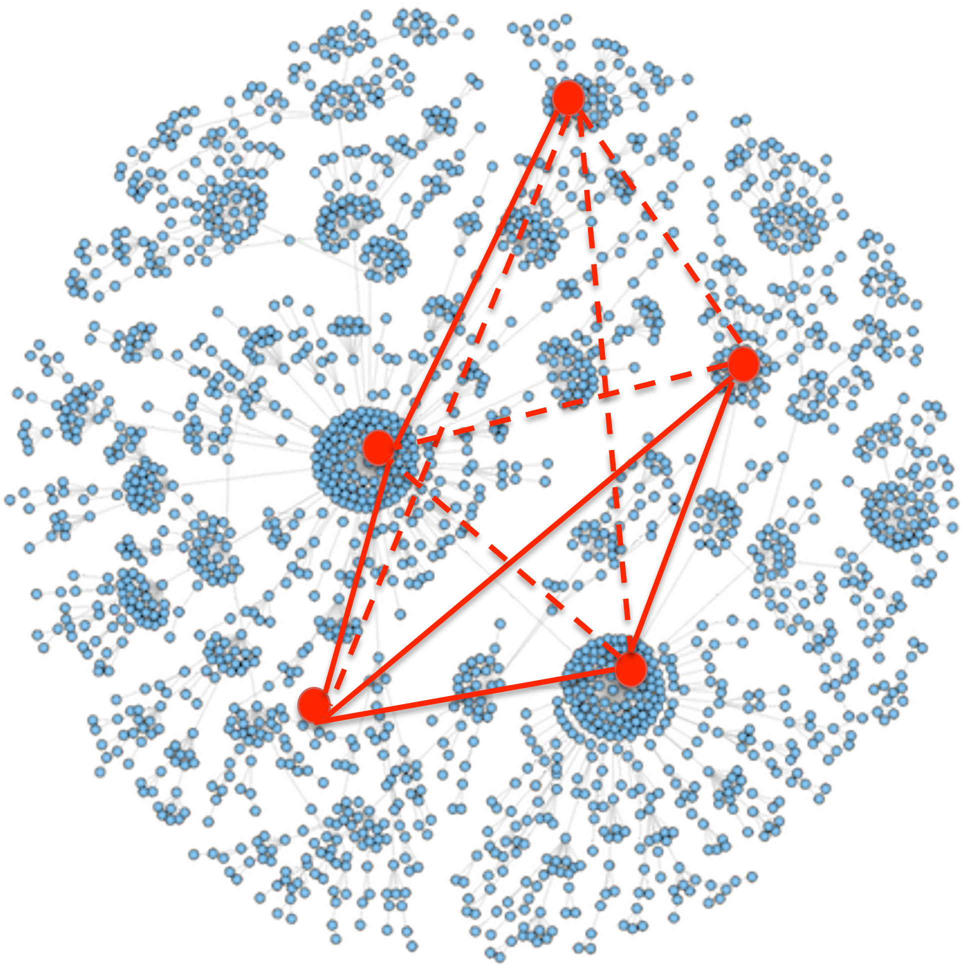
1. Input graph

A Common Framework



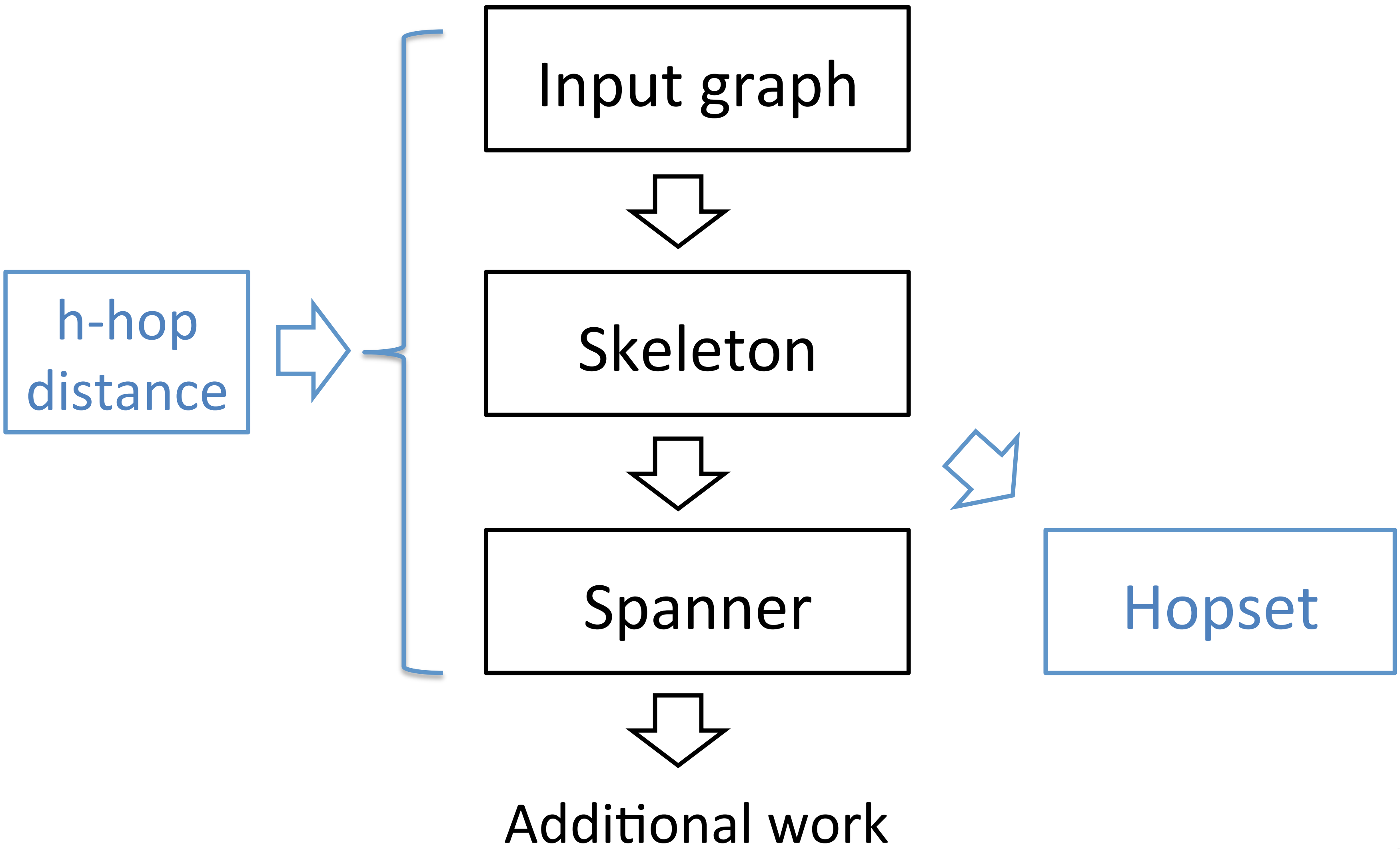
2. Skeleton

A Common Framework

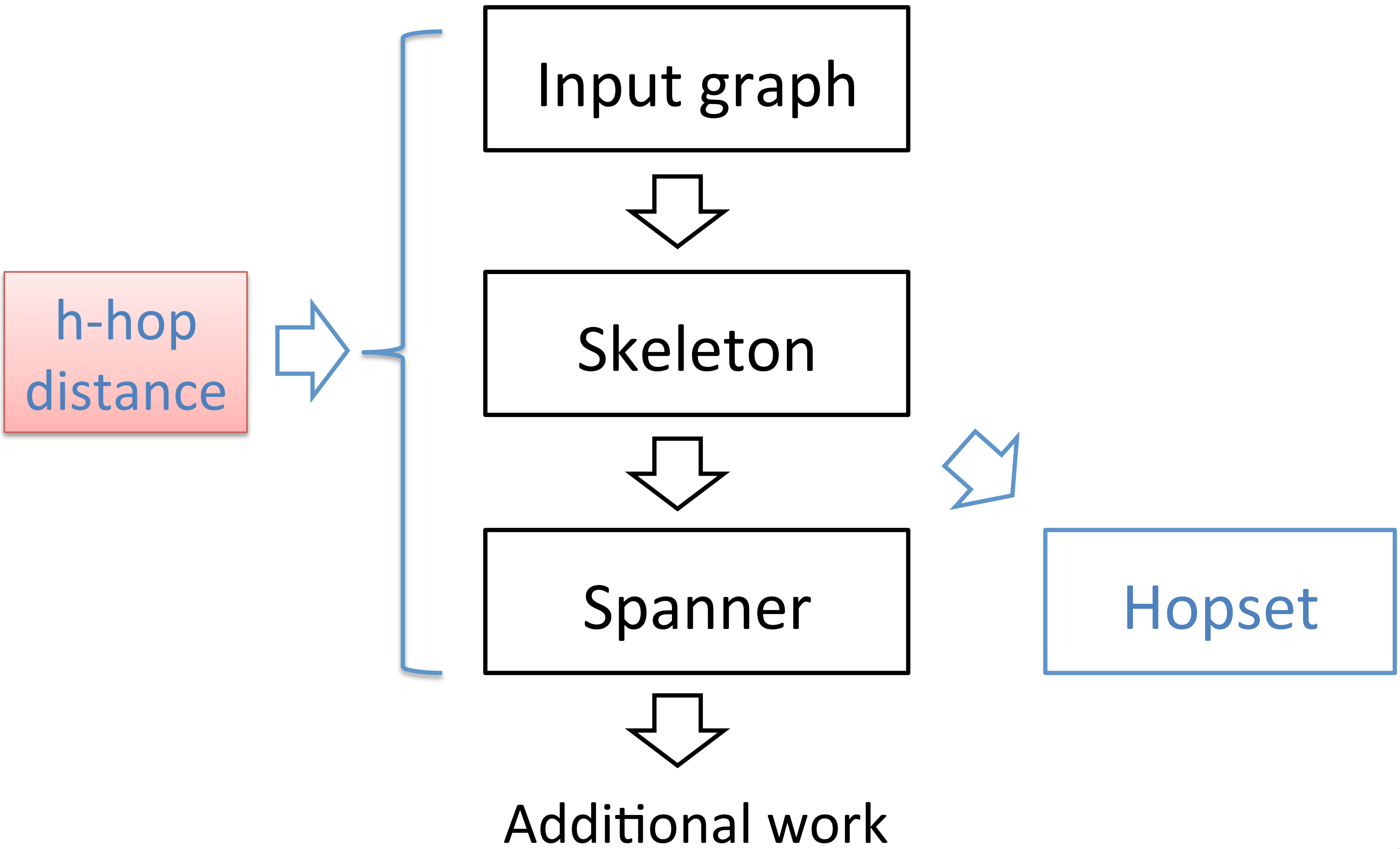


3. Sparse spanner/emulator of skeleton

A Common Framework

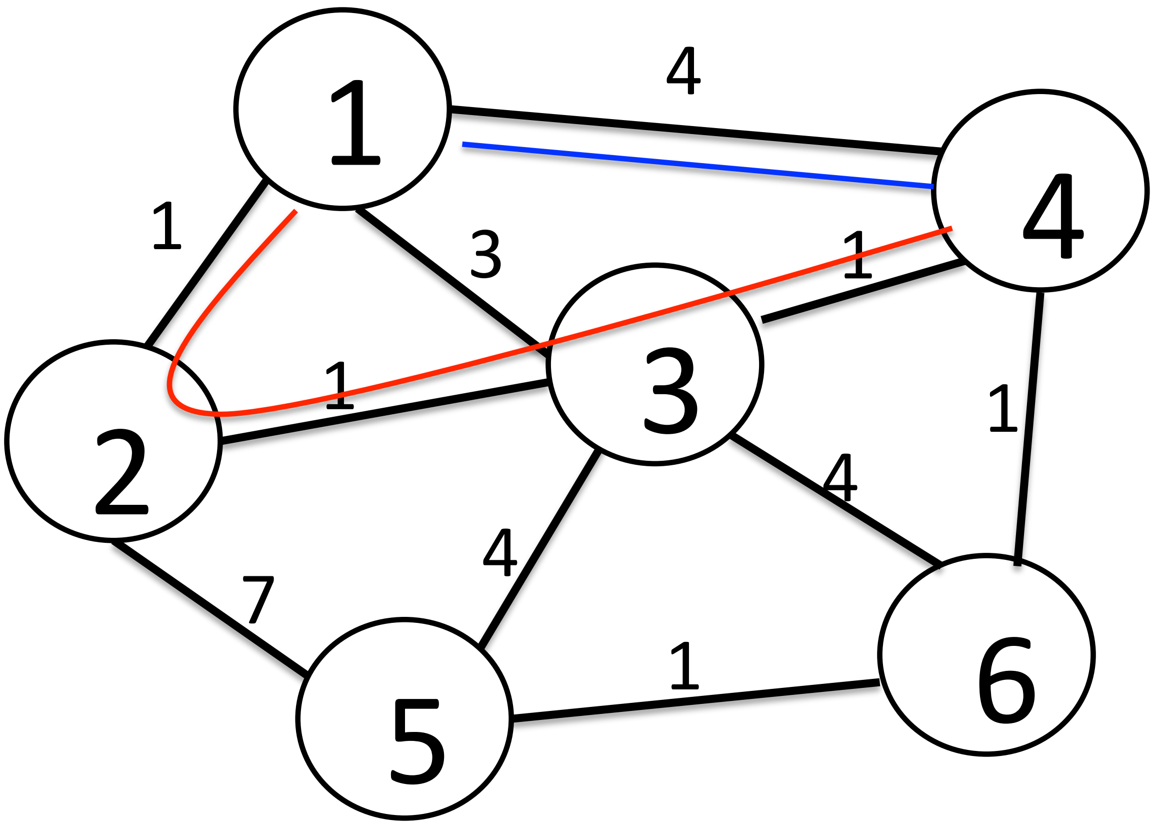


A Common Framework



Definition: h-hop distance

- $dist^h(u,v)$:= smallest total weight among u-v paths containing at most h edges



dist(1, 6) = 3

$dist^1(1, 6) = 4$

Computing single-source h-hop distances in the weighted case is as easy as computing a BFS tree on unweighted graphs

(With the cost of $(1+\varepsilon)$-multiplicative error)

Theorem 1

