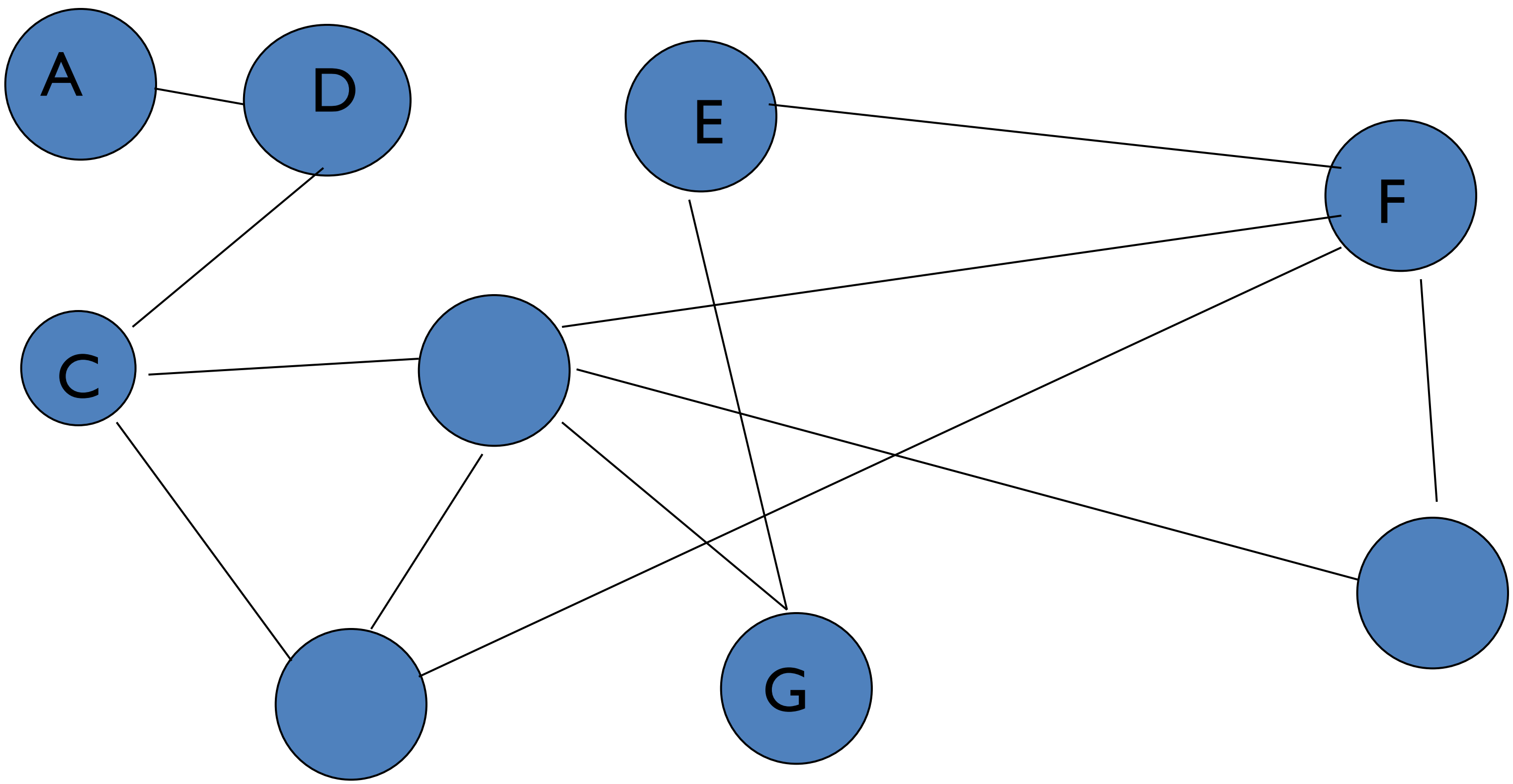
Sublinear communication in a message-passing network: Broadcast and MST

Valerie King
University of Victoria,
Vancouver Island, BC
Canada

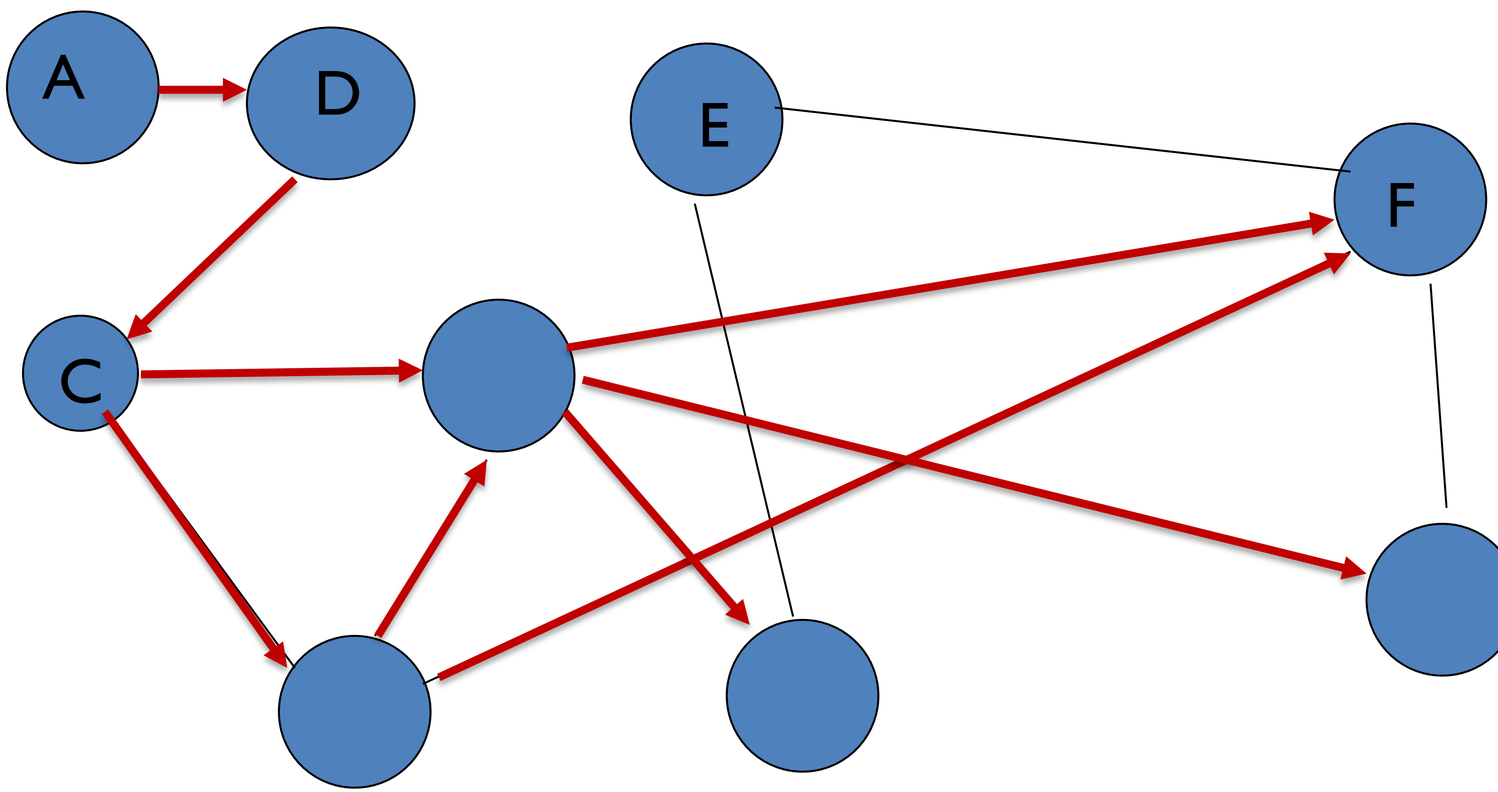
Network with n nodes, m edges

A node wants to send a message to all other nodes.

