

Low-congestion distributed algorithms

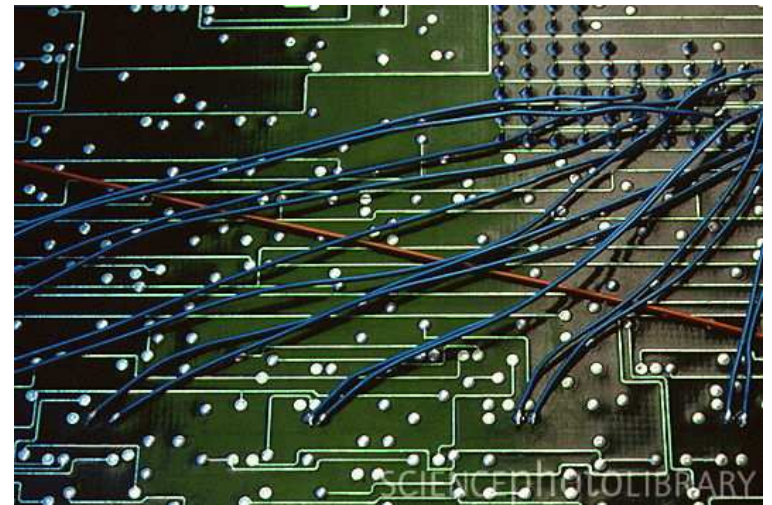
Boaz Patt-Shamir
Tel Aviv University

Talk overview

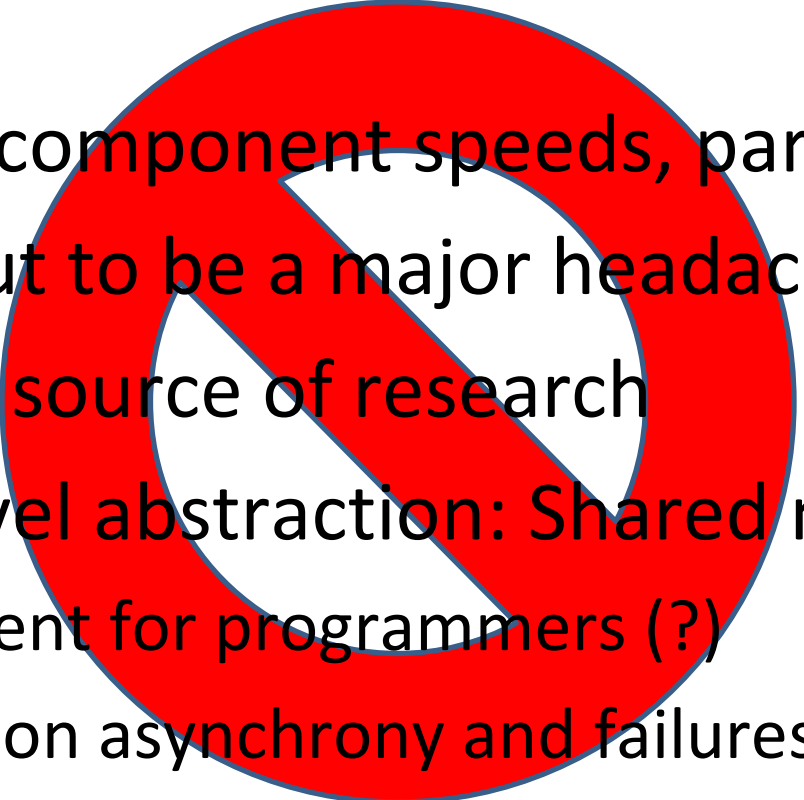
- Introduction to network algorithms: the **LOCAL** and the **CONGEST** models
- Approximate routing in the **CONGEST** model (joint work with Christoph Lenzen)

Distributed Algorithms

- Turing's vision: multiple heads, multiple tapes, but central control
- Today's technology: hook up components by communication lines
- Abstraction: network of processors exchanging messages



Some Issues

- 
- Different component speeds, partial failures
 - Turned out to be a major headache...
 - ... = a rich source of research
 - Higher level abstraction: Shared memory
 - Convenient for programmers (?)
 - Focuses on asynchrony and failures

Not our topic!