We can find single-source $(1+\varepsilon)$-approx. "h-hop distances" in $O(h/\varepsilon)$ time

“Light-Weight” Feature

Can be parallelized efficiently

Two techniques to paralellize BFS trees

1. Random delay -- N [STOC’14]
2. Deterministic scheduling – Holzer et al. [PODC’12]

Theorem 2

We can find k-sources $(1+\varepsilon)$-approx. h-hop distances in $O(k+h/\varepsilon)$ time

Some Technical Details

Suppose that we are finding distance(s, t)

Suppose that we are finding distance(s, t)

and know that for some h and W

Suppose that we are finding distance(s, t)

and know that for some h and W

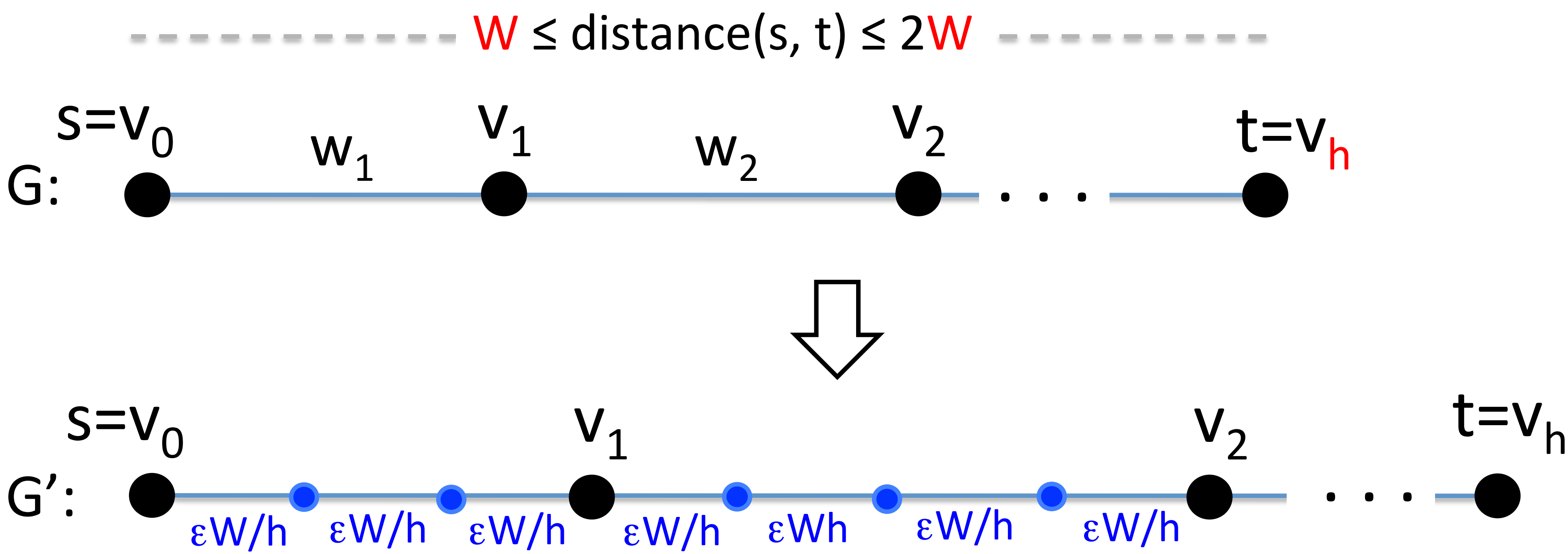
1. The shortest s-t path has h hops

Suppose that we are finding distance(s, t)