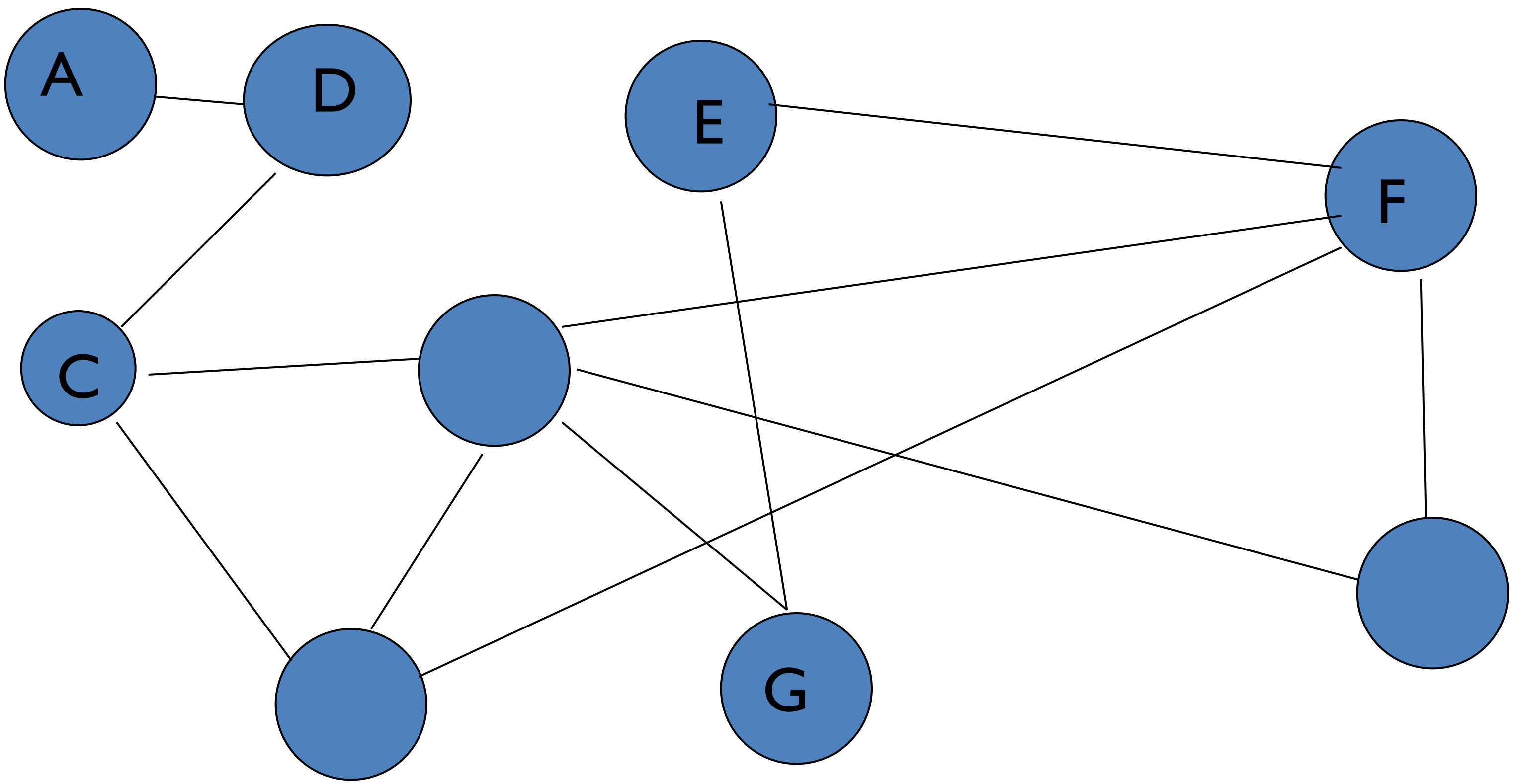
How many messages are needed?



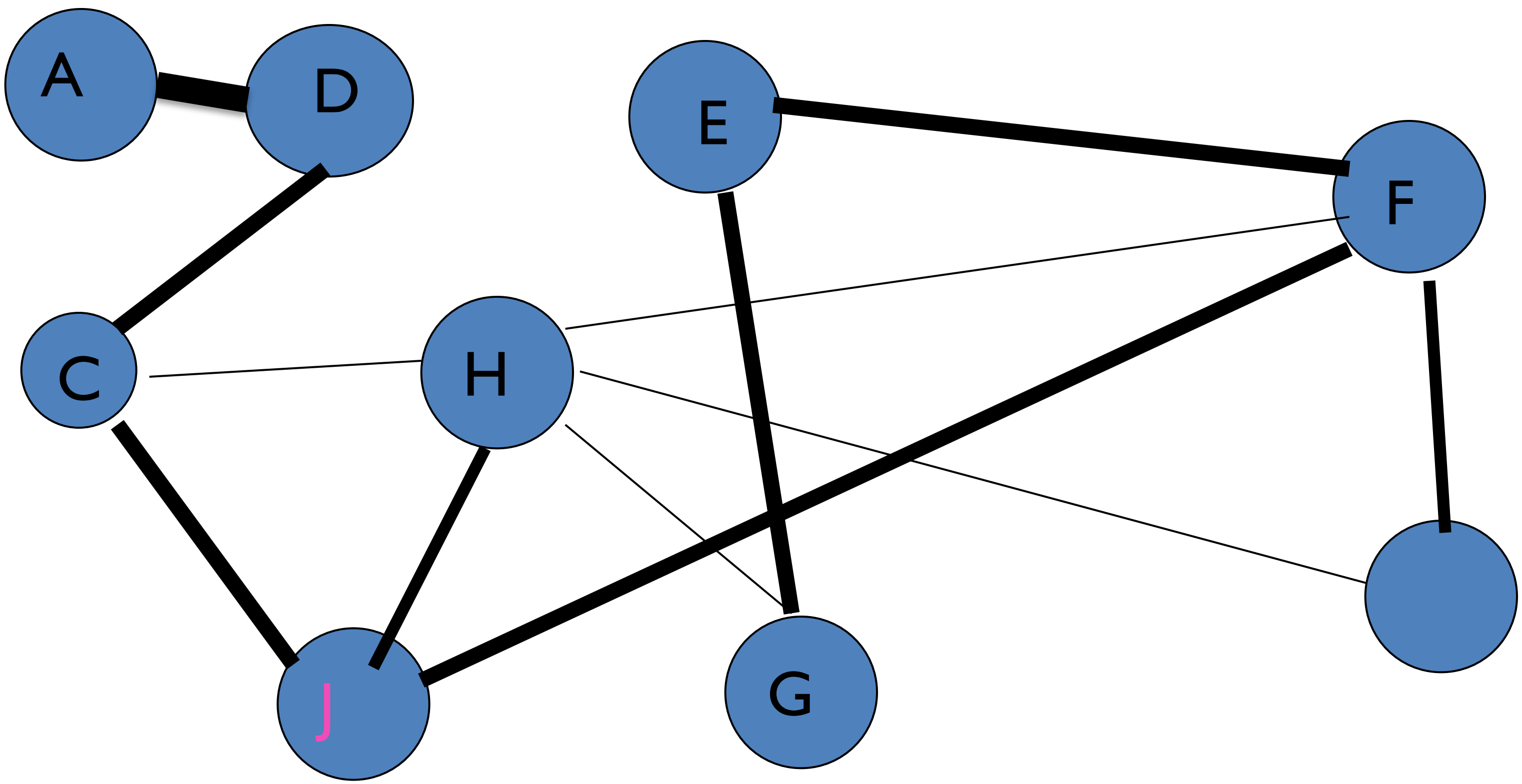
Use flooding from a source node
O(m) messages.