Our Focus: Locality Constraints

The **LOCAL** model

- Connectivity \equiv a graph $G = (V, E)$
 - Nodes are processors, edges are links
- Nodes can send arbitrary messages
 - No failures (processors or communication)
- Running time:
 - DEFINE: longest message delay \equiv one time unit
 - Neglect local processing time

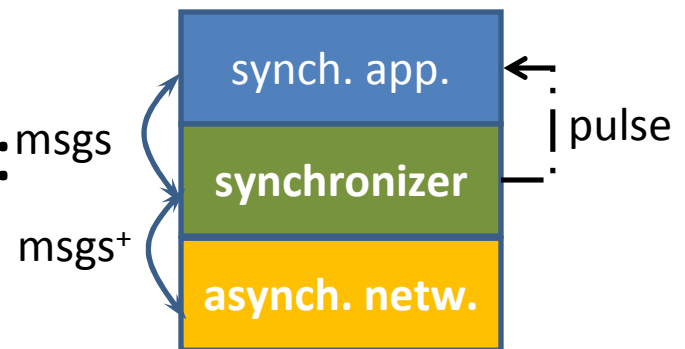


The **LOCAL** model: Typical Tasks

- Compute functions of the topology
 - Spanning Trees: Breadth-First (shortest paths), Minimum weight (MST)
 - Maximal Independent Set, Maximal Matching
- Communication tasks
 - Broadcast, muticast, gossip, end-to-end
- In general: **input/output relation**
 - Dynamic version: **reactive tasks**

Communication vs. Time

- Original Goal: minimize total #messages sent
 - A measure of total (sequential) work
 - Example: MST algorithm [GHS'83], BFS [A'89]
 - Sometimes count total #bits (broadcast [AGPV'88])
- Major problem: Asynchronous communication
- Awerbuch's synchronizer: pay with messages to buy speed



Synchronous **LOCAL** model

- Operates in synchronous global rounds
- In each round, all processors:
 - Receive messages from the previous round
 - Do some local computation
 - Send messages to neighbors
- Eventually produce outputs
- Inputs can be viewed as messages sent by the environment at round 0

Synchronous **LOCAL** model

- **Any** i/o-relation solvable in diameter time:
 1. Construct a BFS tree
(need IDs/randomization to choose root: Leader Election!)
 2. Send all input to root (“convergecast”)
 3. Root computes all outputs, sends back (“broadcast”)
- Ridiculous? That’s the client-server model!
 - Bread-and-butter distributed computing in the 70’s-90’s, and beyond...
- Interesting? Theoretically : sub-diameter upper and lower bounds

Canonical Algorithm and Interpretation

A problem can be solved in T time in **LOCAL** iff the following algorithm works:

- Each node collects full information from T -neighborhood, and computes output

Time is the “radius” of input influence.

LOCAL time = problem locality!

- Example: $\Omega(\sqrt{\log n})$ lower bound on time for approx. MIS, Matching [KMW'06]

Bounded messages make a difference

- Broadcast when only neighbors are known
 - $O(n)$ unbounded messages
 - $\Omega(m)$ bits [AGPV'88]
- Early 90's surprise: If messages have $O(\log n)$ bits, then MST construction requires $O(\sqrt{n} + D)$ time* using [GKP'93]

* All asymptotic notation is **soft**!
 $O(f)$ really means $O(f \log^c f)$,
 $\Omega(f)$ means $\Omega(f / \log^c f)$,
for some $c > 0$.

n : # nodes
 m : # edges
 D : diameter

Ergo: The **CONGEST** model

- Same as **LOCAL**, but *messages may contain up to **B** bits*

- Usually

- Allow
 - of m
 - Simil
 - Exact

- Capture



variables

ithfully

Extreme Case: Clique

- In **LOCAL**: only length (distance) is important
- In **CONGEST**: length & width
- Interesting case: width only
- Model: graph is fully connected, messages are $O(\log n)$ bits
- Can do MST in $O(\log \log n)$ time [LPPP'03]
- Can sort $n \times n$ inputs in constant time [PT'11]⁺
- No non-trivial lower bounds!