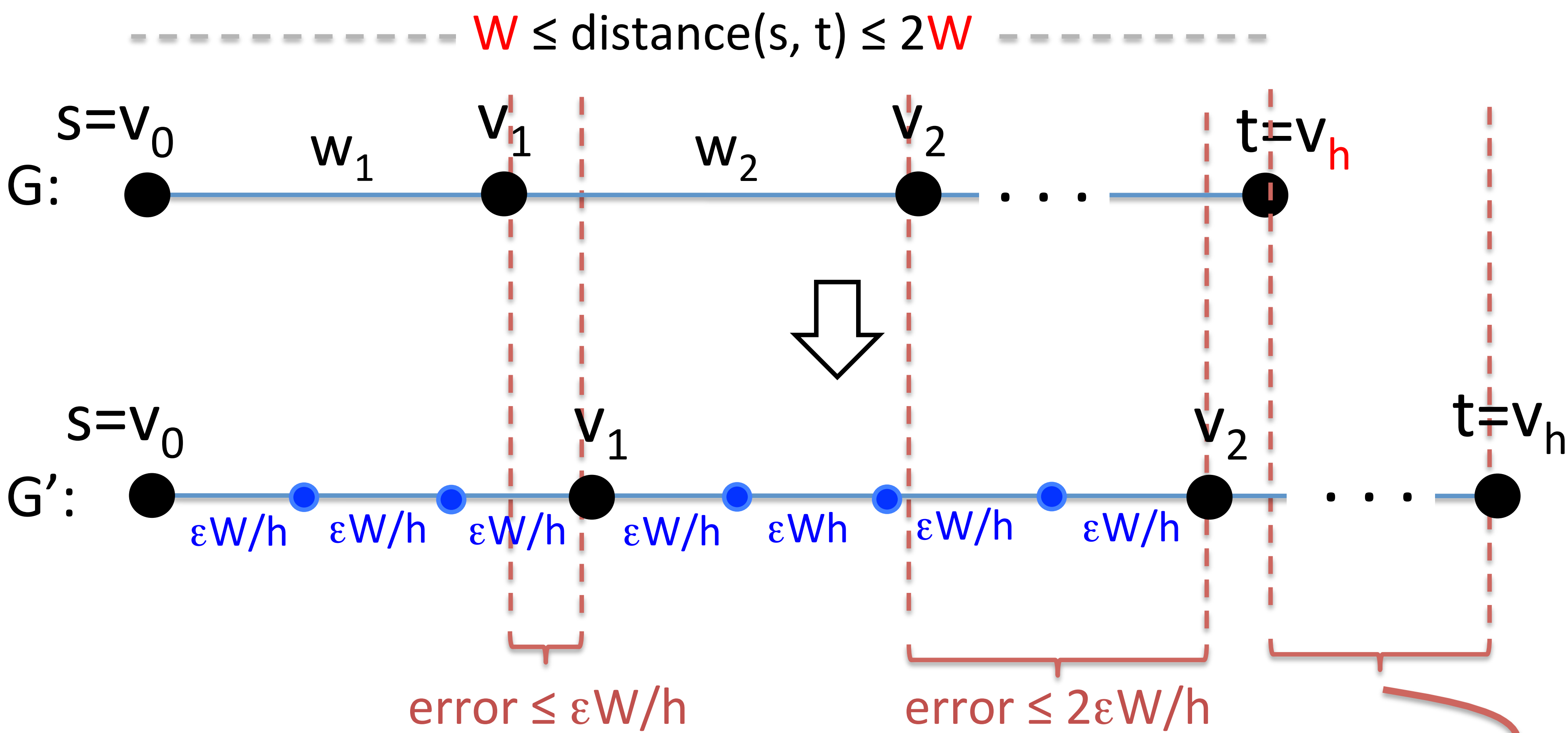
and know that for some h and W

1. The shortest s-t path has h hops
2. W ≤ distance(s, t) ≤ 2W

Step 1: Round weights up to a multiple of $\varepsilon W/h$

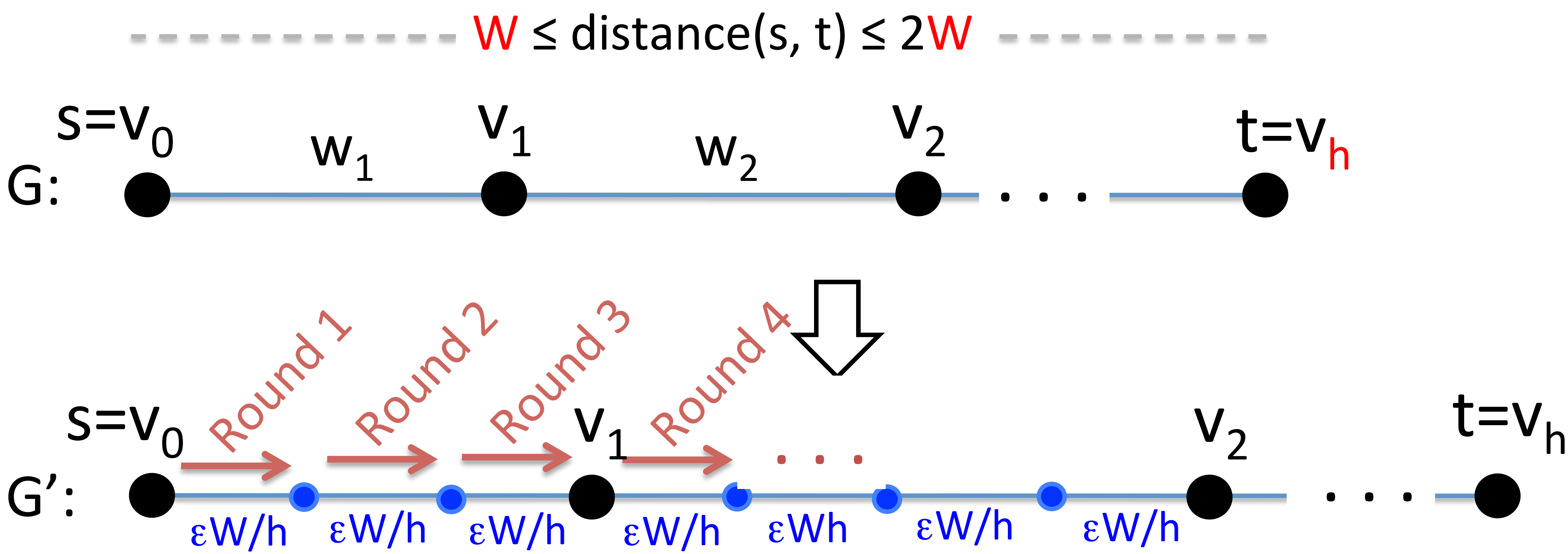


Claim: G' gives $(1+\varepsilon)$-approximate distance



Total error $\leq h(\varepsilon W/h) \leq \varepsilon \text{dist}(s, t)$

Step 2: Run BFS algorithm on G'



Number of rounds = $O(\text{dist}(s,t)/(\varepsilon W/h))$

$= O(h/\varepsilon)$

Summary

We can "pretend" that the graph is unweighted by losing a $(1+\varepsilon)$-approximation factor

Theorem

We can find k-sources $(1+\varepsilon)$-approx. h-hop distances in $O(k+h/\varepsilon)$ time

Caution: The new unweighted graph may have diameter more than D!

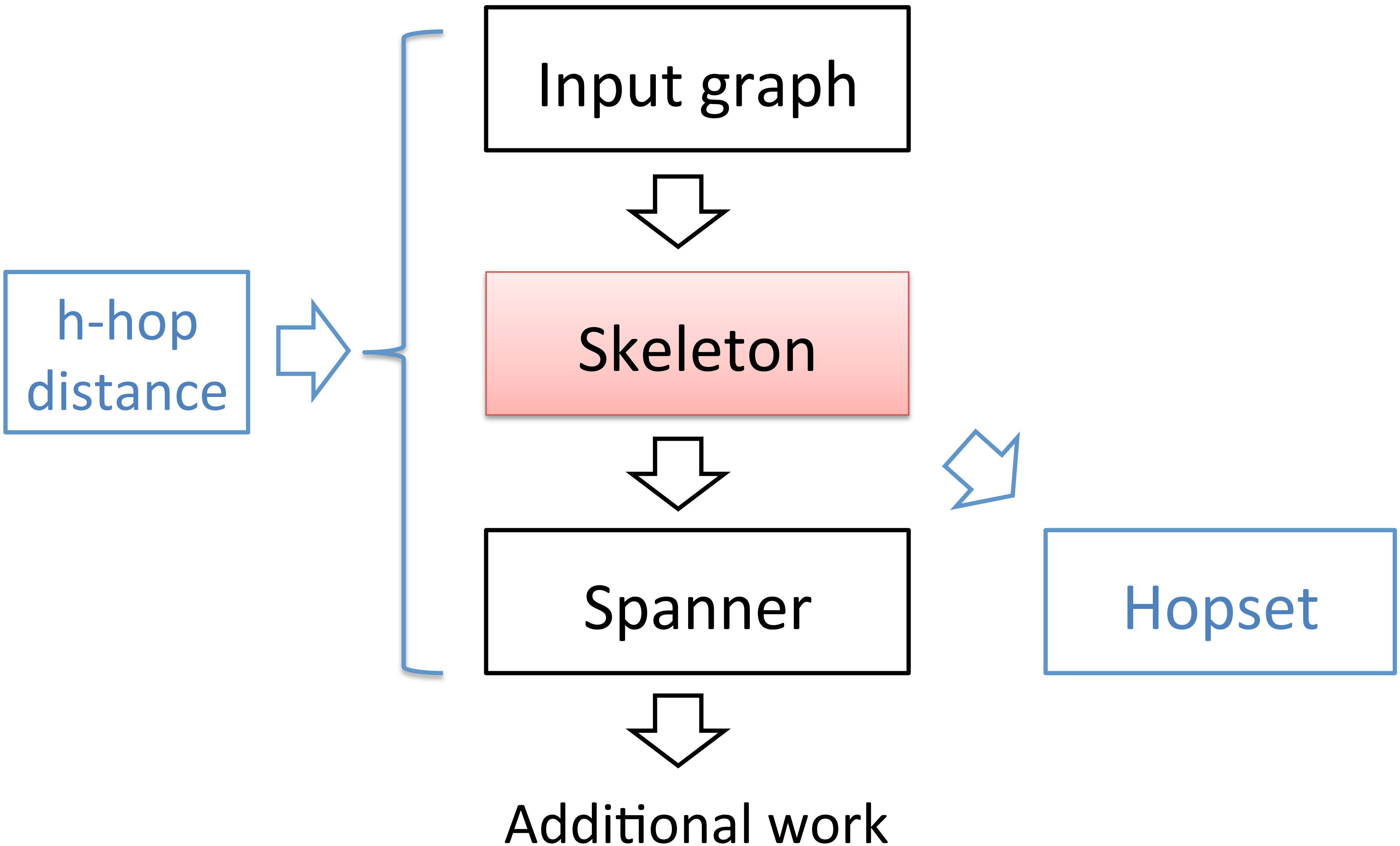
Corollary

We can compute $(1+\varepsilon)$-approx. all-pairs distances in $O(n/\varepsilon)$ time

Open problem

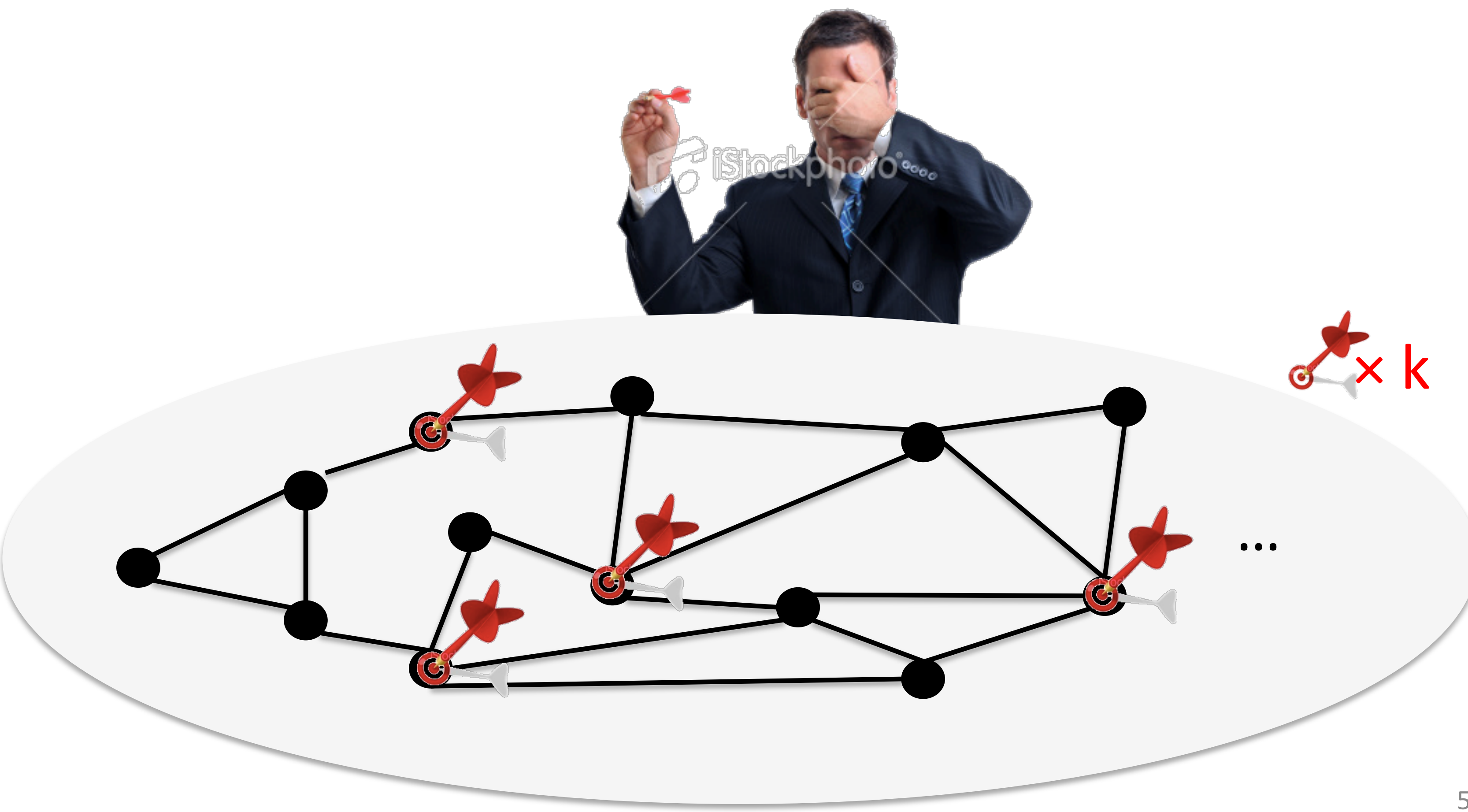
- Can we find k-sources ~~$(1+\varepsilon)$-approx.~~ **exact** h-hop distances in $O(k+h)$ time?
- If so, we will be able to solve APSP exactly in linear time.
- We will also be able to solve SSSP exactly in sublinear time.