Can we do better than O(m) messages?

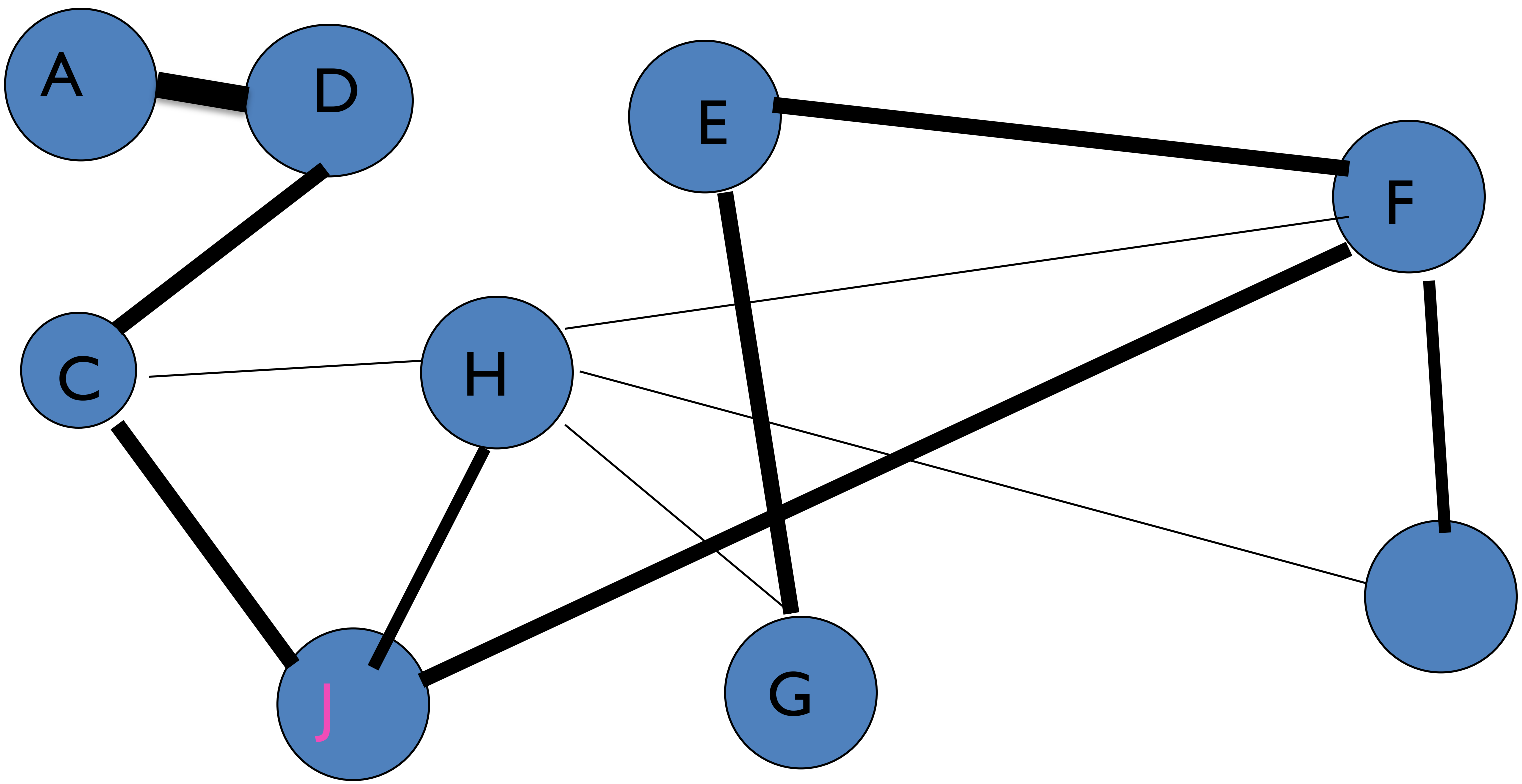


Given a spanning tree, can send broadcast in n-1 messages



Given a spanning tree, can send broadcast in n-1 messages

What if each node has only **local** info?



Talk Outline

- Models and lower bounds for spanning tree (and MST) construction.
- Upper bounds
  - Synchronous
  - Asynchronous
- An even simpler communication problem
- Time-Communication Tradeoffs
- Open problems

MODELS

Nodes have distinct IDs

Know only **LOCAL** info re topology
Auerbuch, Goldreich, Peleg, Vainish (AGPV)
(JACM 1990)

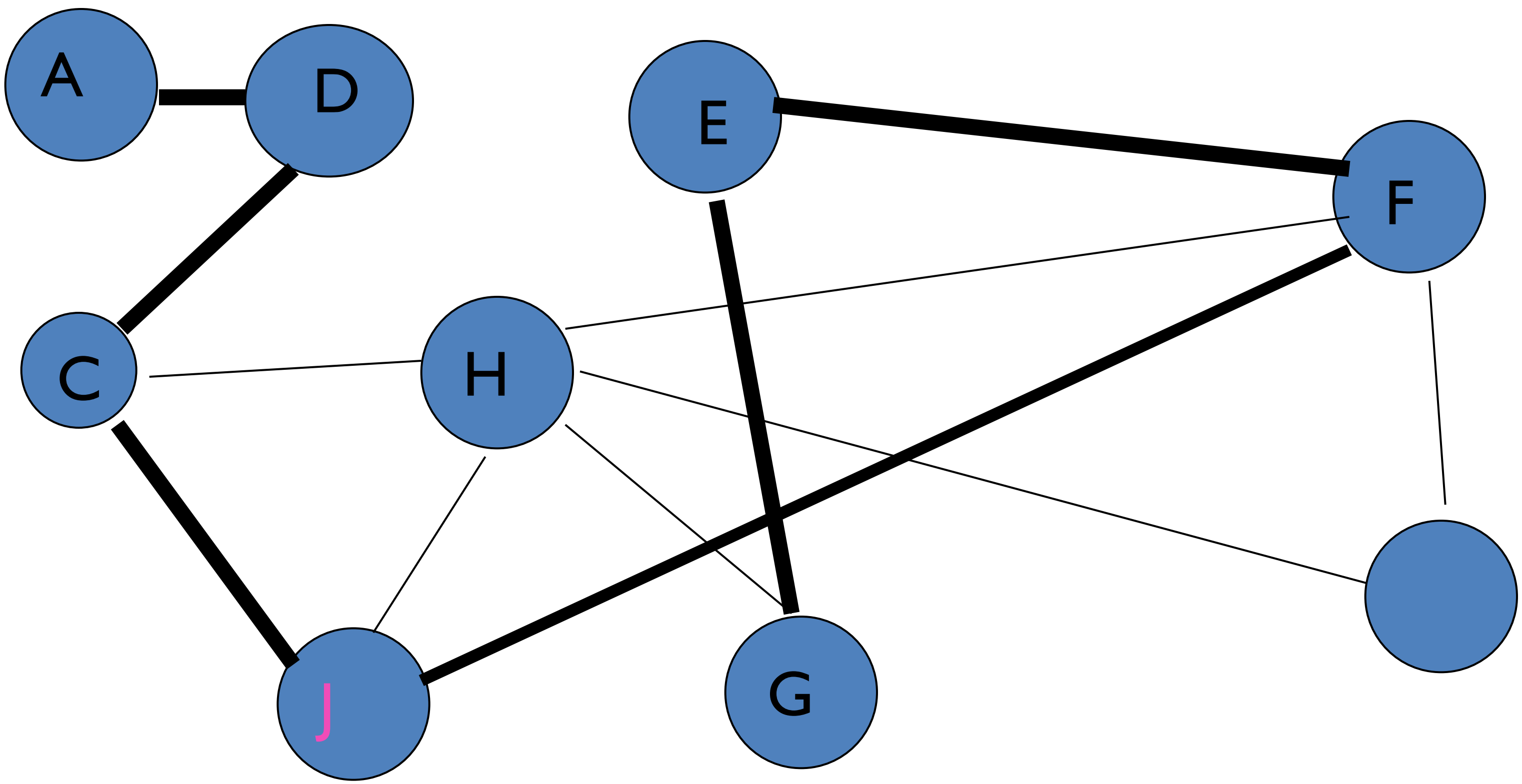
--KT_0 each node has a port to each neighbor