Bad Graphs for **CONGEST**

- Low diameter: there's always a shortcut
 - Good enough for **LOCAL**
- In **CONGEST**: when shortcuts are narrow, low diameter not enough to transmit massive data
- State of the art: graphs of diameter $\log n$ for problems that need to transport \sqrt{n} bits
 - Extends to diameter 3 with weaker lower bounds

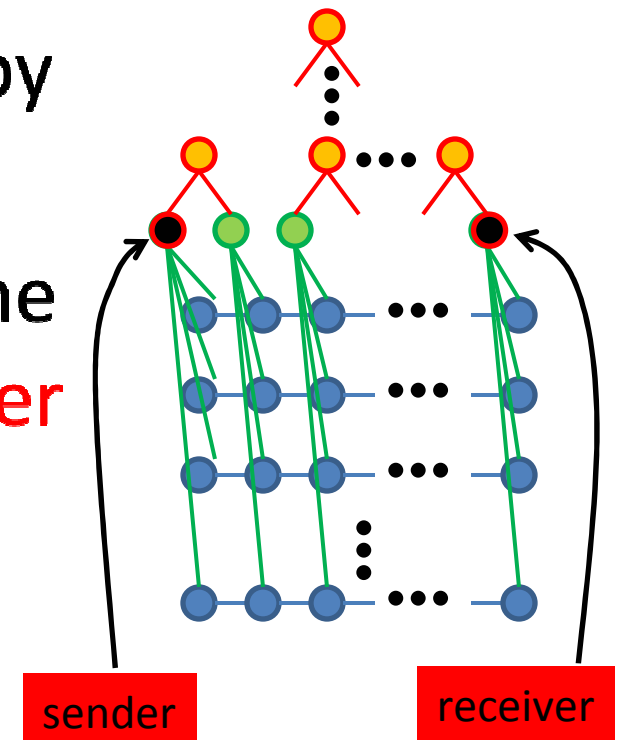
Bad Graphs for **CONGEST**:

Basic Construction

- Bulk: \sqrt{n} paths of length \sqrt{n} each
- Connect corresponding nodes by a star
- Build a tree whose leaves are the star centers $\Rightarrow O(\log n)$ diameter

File Transfer Problem: transmit \sqrt{n} bits from sender to receiver

- PR'99: Must take $\Omega(\sqrt{n})$ time!



Bad Graphs for **CONGEST**:

Very small diameter [LPP'01,E'04]

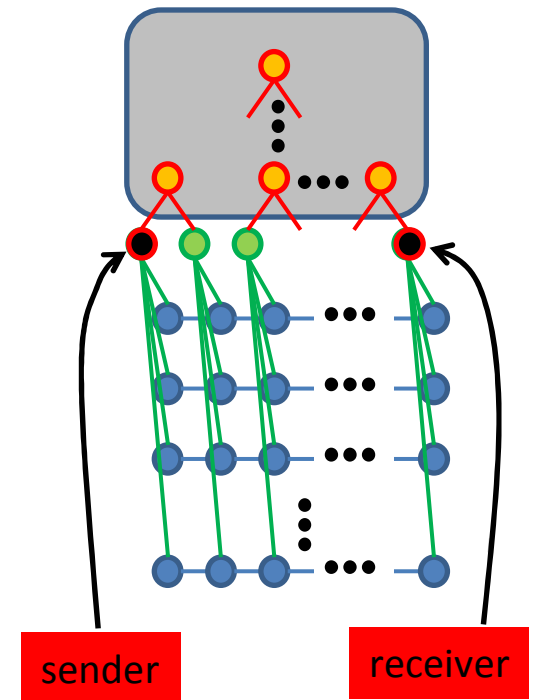
For $D \ll \log n$, replace binary tree with either

- d -ary tree, for $d \leq \sqrt{n}$, get
 $D = 2 + 2\log_d \sqrt{n} \quad (\geq 4).$