Common Framework



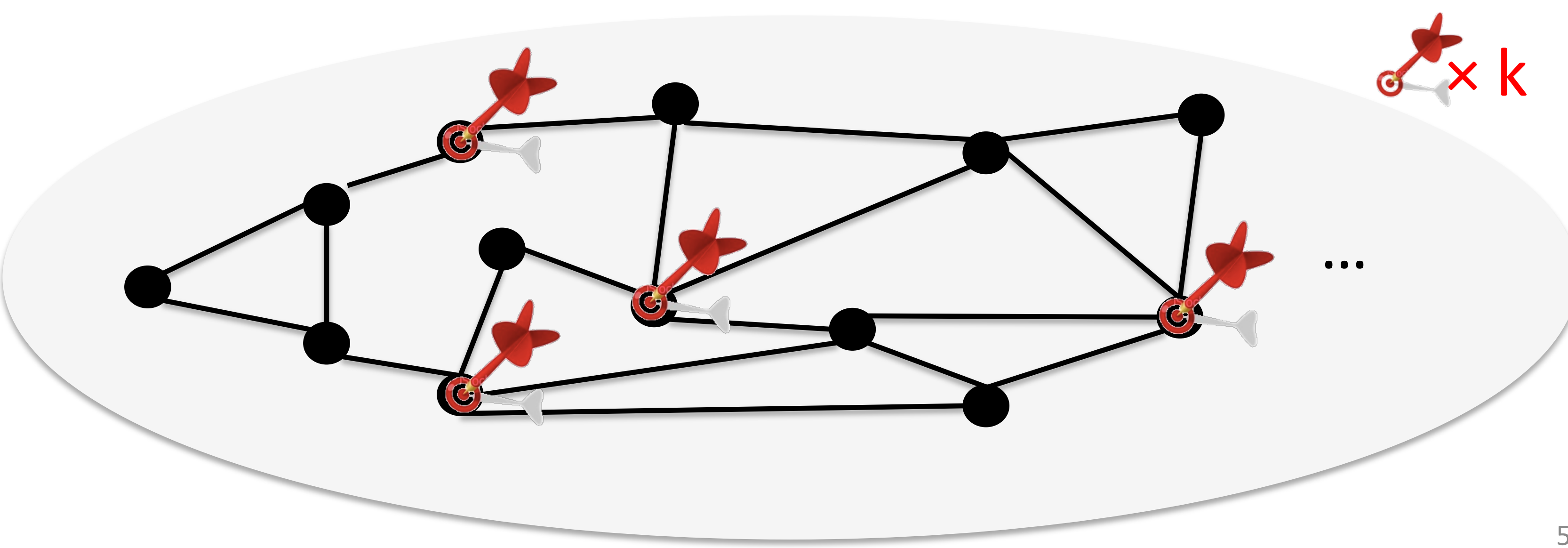
k-skeleton

1. Randomly pick k (≈$n^{1/2}$) nodes



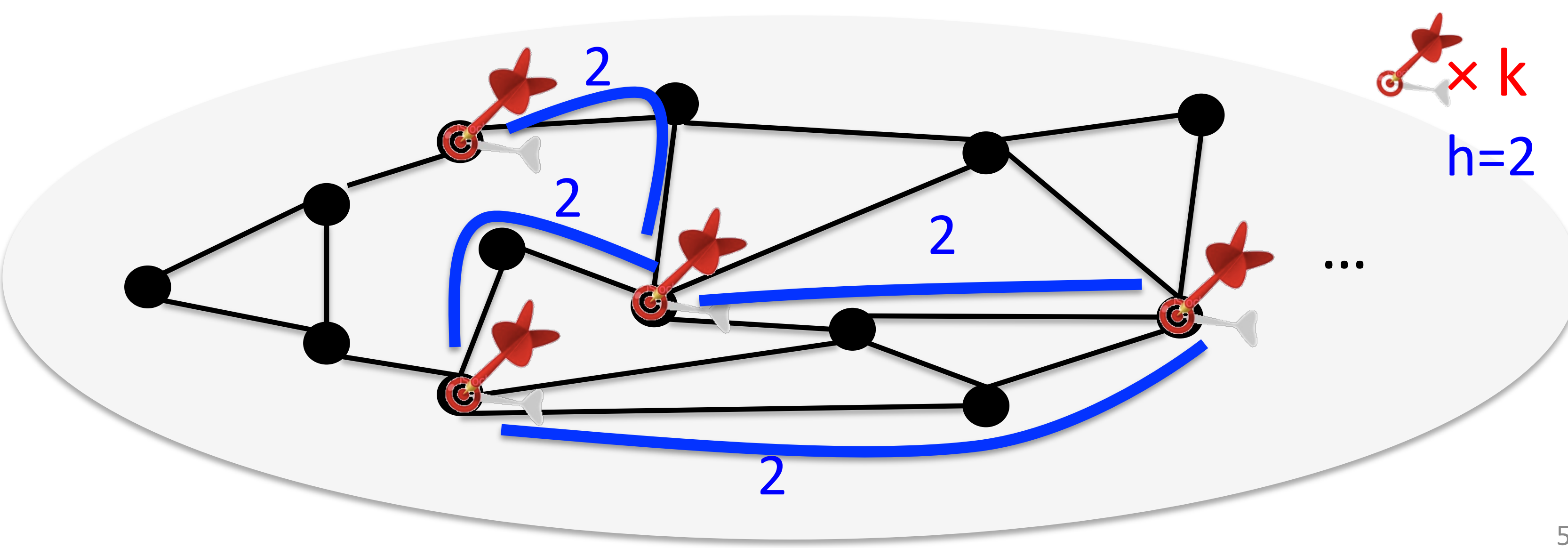
k-skeleton

1. Randomly pick k ($\approx n^{1/2}$) nodes
2. Compute h-hop distance between random nodes where $h = n/k$ ($\approx n^{1/2}$)



k-skeleton

1. Randomly pick k $(\approx n^{1/2})$ nodes
2. Compute h-hop distance between random nodes where $h=n/k$ $(\approx n^{1/2})$
3. Add "virtual edges" between random nodes. Weight = h-hop distance.



Constructing an approximate skeleton

Recall: We can find k-sources $(1+\varepsilon)$-approx. h hop distances in $O(k+h/\varepsilon)$ time.

Corollary: We can compute a k-skeleton in $O(k+n/k\varepsilon)$ time with $(1+\varepsilon)$-approximate distances between random nodes.

Consequence to s-t distance computation

Lemma: For every pair of nodes u and v on a skeleton H of graph G,

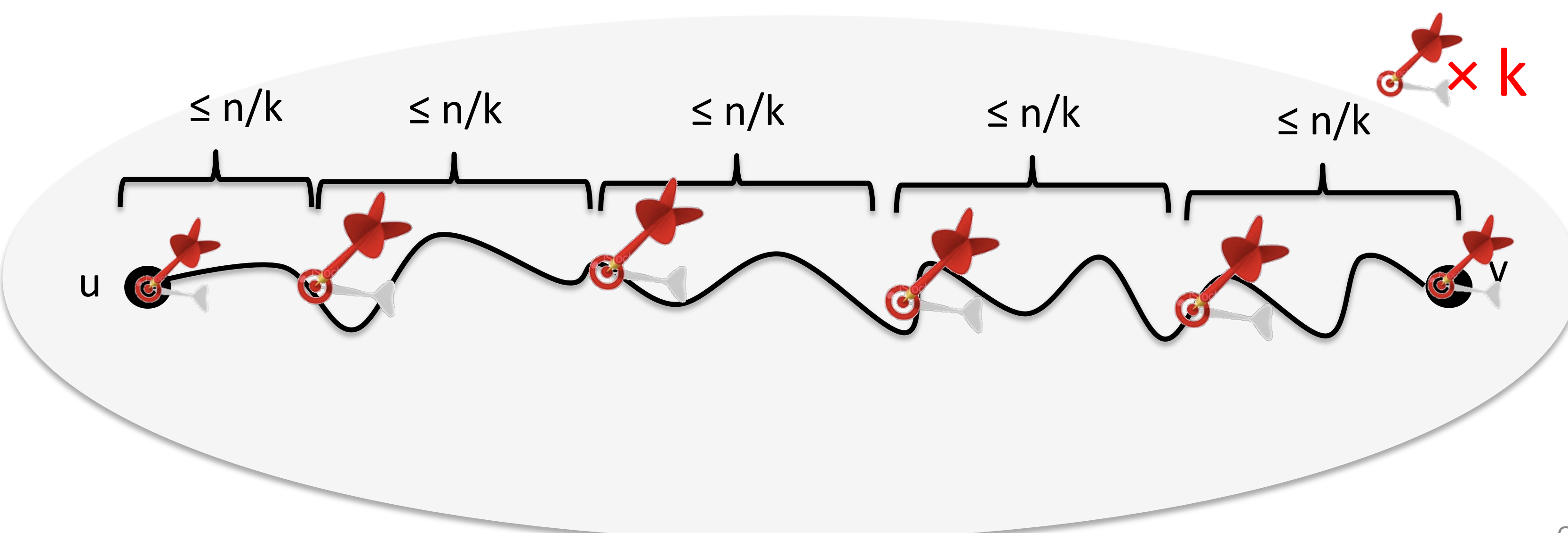
$$dist_G(u, v) = dist_H(u,v)$$

Consequence to s-t distance computation

Lemma: For every pair of nodes u and v on a skeleton H of graph G,

$$dist_G(u, v) = dist_H(u,v)$$

10-second proof: k random nodes will split u-v path into subpaths of length at most n/k