--KT_1 each node knows its neighbors' IDs

communication

- Synchronous: messages sent in a round received in the same round.
- Asynchronous: algorithm may wake up all or some nodes to start; later messages are sent in response to receipt of a message. (”event-driven”)

A network builds a subgraph G’ when nodes mark their incident edges which are in G’



Can we do better than flooding?
"No, according to folklore." (AGPV)

"If each proc knows only its ID and the IDs of its neighbors, then flooding (O(m)) is the best that can be done [even in the synchronous model]"

"Obvious" for KT_0 (AGPV1990) ???

lower bounds for spanning tree: KT_0

Kutten, Pandurangan, Peleg, Robinson, Trehan (PODC2013, JACM 2015)

Constructing a tree requires $\Omega(m)$ communication (where range of IDs has size n^4)

Even if

- synchronous
- nodes are all initialized at the start
- alg is Monte Carlo
- nodes know n

lower bounds for spanning tree: KT_0

Kutten, Pandurangan, Peleg, Robinson, Trehan (PODC2013, JACM 2015)

Constructing a tree requires $\Omega(m)$ communication where range of IDs has size n^4

Even if

- synchronous
- nodes are all initialized at the start
- alg is Monte Carlo
- nodes know n

First even for deterministic bound!

But much earlier lower bound for MST: Korach, Moran, Zaks PODC 1984)

lower bounds for spanning tree in KT_1

AGPV 1990

Constructing a tree requires $\Omega(m)$ communication where range of IDs has size $2n$

Even if

- synchronous
- nodes are all initialized at the start
- alg is Monte Carlo
- nodes know n

Provided…

lower bounds for spanning tree in KT_1

AGPV 1990

Constructing a tree requires $\Omega(m)$ communication where range of IDs has size 2n

Even if

- synchronous
- nodes are all initialized at the start
- alg is Monte Carlo
- nodes know n

Provided each message contains a constant number of ids and ids can only be compared

More on KT_1 : Without proviso

- AGPV requires use of Ramsey Theory,

  very large ID space, so that IDs are not O(log n) bits.

--><u>The lower bound is wide open in KT_1 even for deterministic algorithms.</u>

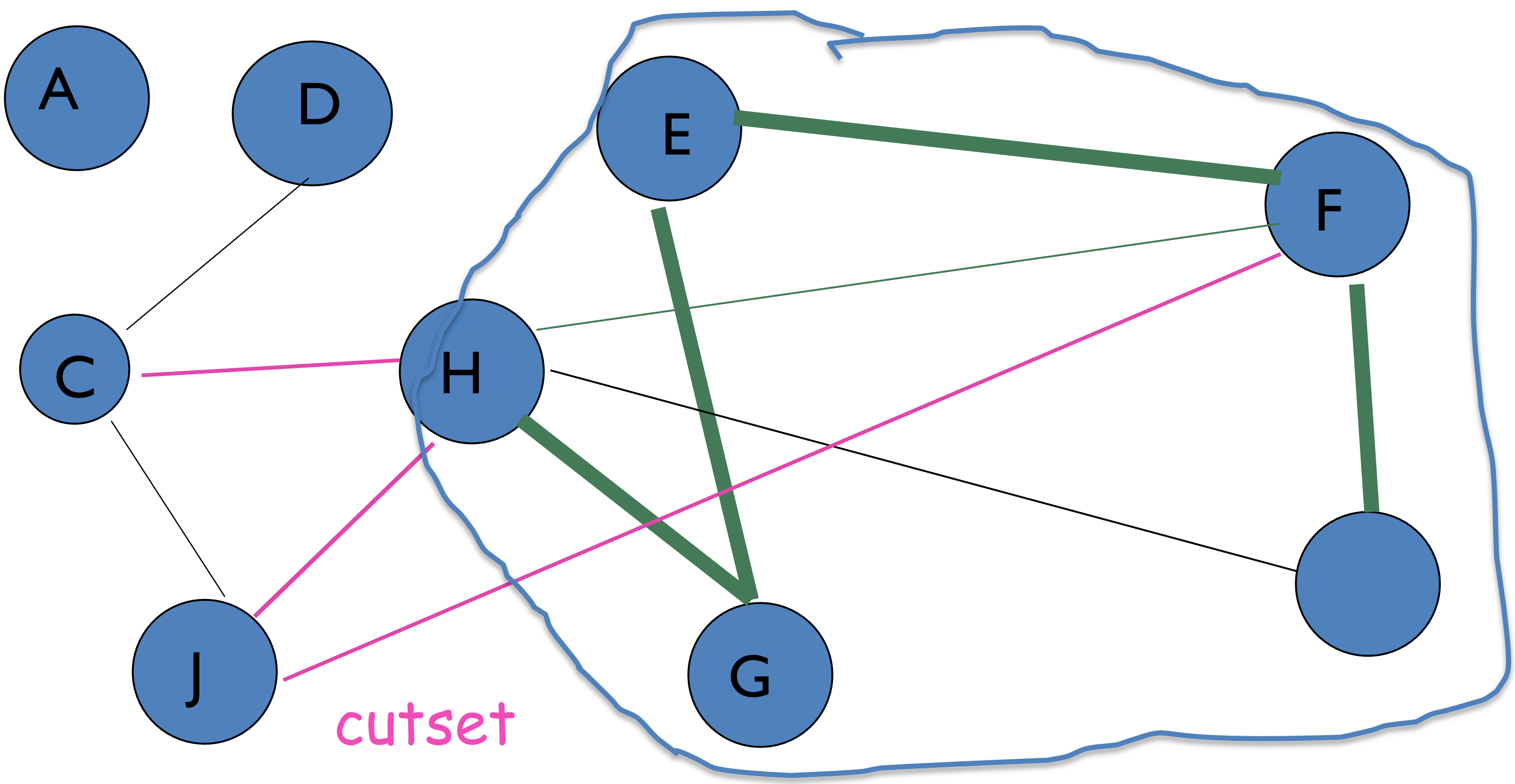
An aside: a non-communications network lower bound

- Where knowledge of an arbitrary graph's edges can be partitioned arbitrarily among n parties in a clique, $\Omega(n^2)$ bits of communicaton are required. Phillps, Verbin, Zhang (SODA 12)

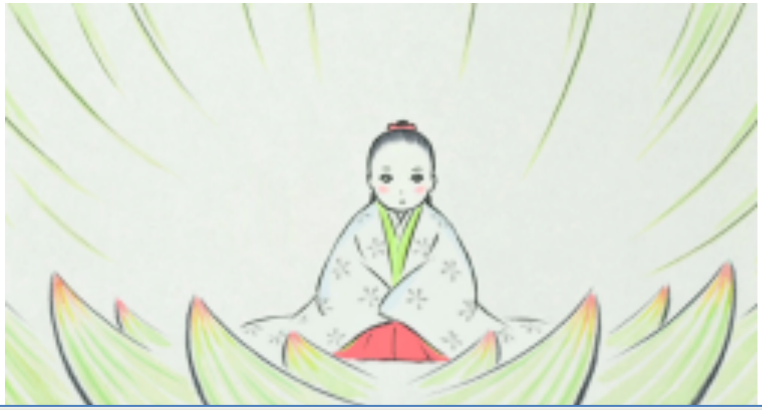
Upper bounds

Main Idea: How to find an edge leaving a marked tree T ("outgoing edge") with $\tilde{O}(|T|)$ bits

.