Lower bound: $\Omega\left(n^{\frac{1}{2} - \frac{1}{2D-2}}\right)$

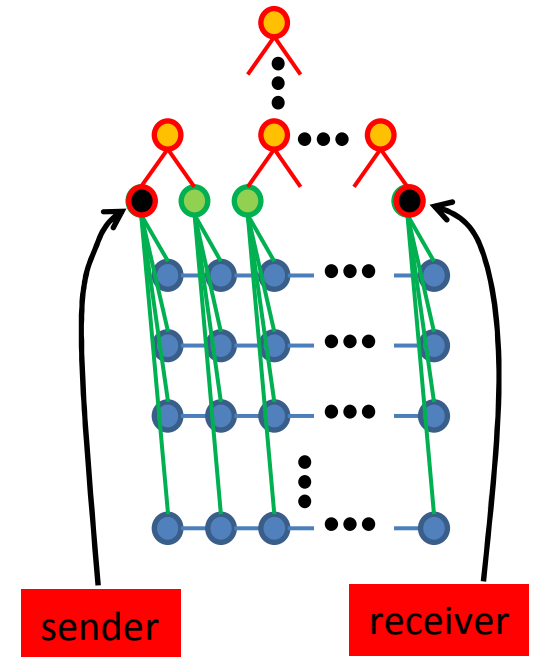
- Clique, get $D = 3.$

Lower bound: $\Omega\left(n^{\frac{1}{4}}\right)$



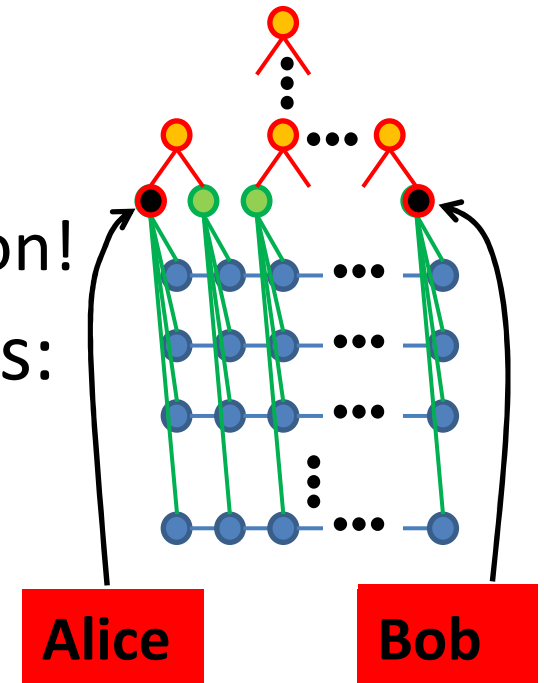
Bad Graphs for **CONGEST**: Uses

- To prove a time lower-bound on Π , reduce “file transfer” to π :
 - $\Pi = \text{MST}$ (edge weights) [PR’99]
 - Stable Marriage (rankings) [KP’09]
- Strengthening [E’04]: Can’t even **approximate** MST



Recent Progress [SHKKNPPW'11]

- Idea: Reduce problems to **Set Disjointness**, apply communication complexity lower bound.
- Easy to prove for **verification**
 - Sometimes harder than construction!
- But also for many graph problems: shortest paths, min-cut
- Side benefit: bound holds for any approximation

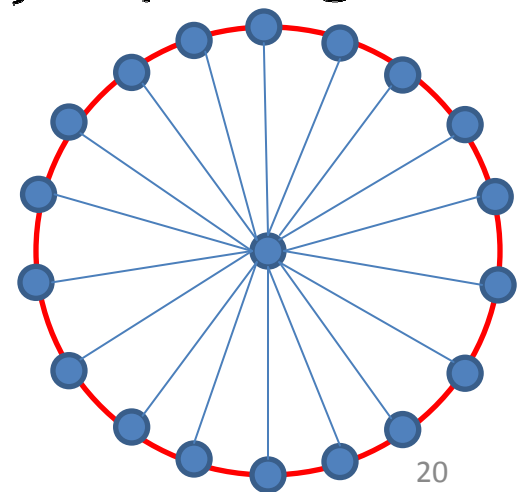


What About Upper Bounds?

- MST is computable in $O(\sqrt{n} + D)$ time [KP'95]
- Idea:
 - grow connected components by adding min-weight outgoing edges (parallel version of Prim)
 - Until components reach size $O(\sqrt{n})$
 - Then send all info on a BFS tree, root computes remainder and sends back (diameter time)

Interesting Question: Diameter

- Trivial to 2-approximate in $O(D)$ time (BFS!)
- Cannot be determined in $o(n)$ time [FWW'11]
 - Or even approximated to within 1.5-factor
- And how about **weighted** diameter?
 - Shortest weighted paths can be $\Omega(n)$ hops long, regardless of the hop-diameter
 - Usually done by solving all-pairs shortest paths, or, in Internet lingo, **routing**



Talk overview

Introduction to distributed computing: the **LOCAL** and the **CONGEST** models

Approximate routing in the **CONGEST** model
(joint work with Christoph Lenzen)