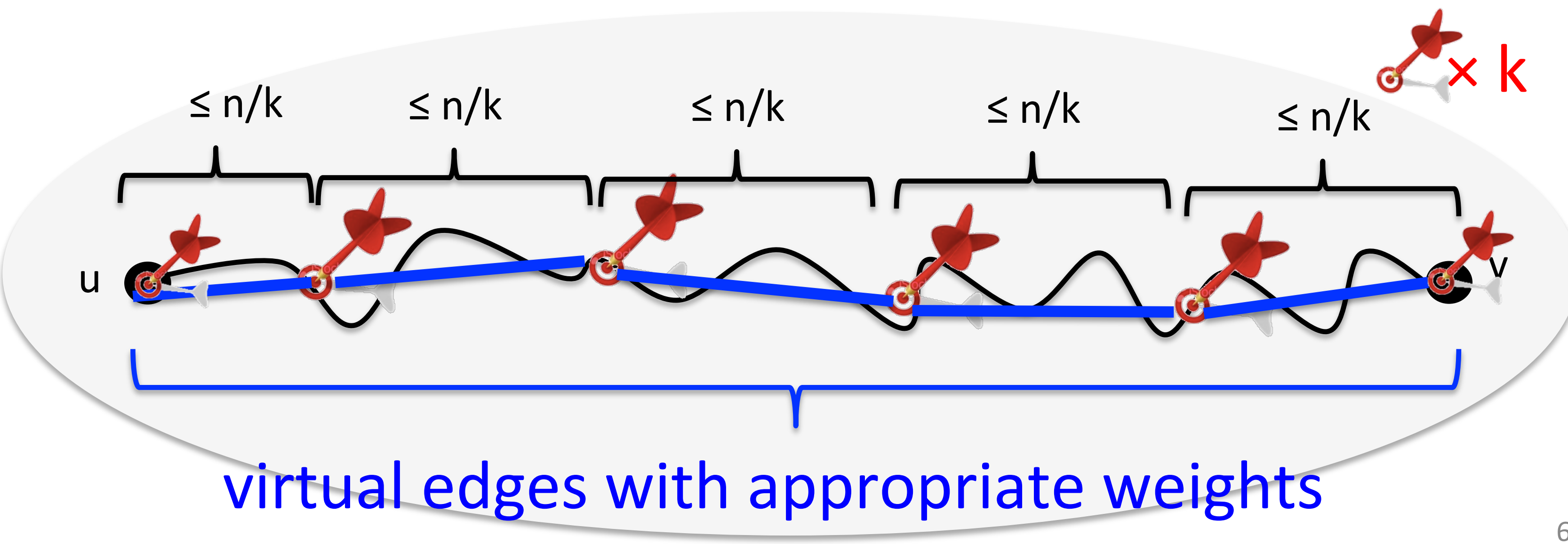


Consequence to s-t distance computation

Lemma: For every pair of nodes u and v on a skeleton H of graph G,

$$dist_G(u, v)=dist_H(u,v)$$

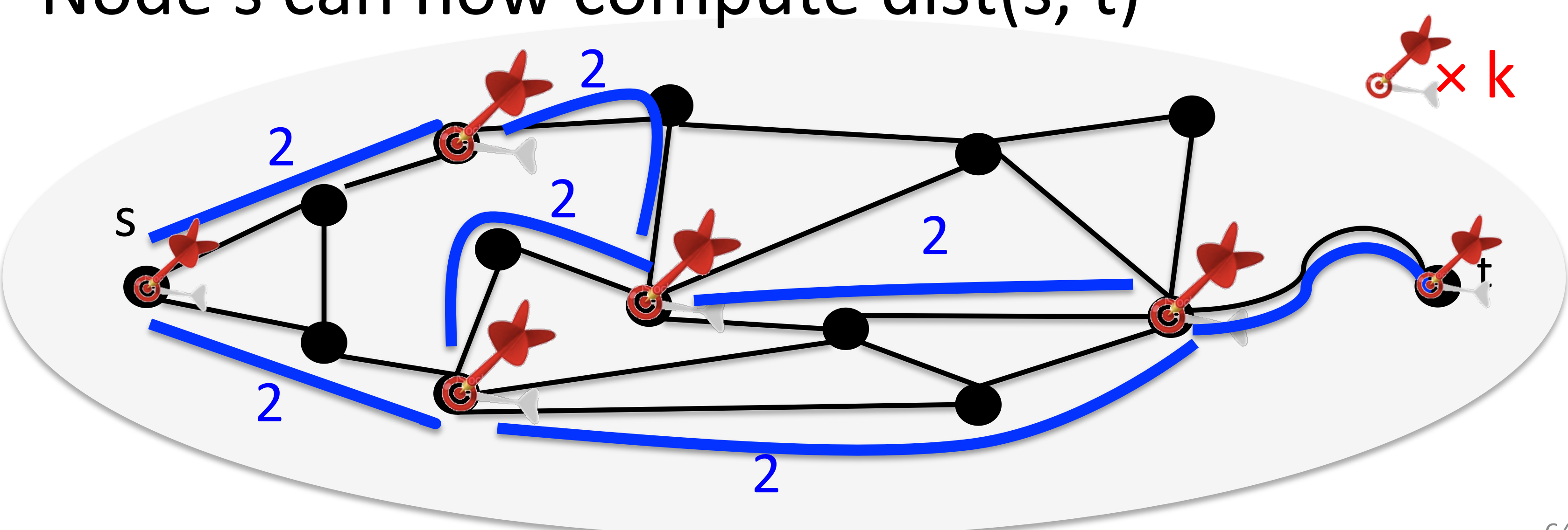
10-second proof: k random nodes will split u-v path into subpaths of length at most n/k



Consequence to s-t distance computation

We can $(1+\varepsilon)$-approximate $dist(s,t)$ in $O(n^{2/3}+D)$ time

1. Construct a k-skeleton H *that includes s and t*
   (Suffice to find $dist_H(s, t)$) $O(k+n/k)$ time
2. Broadcast this k-skeleton to all nodes. $O(k^2+D)$ time
   (s now knows about all blue edges)
3. Node s can now compute dist(s, t)



Consequence to s-t distance computation

We can $(1+\varepsilon)$-approximate $dist(s,t)$ in $O(n^{2/3}+D)$ time

1. Construct a k-skeleton H *that includes s and t*
   (Suffice to find $dist_H(s, t)$) $O(k+n/k)$ time
2. Broadcast this k-skeleton to all nodes. $O(k^2+D)$ time
   (s now knows about all blue edges)
3. Node s can now compute dist(s, t)

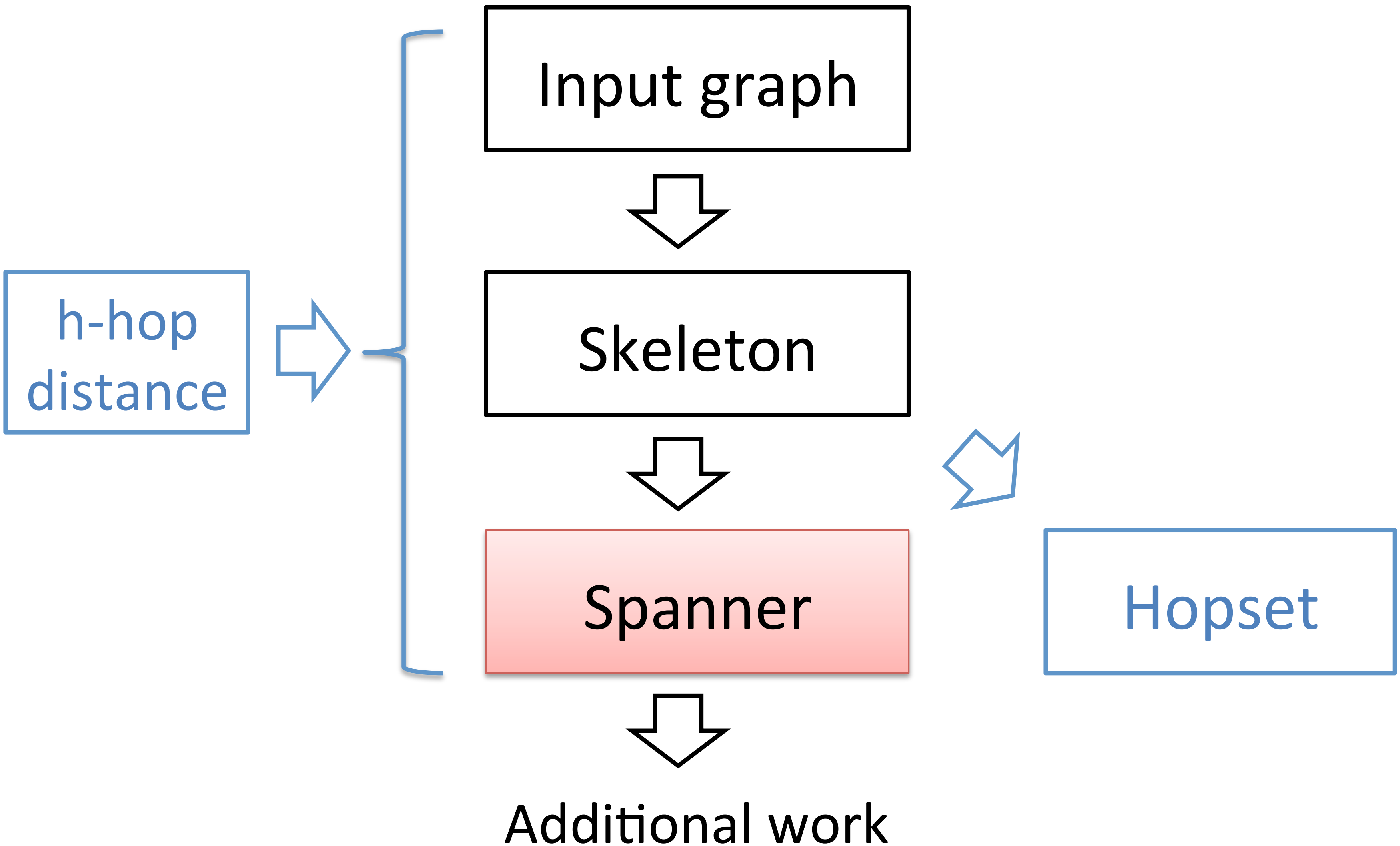
Set $k=n^{1/3} \rightarrow$ time = $O(n^{2/3}+D)$

Open problem

- Can we find **exact** k-skeleton in $O(k+n/k)$ time?
  (Recall: We can find an $(1+\varepsilon)$-approximate k-skeleton in $O(k+n/k\varepsilon)$ time.)

- If so, we will be able to solve SSSP exactly in sublinear time.

Common Framework



Can we reduce the running time for finding dist(s,t) further (e.g., from $O(n^{2/3}+D)$ to $O(n^{1/2}+D)$)?

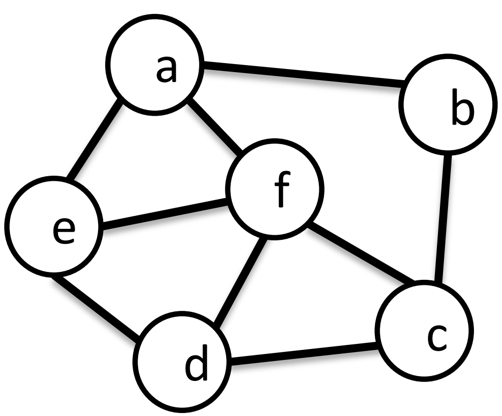
Main problem: There are too many edges (up to $O(k^2)$) in the skeleton

Skeleton alone is not usually useful because it’s too big.

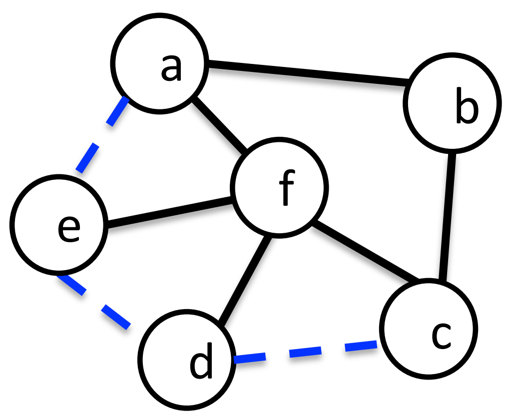
We must sparsify it.