XOR method

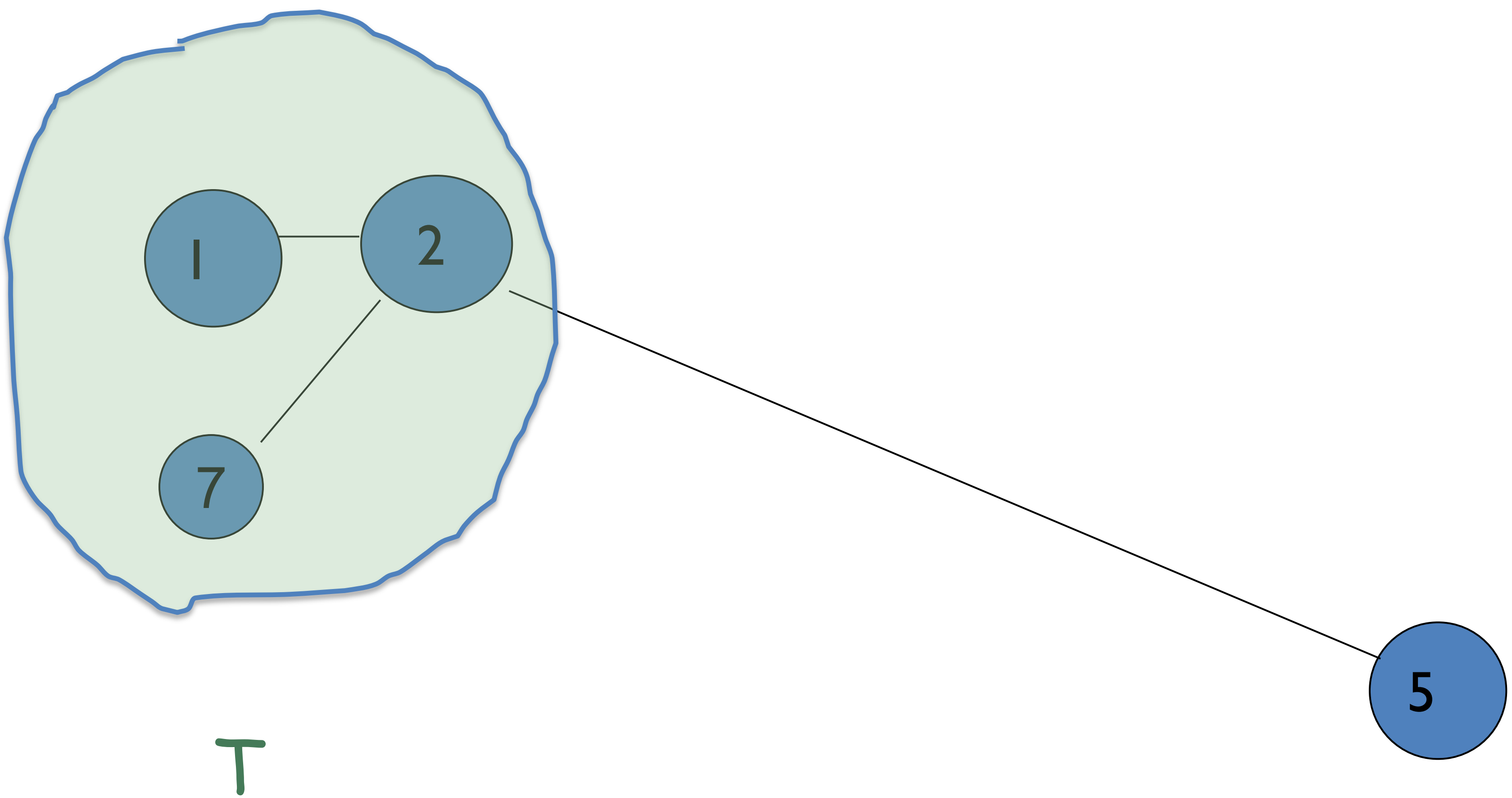


Old idea:

The sum of the degrees of nodes in a component is:

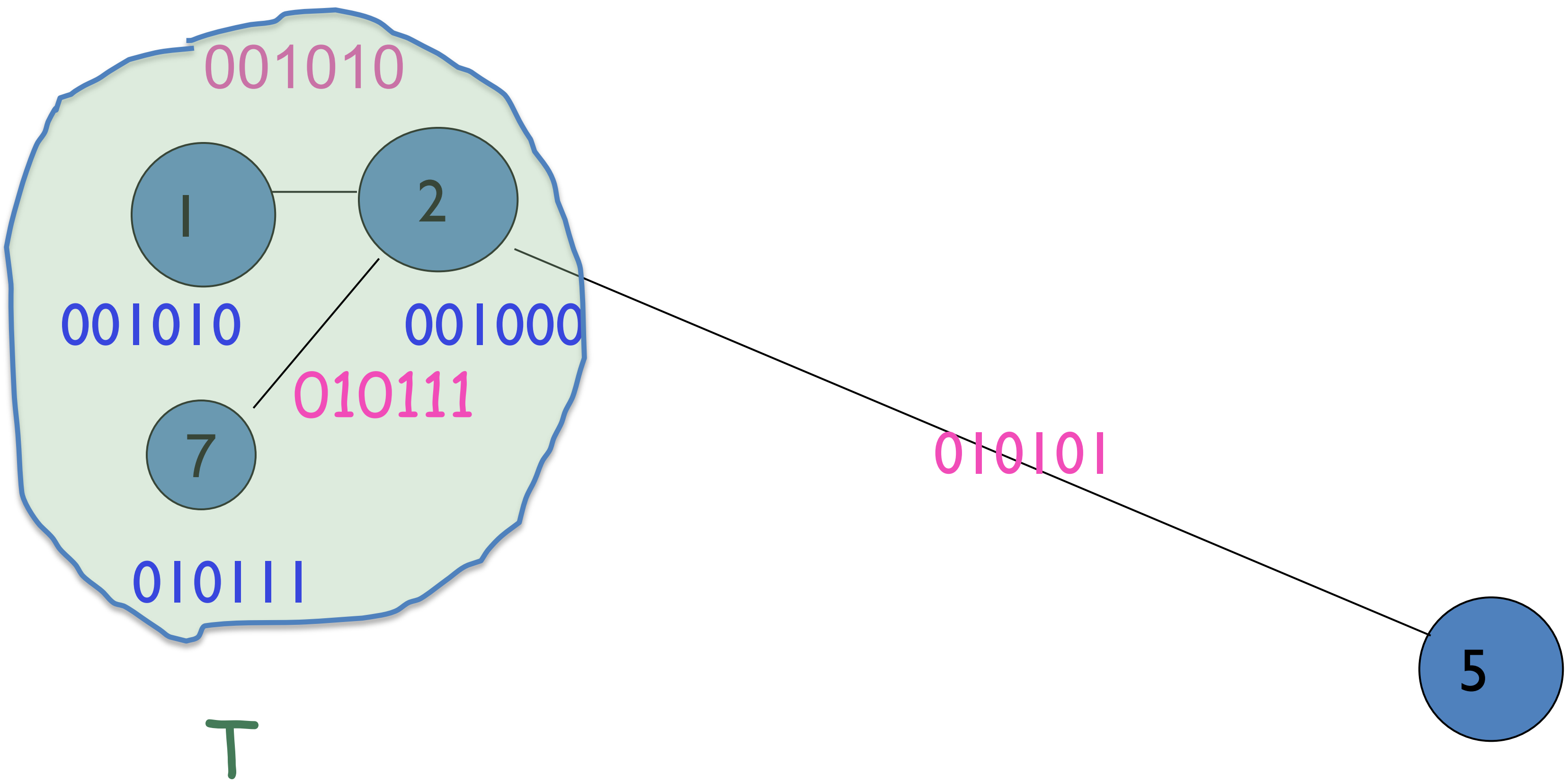
- **Even** *if it has no outgoing edges—each edge counted twice*
- **Odd** if *exactly* one edge is “leaving” component—one edge counted once