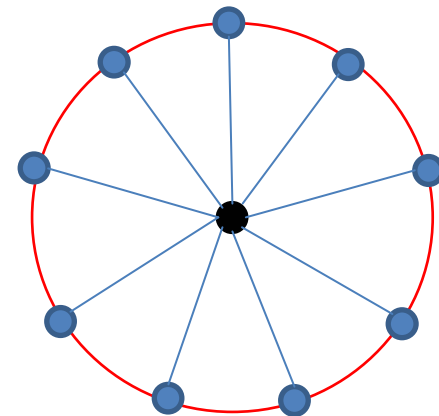
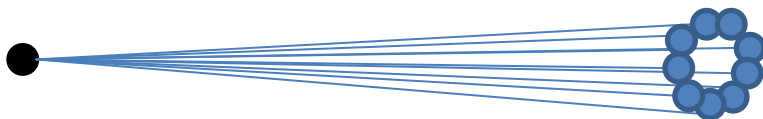
The routing problem

- Each node must be able to tell each packet what link to take next based on its destination
- Means: routing table
- Some issues:
 - **Stretch**: approx. factor wrt shortest paths
 - Stateful routing: next hop depends on history of the packet. Allows for partial table construction
 - Labeling nodes allows for encoding location in node's name



Routing Over Approx. Shortest Paths

- The $\Omega(\sqrt{n} + D)$ lower bound applies even for approximate shortest paths, but not $\Omega(n)$...
- One of the basic problems in the Internet
- Abstractly: two super-imposed topologies
 - Time complexity determined by hop count
 - Weights determine shortest paths



Tools

- Distributed Bellman-Ford algorithm
 - “Distance vector” algorithms
- Topology collection and central computation
 - “Link state” algorithms

We add:

- Hierarchical random sampling
- Spanner construction to reduce number of edges

Aside: Routing In the Internet

- Basic B-F used and taught under the name “distance vector algorithm ”
 - Each node advertises the vector of its distances from all sources (= destinations)
 - Packaged in BSD (called RIP), used in (E)IGRP
 - Hardwired bound on #hops (say, 30)
- Topology collection used in OSPF

The main idea

- Select $O(\sqrt{n})$ nodes uniformly at random: the “skeleton”
- **Key:** W.h.p., any set of ℓ nodes (paths too!) contains $\Theta(\ell/\sqrt{n})$ skeleton nodes
- Need to find:
 1. Short paths to close nodes (“Short Range” alg.)
 - One of them is a skeleton node
 2. Short paths on the skeleton (“Long Range” alg.)

The short range algorithm

- Tool: Bounded Bellman-Ford
 - Look only for the closest s sources (bdd **overlap**)
 - Go only for h iterations (bdd **distance**)
 - Will find routes to s sources with h or less hops
 - Running time: $O(sh)$
- Short range goal:
in $O(\sqrt{n})$ time, **all** nodes find good routes to
their $O(\sqrt{n})$ closest nodes

The Bellman-Ford Algorithm

Basic: Source has $d_s = 0$ fixed, other nodes apply in each round the relaxation:

$$d_v = \min\{w(u, v) + d_u \mid (u, v) \in E\}$$

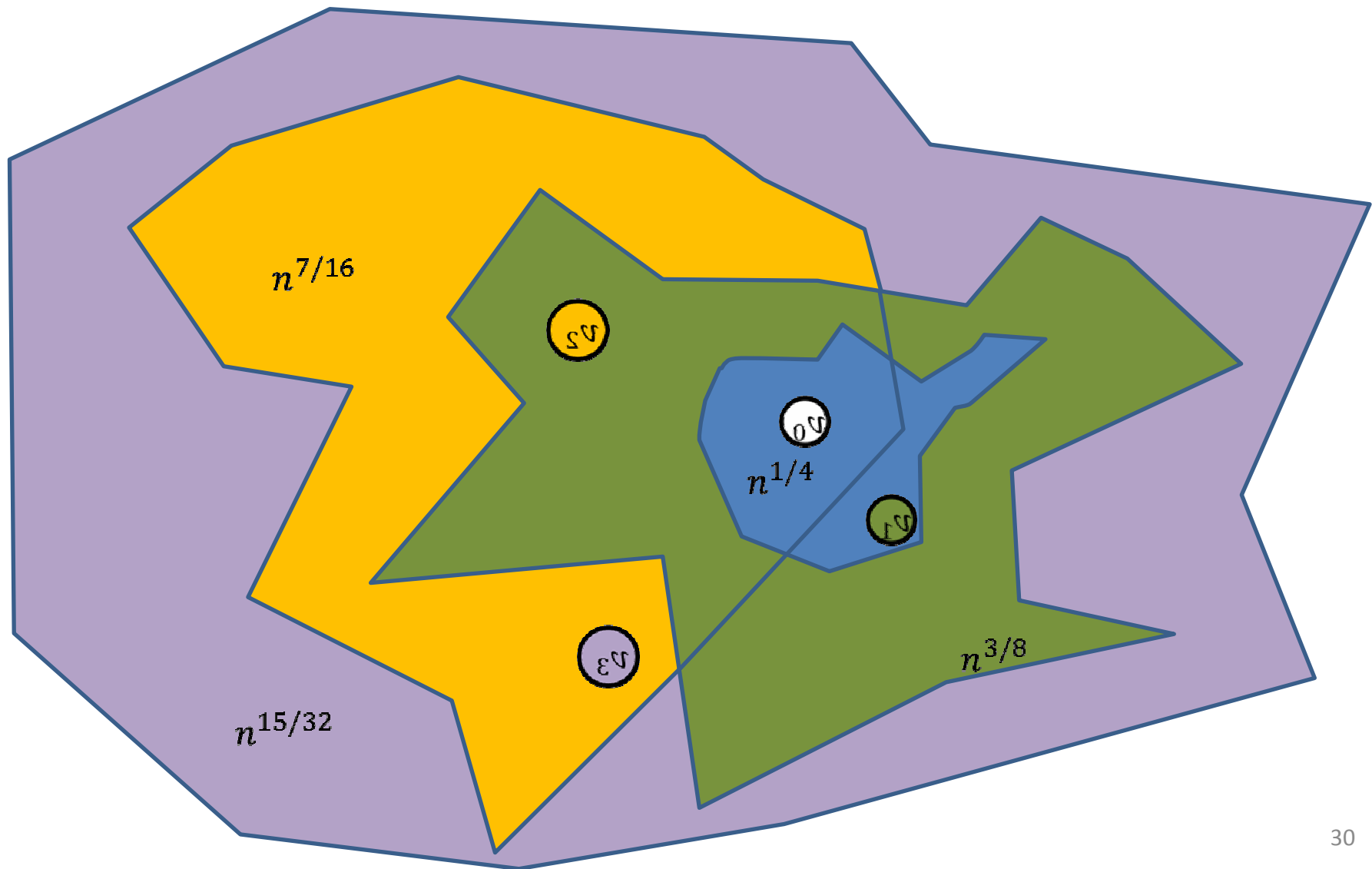
- After t rounds, know about all paths of t hops
- s sources: a different version of B-F for each
 - Time increases by factor s

Lemma: All nodes can find closest r sources in $O(rh)$ rounds if they are always within h hops

Short range: bootstrapping

1. Find routes to closest $n^{1/4}$ nodes
 - $n^{1/4}$ overlap, $n^{1/4}$ range: $n^{1/2}$ time; stretch: 1
2. Take $n^{3/4}$ random nodes as sources (“reps”)
 - W.h.p, know how to route to closest rep
 - Can go to $n^{3/8}$ range with $n^{1/8}$ overlap; stretch: 3
- ... etc. Level i : closest $n^{\frac{1}{2}-2^{-i}}$ nodes, stretch $< 2i$
- Routing: to and from each rep

Schematically:



Short-Range Algorithm

Goal: find paths to skeleton and closer nodes

Idea: Use bounded Bellman-Ford hierarchically

1. Find s.p. to $s_1 = n^{1/4}$ closest nodes
 - $n^{1/4}$ hops, $n^{1/4}$ overlap: $n^{1/2}$ time
2. Take $s_2 = n^{3/4}$ random nodes as sources
 - W.h.p., we know how to route to one of them
 - We can look for the closest them find $n^{3/8}$ closest nodes:
 - $n^{3/8}$ hops, $n^{1/8}$ overlap: $n^{1/2}$ time
- And so on, up to $O(\log \log n)$ levels:
 - Level i finds $n^{\frac{1}{2}-2^{-i}}$ closest nodes (overlap $n^{2^{-i}}$)

Short range: some details

- Route: from node to first rep that covers target and then from rep to target
 - Need relabeling! (unavoidable)
- Stretch is $L = \text{\#levels}$.
- Running time is $O(L n^{\frac{1}{2}+2^{-L}})$
 - If $L = \Theta(\log \log n)$ then running time is $O(\sqrt{n})$
 - If $L = O(1)$, get constant stretch and $O(n^{\frac{1}{2}+\epsilon})$ running time

Short range to Long Distance

- Can route to close nodes via closest rep covering both source and destination
 - Can't break the $\Omega(\sqrt{n})$ barrier
- At this point we go flat.