Definitions

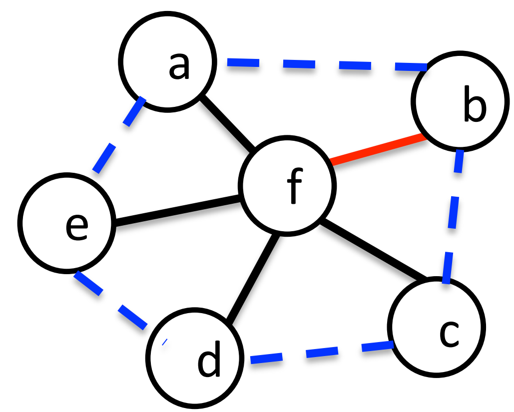
- p-spanner: Subgraph that preserves distances with multiplicative error p
- p-emulators: Graph on the same set of vertices that preserves distances



input graph



2-spanner



2-emulator

Computing spanner on distributed networks

- Baswana-Sen [Random Structures and Algorithms 2007]: $(2p-1)$-spanner of size $O(n^{1+1/p})$ in $O(p)$ rounds for any p.
- There's a huge literature on this.
  – See, e.g., Pettie [Distributed Computing 2010]

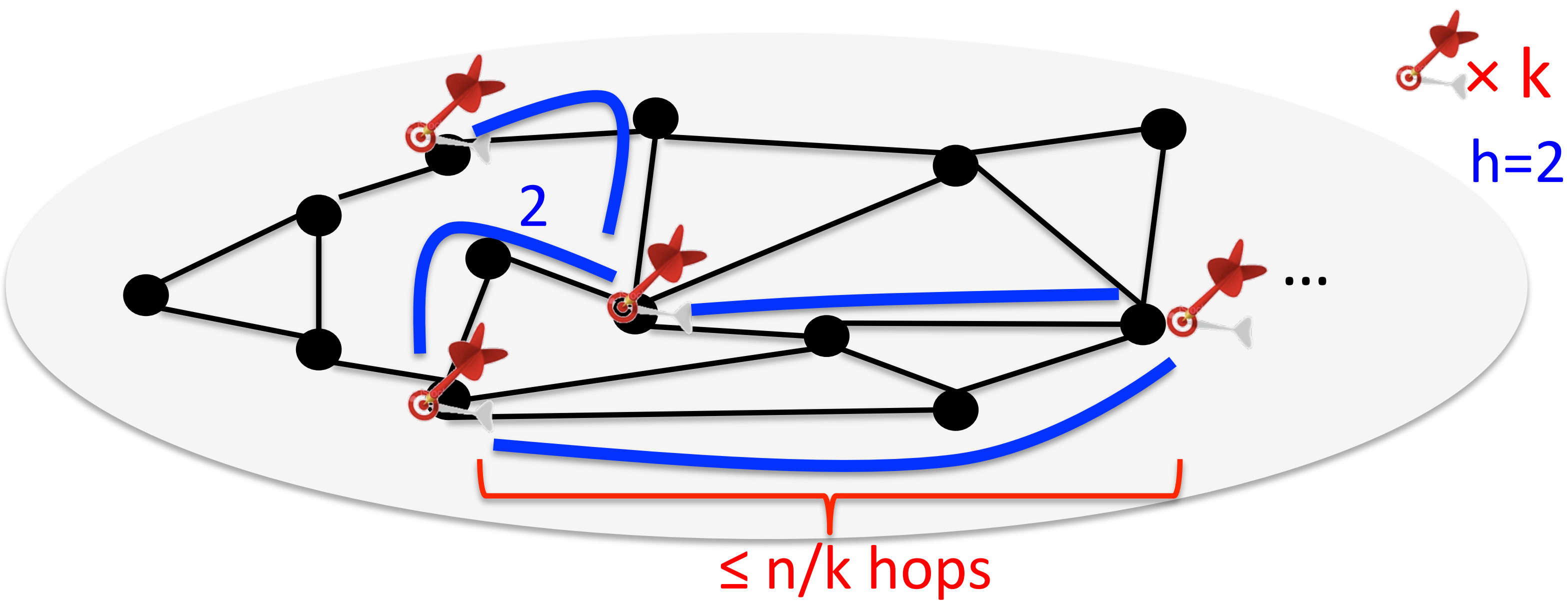
* It was pointed out by Pettie that the size of Baswana-Sen's spanner is $O(kn+(\log n)n^{1+1/k})$

Computing spanner on k-skeleton

- Simulate Baswana-Sen [Random Struct. & Algo. 2007]: (2p-1)-spanner of size $O(k^{1+1/p})$ in $O(pn/k)$ rounds for any p.

Computing spanner on k-skeleton

- Simulate Baswana-Sen [Random Struct. & Algo. 2007]: (2p-1)-spanner of size $O(k^{1+1/p})$ in $O(pn/k)$ rounds for any p.
  - We need n/k time to simulate each round.
  - No need to worry about congestion.



Computing spanner on k-skeleton

- Simulate Baswana-Sen [Random Struct. & Algo. 2007]: $(2p-1)$-spanner of size $O(k^{1+1/p})$ in $O(pn/k)$ rounds for any p.
  - We need n/k time to simulate each round.
  - No need to worry about congestion.
- In particular: $O(\log n)$-spanner of size $O(k)$ in $O(n/k)$ rounds for any p.
- Note: spanner of the skeleton can be computed *without* computing the skeleton
  - Lenzen, Patt-Shamir [STOC 2013]

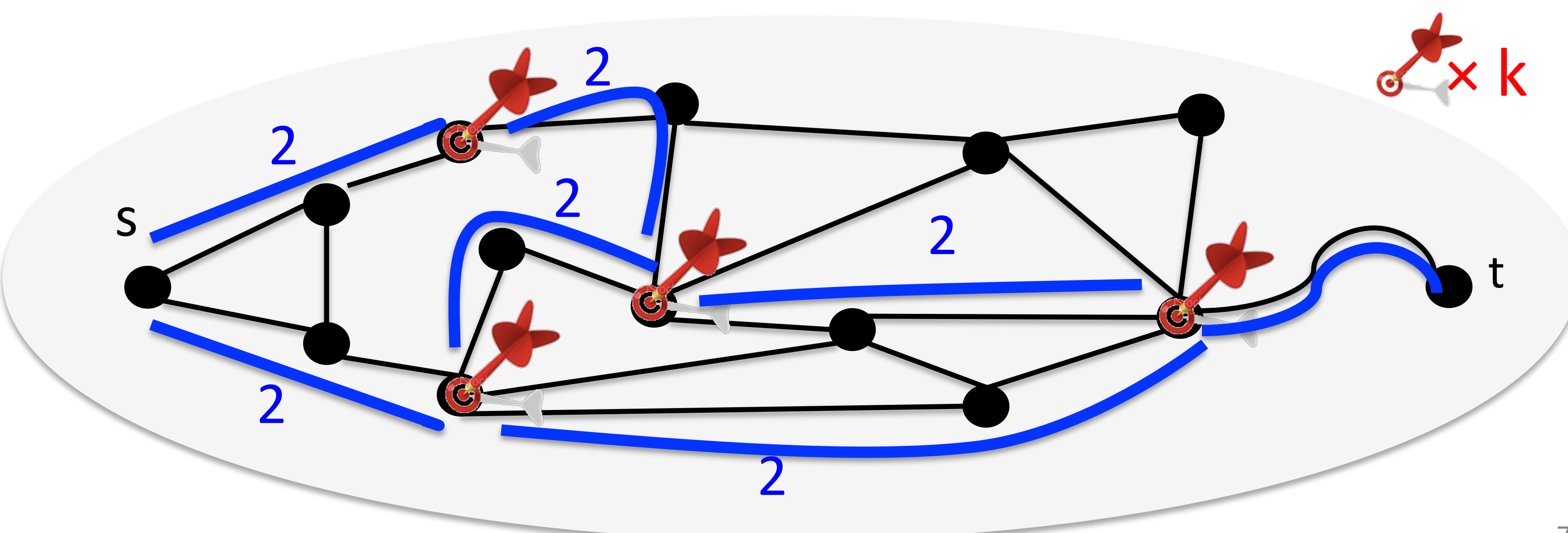
Computing spanner on k-skeleton

- Simulate Baswana-Sen [Random Struct. & Algo. 2007]: (2p-1)-spanner of size $O(k^{1+1/p})$ in $O(pn/k)$ rounds for any p.
- Simulate Thorup-Zwick [JACM 2005]: (2p-1)-emulator of size $O(k^{1+1/p})$ in $O(D+pn/k)$ rounds for any p.

Consequence to s-t distance computation

We can $(1+\varepsilon)$-approximate dist(s,t) in $O(n^{2/3}+D)$ time

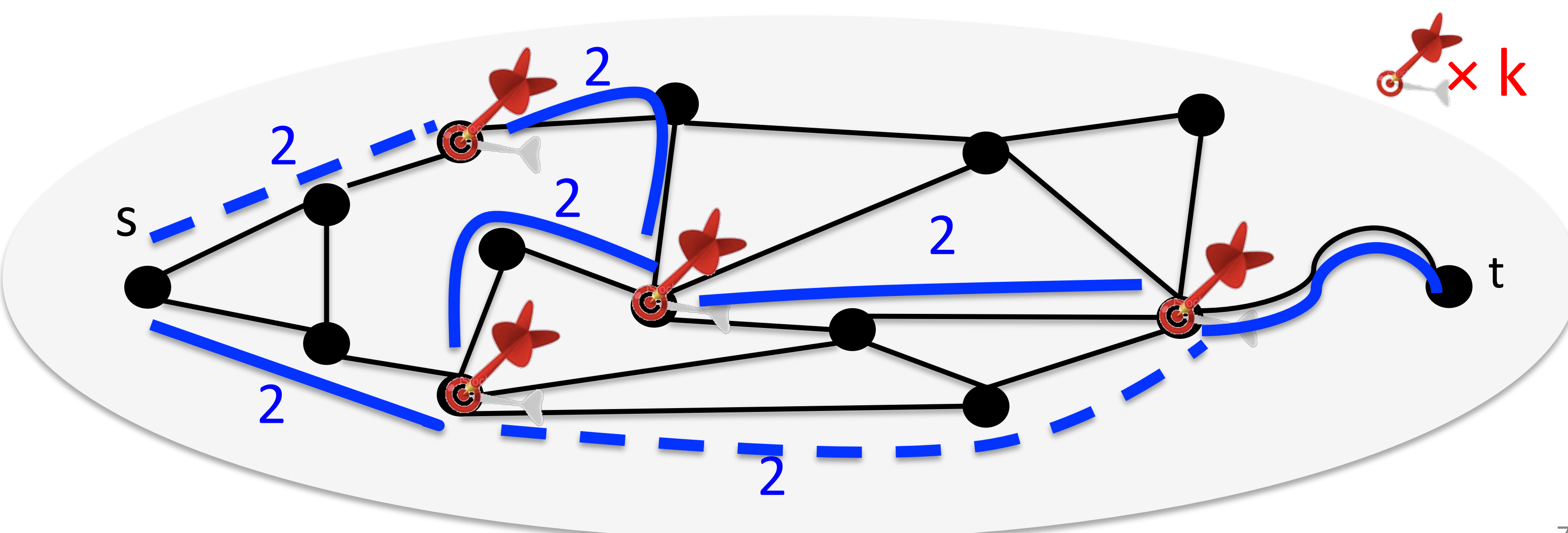
1. Construct a k-skeleton that includes s and t