FindAny: How to find an edge leaving the tree T
if there is exactly one such edge?

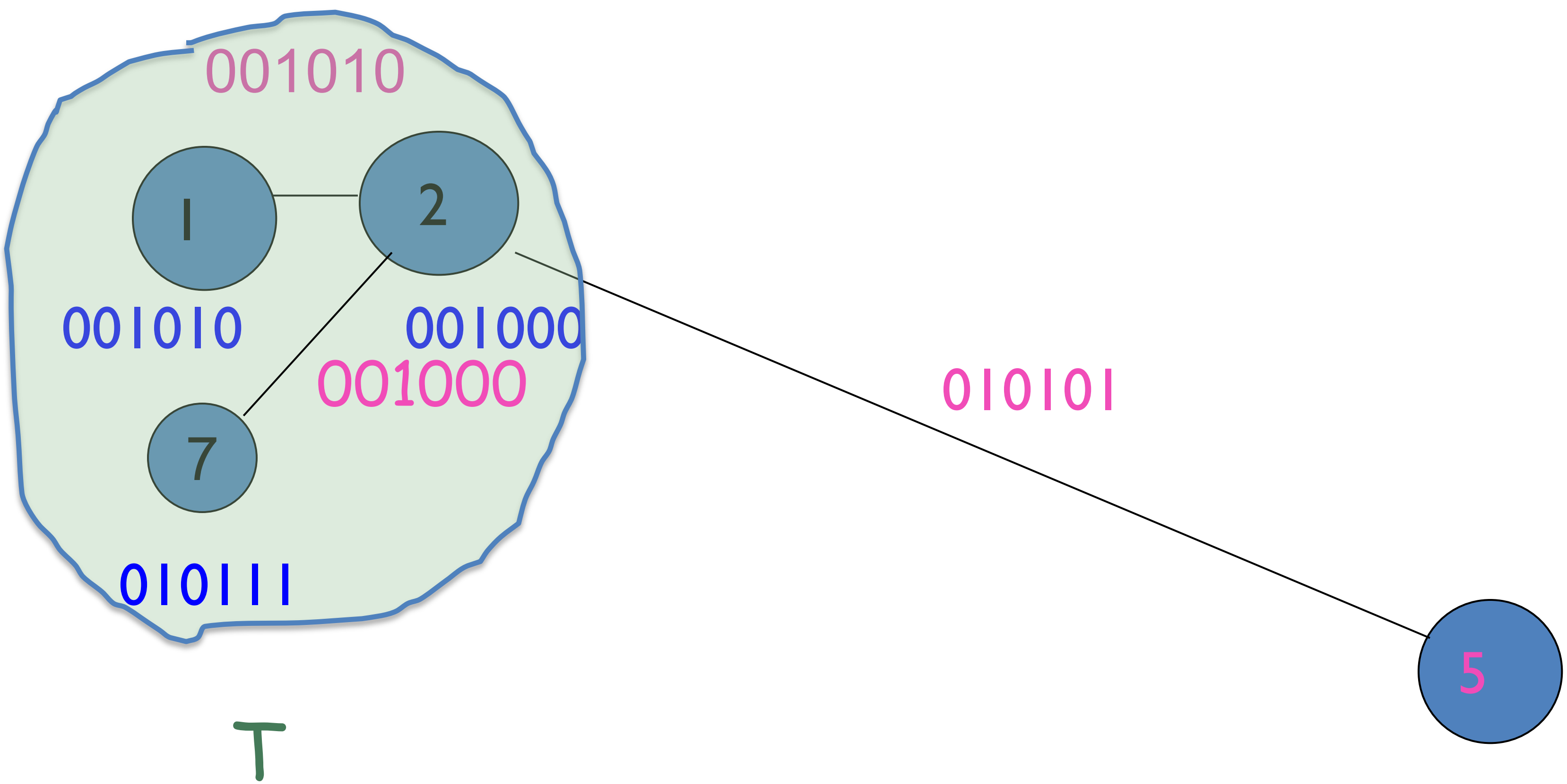


EDGE name <a,b>: a (in binary) followed by b(in binary), if a < b

For each vertex a form vector v(a) =XOR (edges incident to a)



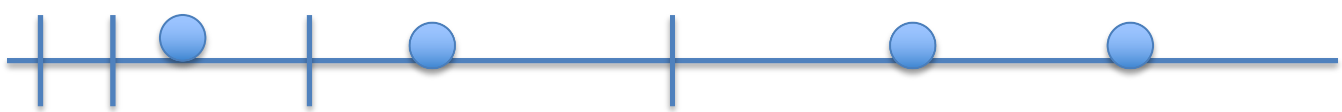
XOR of v(1) = 001010
in S +v(2) = 001000
+v(7) = 010111
=010101 = name of edge leaving S when
<u>there is exactly 1 edge in cutset</u>



To reduce cutset to size 1:

Use 2-wise independent hash to sample edges

h: [edge names] → $[0, 1,..,n^2]$
Edge e ∈ Sample **i** iff $h(e) \leq 2^i$
i=0,1,2,..,2lg n

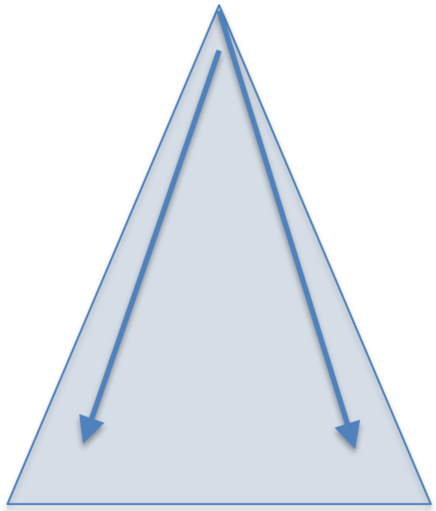


→ w/ constant prob, for i=lg |cutset|,
Sample i contains exactly 1 edge in the cutset.