Find all-all routing for top level nodes

➤ How can this be done with cost $O(\sqrt{n} + D)$?

Long-Range Algorithm

Skeleton: $O(\sqrt{n})$ random nodes (top level of short-range hierarchy)

Goal: find shortest paths on **skeleton graph**

- The graph is virtual: edge = path
 - Use B-F: $O(\sqrt{n})$ range suffices
- 

But still, overlap may be high ...

**Solution: Build skeleton graph while sparsifying it!
Use spanner-construction algorithm [BS'03]**

Integrated Spanner Construction

Adapted from [BS'03]: k iterations of growing and merging clusters

- BFS tree used for global information dissemination

Properties:

- Increases distances by factor $O(k)$
- Reduces the number of sources to $O(n^{1/k})$
 - Total running time $O(k n^{1/2+1/k})$
- Reduces the number of edges to $O(k n^{1/2+1/k})$

Long range: Properties

- Given k : Skeleton spanner ends up with stretch $O(k)$ but only $O(n^{1/2+1/k})$ edges
- Can send skeleton edges to all skeleton nodes
 - Over a BFS tree, in $O(n^{1/2+1/k} + D)$ rounds
- Each skeleton node computes locally shortest paths to all skeleton nodes.

Results

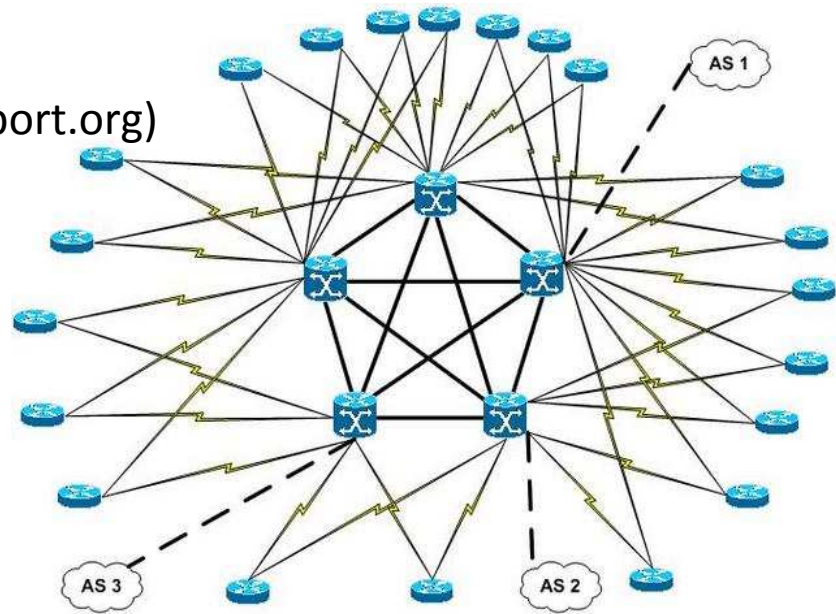
Theorem: Computing $(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ -stretch routes can be done in time $O(n^{1/2+\epsilon} + D)$ in **CONGEST**.

Corollaries:

- Approximate distances and **distance sketches**.
- Approximate **diameter**.
- Approximate **Generalized Steiner Forest**
 - Improving approximation from $O(\log n)$ to $O(1)$ while improving the running time!

Aside: Routing In the Internet

- The Internet is divided into “Autonomous Systems” (AS’s)
 - About 35000 (source: cidr-report.org)
- One protocol to route between AS’s (BGP)
- Another to route within each AS
 - Whatever the AS chooses
- Routers can find the AS by looking at the IP address (about 900,000,000 hosts)
source: isc.org



Open Problems

- **Stateful routing:** Should routers only forward packets?
 - **Dynamic topologies**
- **Competitive algorithms:** How to tailor the algorithm to the network at hand?
- The **topology of time-complexity**: why only unit-weight?
- Lower bound for the CLIQUE model

Conclusion

- $\sqrt{n} + D$ is a natural time bound in distributed systems with bandwidth constraints
- A new routing scheme for the Internet? 😊
- Many applications
- Many open problems that matter

Thanks!