Consequence to s-t distance computation

We can $(1+\varepsilon)$-approximate $dist(s,t)$ in $O(n^{2/3}+D)$ time

1. Construct a k-skeleton that includes s and t
2. Construct an **O(log n)-spanner of** this k-skeleton
3. Broadcast this spanner to all nodes. $O(k+D)$ time
   (s now knows about all red edges)



Consequence to s-t distance computation

We can $O(\log n)$-approximate dist(s,t) in $O(n^{1/2}+D)$ time

1. Construct a k-skeleton that includes s and t
2. Construct an **$O(\log n)$-spanner of** this k-skeleton
3. Broadcast this spanner to all nodes. $O(k+D)$ time
   (s now knows about all red edges)
4. Node s can now compute dist(s, t)

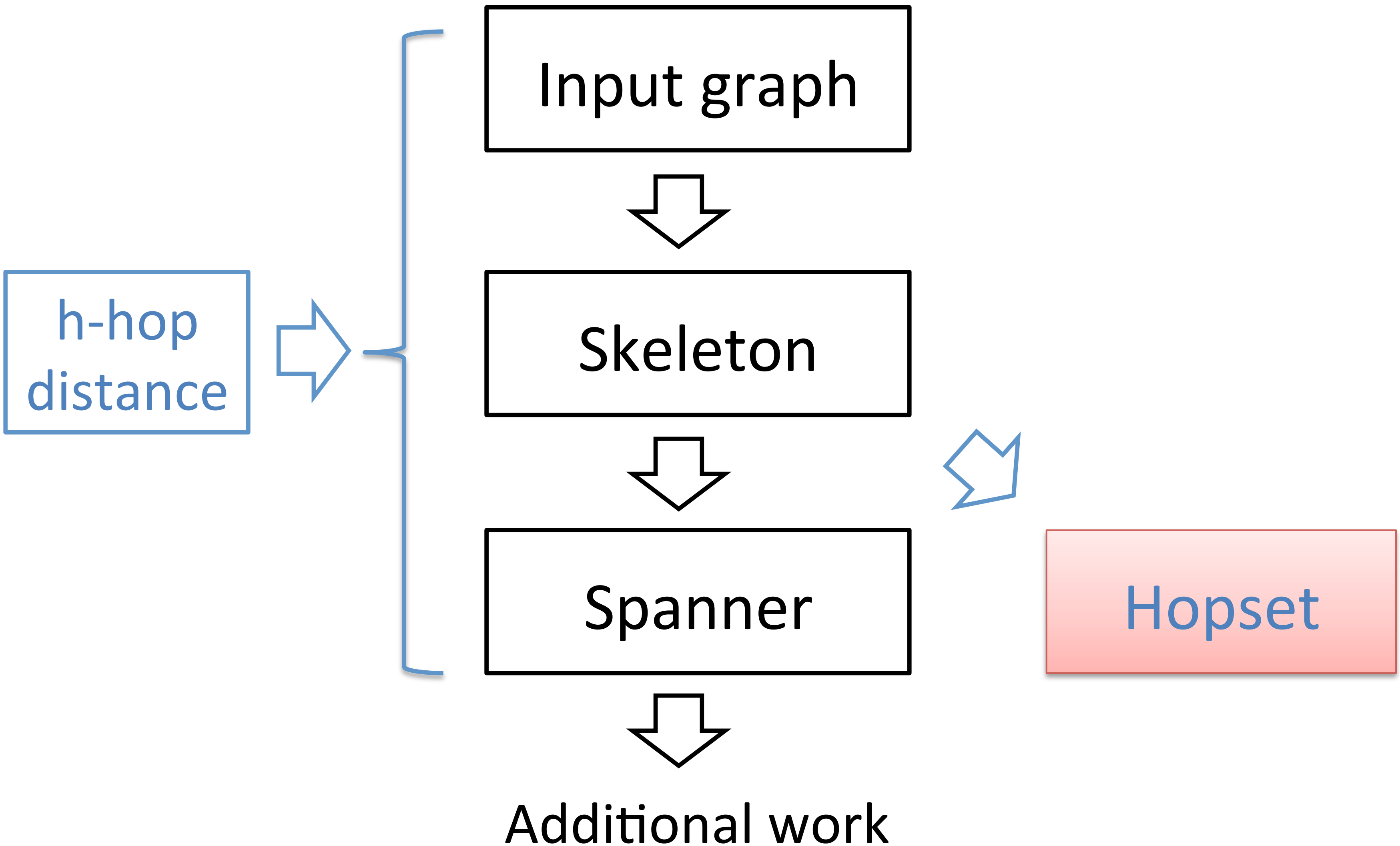
Set $k=n^{1/2}$ → time = $O(n^{1/2}+D)$

Other Consequences

Part of algorithms for

- Distance labeling and routing table Lenzen, Patt-Shamir [STOC'13]
- Tree Embedding Ghaffari-Lenzen [DISC'14]
- Steiner Forest Lenzen, Patt-Shamir [PODC'14]

Common Framework



Can we $(1+\varepsilon)$-approximate dist(s,t) in $O(n^{1/2}+D)$ time?

Recall: We can

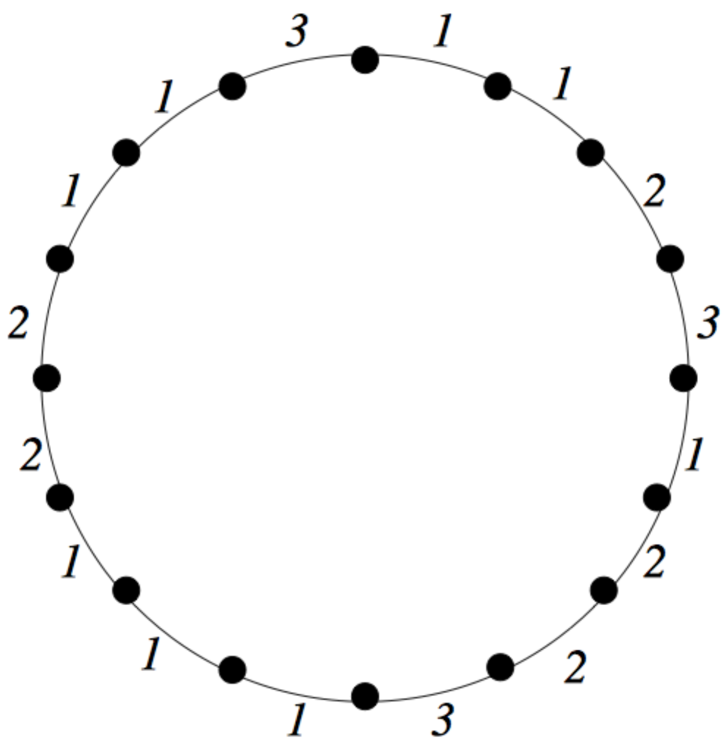
- $O(\log n)$-approximate dist(s,t) in $O(n^{1/2}+D)$ time
- $(1+\varepsilon)$-approximate dist(s,t) in $O(n^{2/3}+D)$ time

Problem with sparsification: it incurs an unavoidable large error

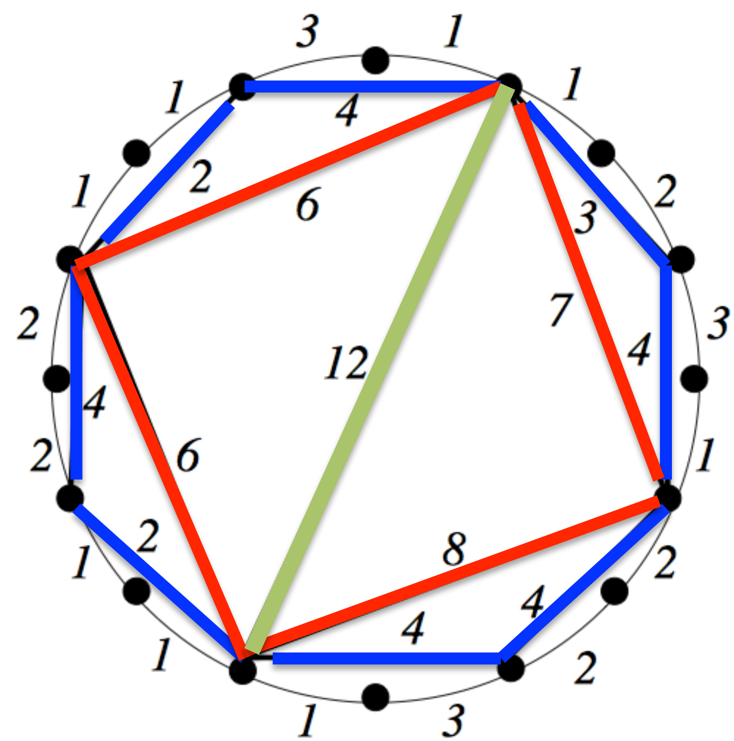
An alternative way: adding shortcuts

Definitions

- (h,ε)-hopset: A set of edges that when added to the input graph, h-hop distances (1+ε)-approximate the original distance



Input graph



Input graph & (5, 0)-hopset

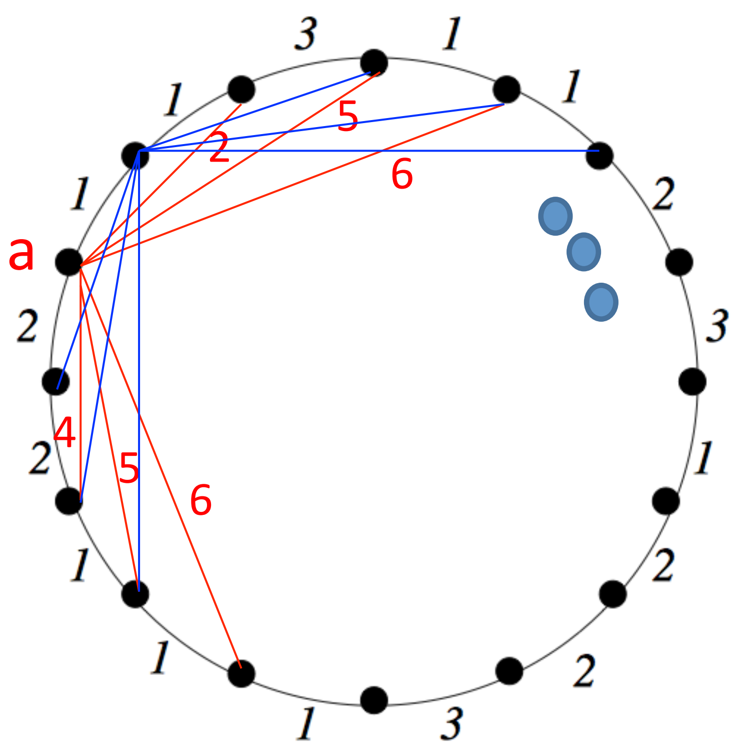
Picture from Cohen [JACM'00]

Theorems

- Cohen [JACM'00]: There is a (polylog n, ε)-hopset of size $n^{1+o(1)}$. (She can construct this in PRAM)
- Bernstein [FOCS'09]: Thorup-Zwick's emulator can be modified to construct an ($n^{o(1)}$,ε)-hopset of size $n^{1+o(1)}$.
- Henzinger, Krinninger, N [FOCS'14]: Bernstein's construction can be to modified to get an ($n^{o(1)}$,ε)-hopset of size $n^{1+o(1)}$ in the partially-dynamic setting.
- Further observation: The construction of Henzinger et al. can be implemented on the k-skeleton in time $O(k^{1+o(1)}+D^{1+o(1)})$

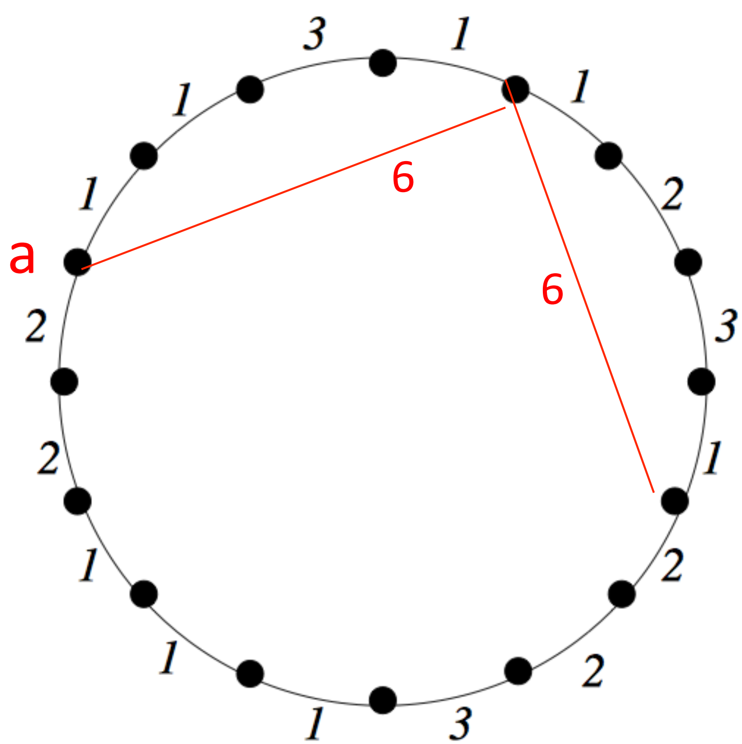
Hopset construction overview

- For every u and v that are ≤n/h hops away, add edge uv with weight $dist^h(u,v)$



Hopset construction overview

- For every u and v that are ≤n/h hops away, add edge uv with weight $dist^h(u,v)$
- This is an (O(h),0)-hopset

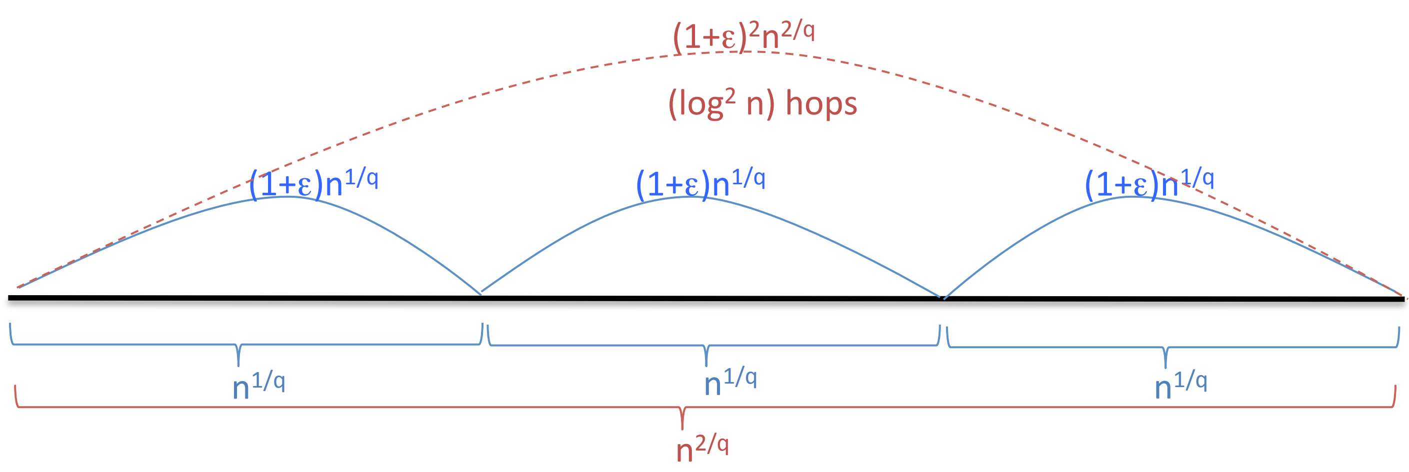


Hopset construction overview

- For every u and v that are $\leq n/h$ hops away, add edge uv with weight $dist^h(u,v)$
- This is an $(O(h),0)$-hopset
- **Sparsify** using Thorup-Zwick's emulator which gives $(1+\varepsilon)$ multiplicative error and some constant additive error.
- The additive error can be ignored if h is large enough

Hopset construction overview (2)

- Thorup-Zwick emulator can't be implemented on distributed networks since it needs to compute single-source shortest paths.
- But it can be implemented using bounded-hop single-source shortest paths & repeat algorithm multiple times.



Consequence

We can $(1+\varepsilon)$-approximate $dist(s,t)$ in $O(n^{1/2+o(1)}+D^{1+o(1)})$ time

1. Construct a k-skeleton that includes s and t
2. Construct an **$O(k^{o(1)}, \varepsilon)$-hopset of** this k-skeleton
   $O(k^{1+o(1)}+D^{1+o(1)})$
3. Compute $dist(s,t)$ on k-skeleton + hopset by simulating **$k^{o(1)}$**–hop distance algorithm.
   $O(k^{1+o(1)}+Dk^{o(1)})$ time (Simulation details omitted)

Set $k=n^{1/2}$ → time = $O(n^{1/2+o(1)}+D^{1+o(1)})$

Another application of hopset

- Routing Schemes Roditty-Tov [DISC'14]

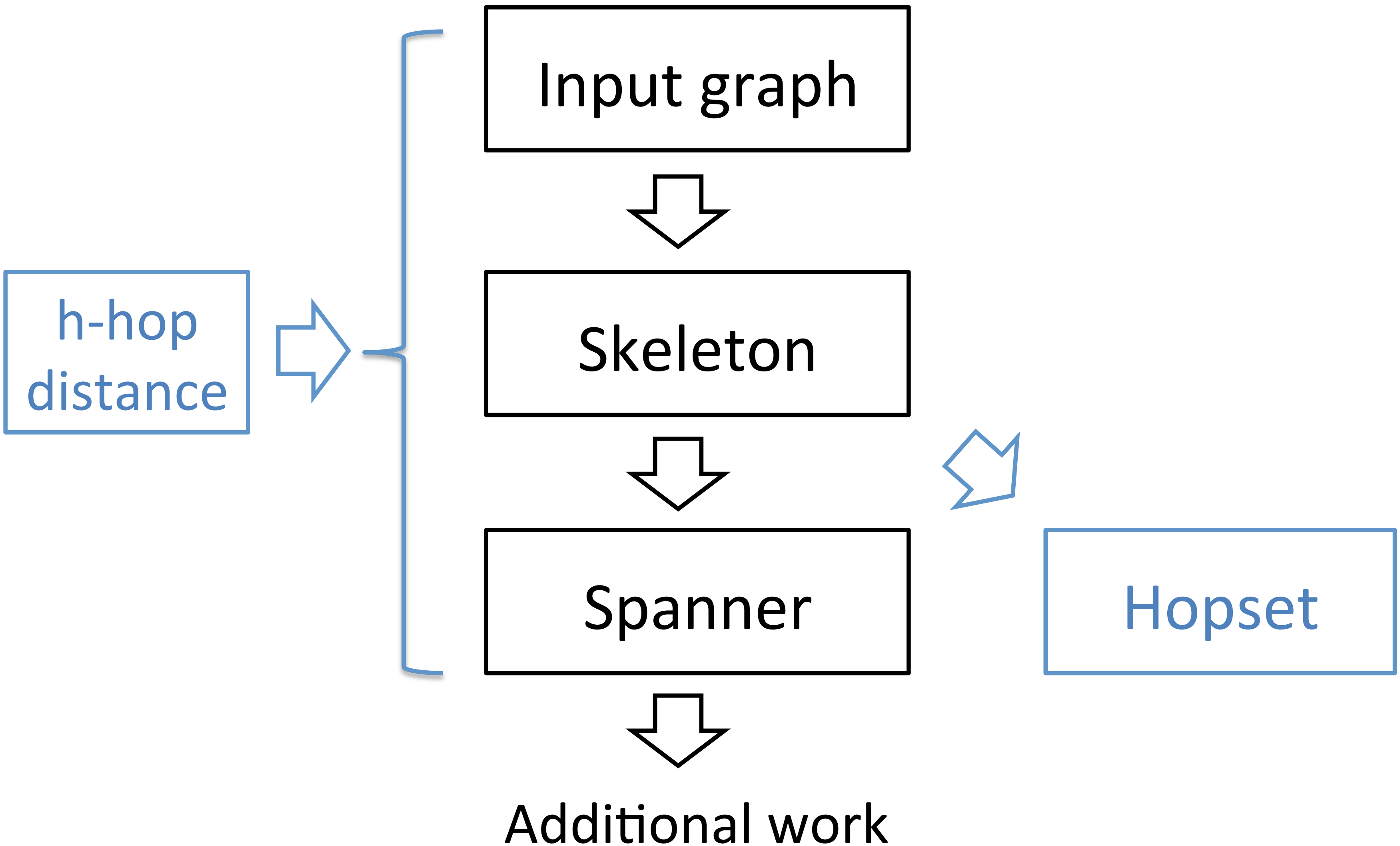
Open Problems

- Can we solve single-source shortest paths in $O(n^{1/2} \text{polylog } n + D)$ time?
  - Current: $O(n^{1/2+o(1)} + D^{1+o(1)})$
- Does there exist a $(\text{polylog } n, \varepsilon)$-hopset of size $O(n \text{ polylog } n)$? If so, can we compute it efficiently on distributed networks?
- Any other application of the hopset?

Part 5

Summary

Common Framework



Last open problem

Any other application of this framework?

Connections to other areas

- Using techniques above, we can solve SSSP in $O(n^{1/2+o(1)}+D^{1+o(1)})$ time.
- Using the same techniques, we can
  - solve SSSP on streams with $O(n^{1+o(1)})$ space and $O(n^{o(1)})$ passes
  - maintain SSSP in the partially-dynamic setting in $O(m^{1+o(1)})$ total time, and
- All results rely on the fact that we can compute/maintain h-hop distances fast
- Do similar connections exist for other problems, e.g. cut and matching?

Thank you