Basic communication pattern:

broadcast-and-echo over tree edges

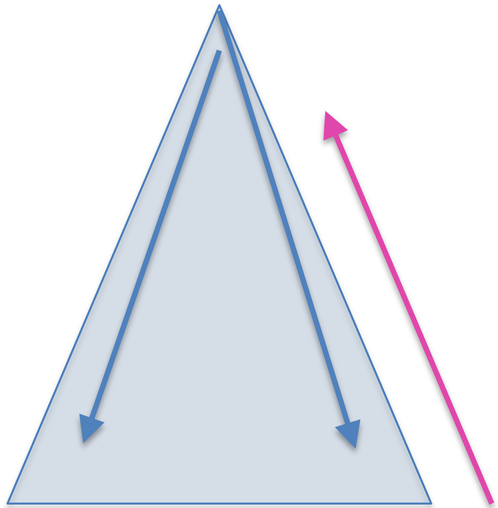
“Leader” of tree broadcasts message down tree,



Basic communication pattern:
broadcast-and-echo over tree edges

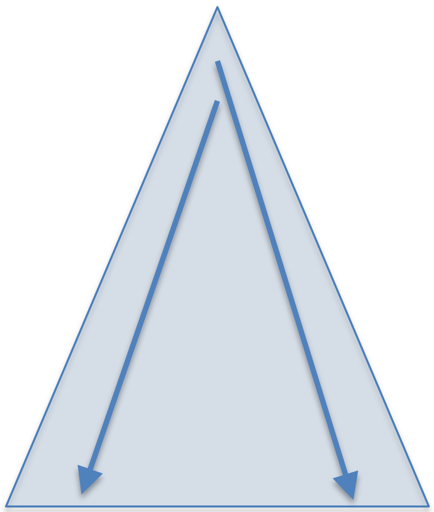
"Leader" of tree broadcasts message down tree,

Response composed from leaves back up to leader



FindAny Alg in 3 broadcasts-and-echoes: First broadcast

"Leader" of tree picks hash h and broadcasts it

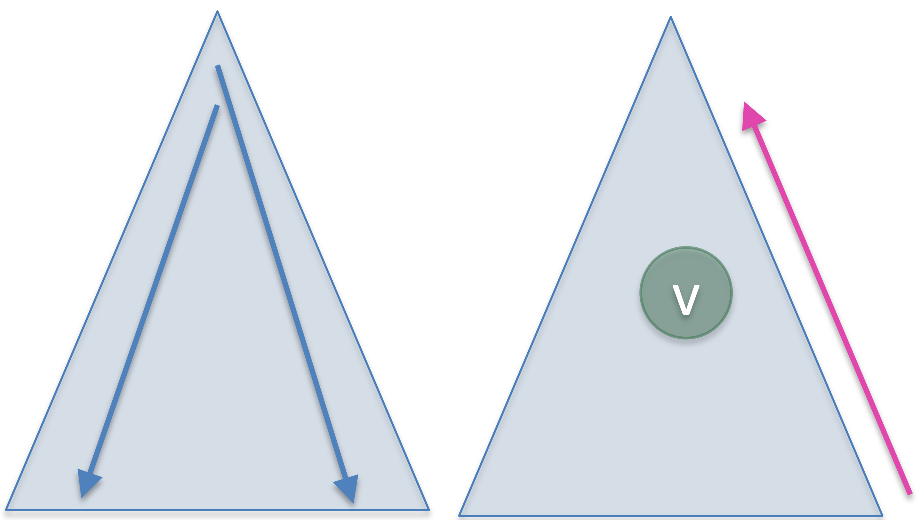


FindAny Alg in 3 broadcasts-and-echoes: first echo

Each node v computes word b(v) where i^{th} bit

b_i (v)=parity of {v's incident edges} in Sample i

Nodes in tree echo up to compute B= $XOR_{v \text{ in } T}$ b(v)



Leader computes min=minimum i s.t. i^{th} bit B_i =1

FindAny: finds edge leaving tree T in 3 broadcast-and-echoes (cont'd)

STEP 2

Leader broadcasts min

Echo: XOR names of edges e incident to all nodes in T with

$$h(e) \leq 2^{min} \text{ (i.e., in Sample min)}$$

All sampled edges which are not outgoing cancel out with XOR, leaving hopefully one edge

STEP 3. Broadcast-and-echo e: Leader <u>verifies</u> that e is name of exactly one edge in the cutset

FindAny does a lot!

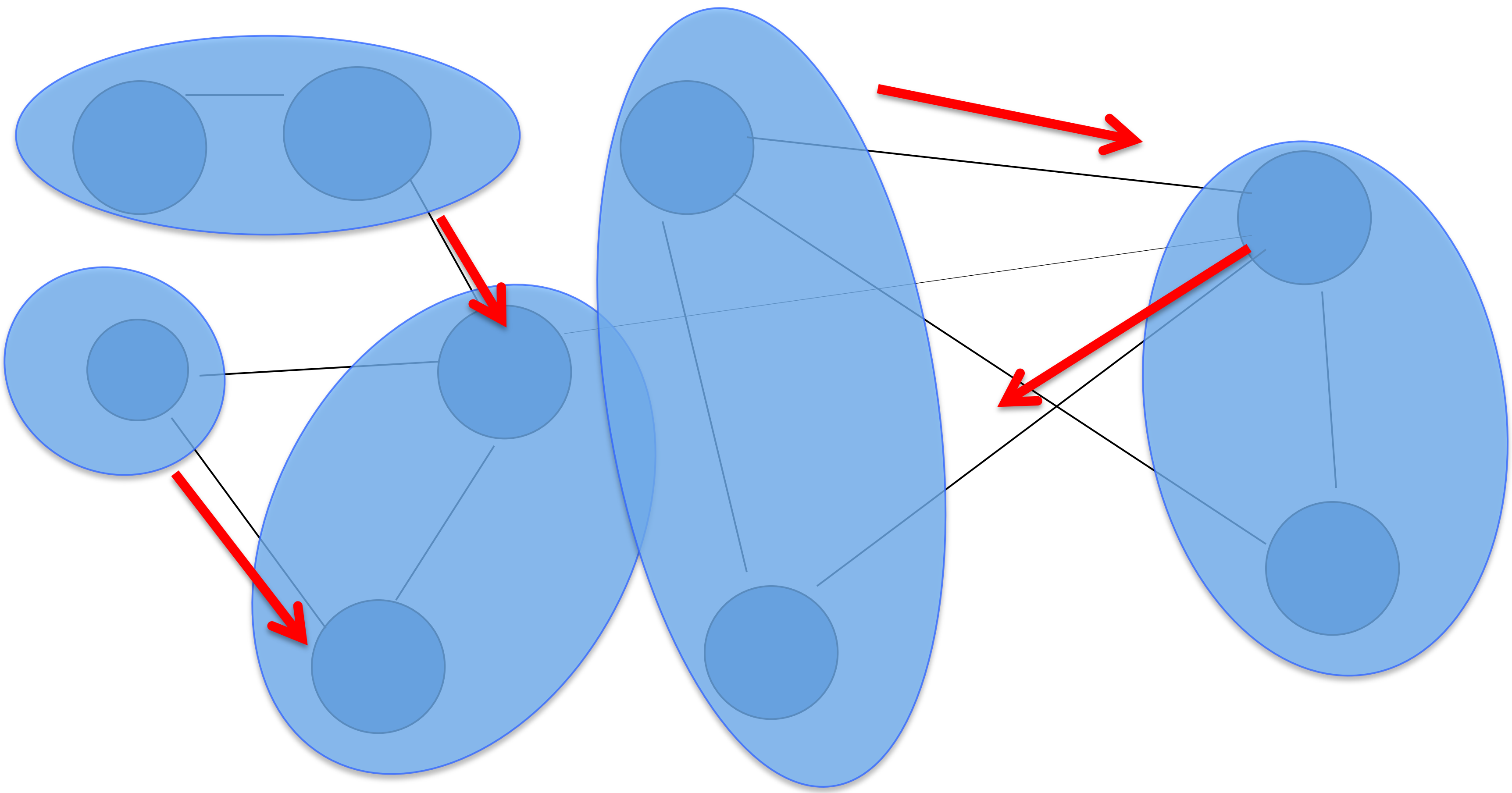
Observe:

1. With constant prob, returns an outgoing edge
2. Edge is a random edge in cutset
3. n^2/min is an estimate of the cutset size

(Observations 2,3 needed for later)

Sketch of synchronous alg

Boruvka style algorithm to build tree:
Phase: In parallel, w/constant prob each tree finds an edge leaving and merges (must prevent cycles)



Minimum spanning tree

- Modify FindAny to do binary search on edge weight ranges to find a minimum weight outgoing edge

Doesn’t work for asynchronous

- No global clock, except for initial wake-up, actions are event-driven
- → One tree will grow, one node at a time, while the others sit, for a cost of
- $\sum_{i=1}^{n} i = \mathrm{O}(\mathrm{n}^2)$

Doesn't work for asynchronous

- No global clock, except for initial wake-up, actions are event-driven
- → One tree will grow, one node at a time, while the others sit, for a cost of
- $\sum_{i=1}^{n} i = O(n^2)$

Why doesn't the Gallagher Humblet Spira (GHS) (1983) technique work?

Asynchronous alg

Ali Mashreghi, K DISC 2018

All nodes awake:

- Low degree ($< \sqrt{n} \log n$) nodes send to all their neighbors
- With prob $1/\sqrt{n}$) nodes choose themselves as stars
- stars send to all their neighbors

Grow tree T in phases, from root:

After each phase:

A. A high degree node (and a new adjacent star node) is added to T

OR

B. T is expanded until the number of outgoing edges to low degree neighbors is reduced by a half.

$\sqrt{n}$ stars $->$ $\sqrt{n}$ type A phases

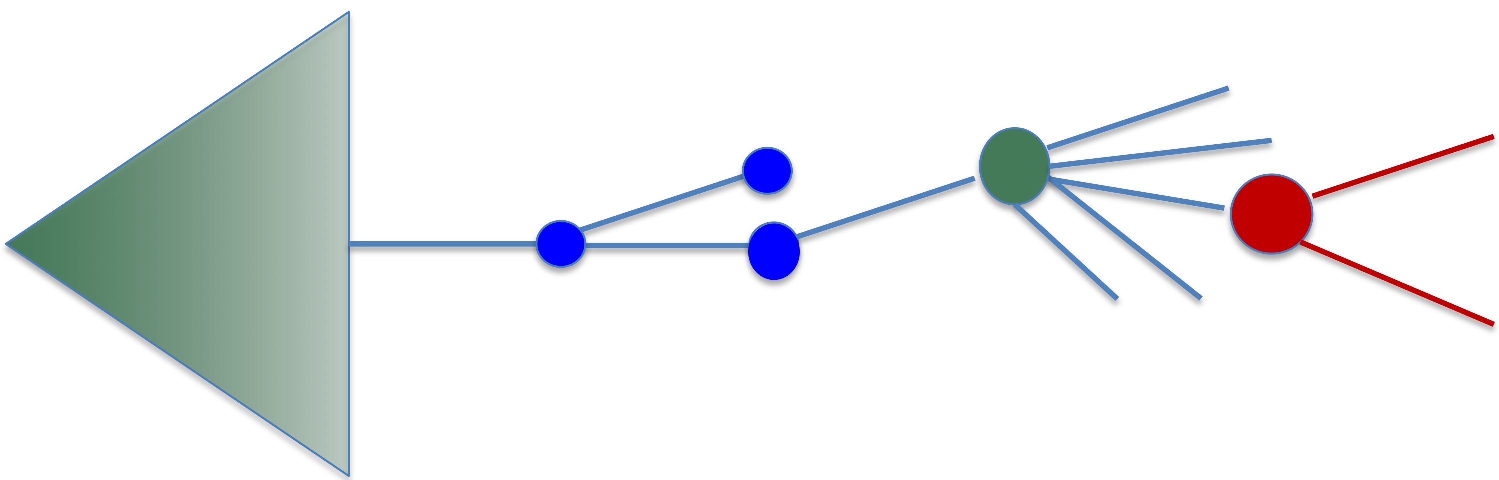
There are no more than lg n type B phases between each type A phase

-$->$ $\sqrt{n}$ lg n total phases

How to grow tree T (Step 1):

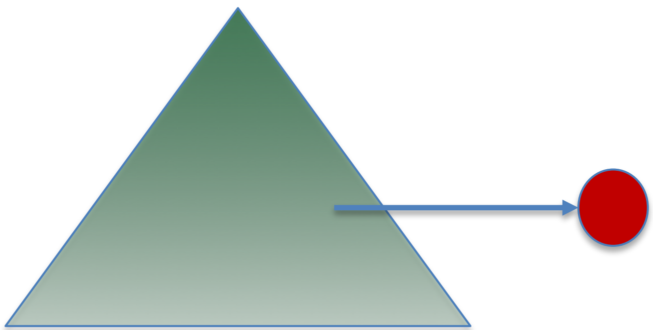
1) Expand T recursively:

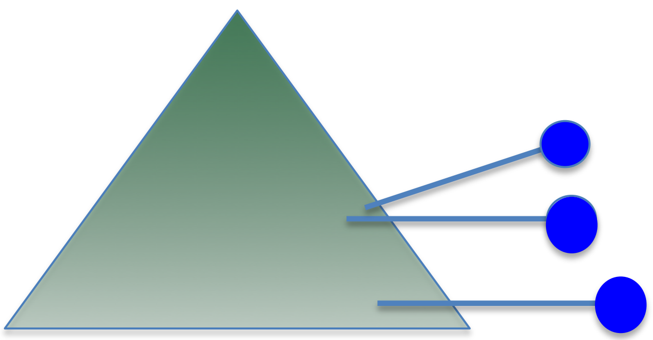
- Low degree nodes in T bring in their all neighbors
- High degree nodes in T bring in at least one Star node w.h.p and
- Star nodes bring in all their neighbors.



How to grow T (step 2)

2). Find an outgoing edge to a High degree node using FindAny OR





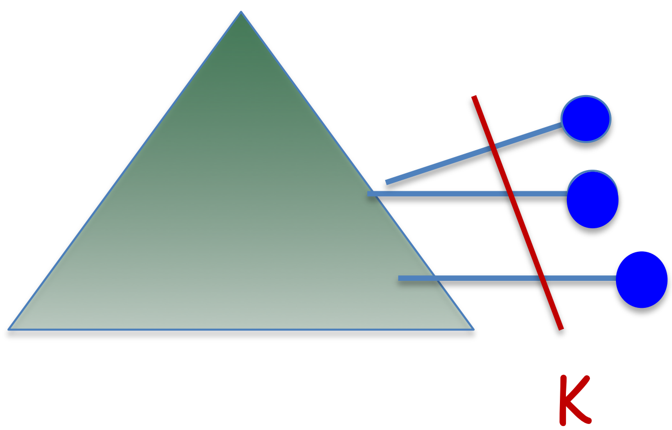
Else { FindAnys fails to find a High degree node}

WAIT until T receives messages from Low degree nodes over half the outgoing edges

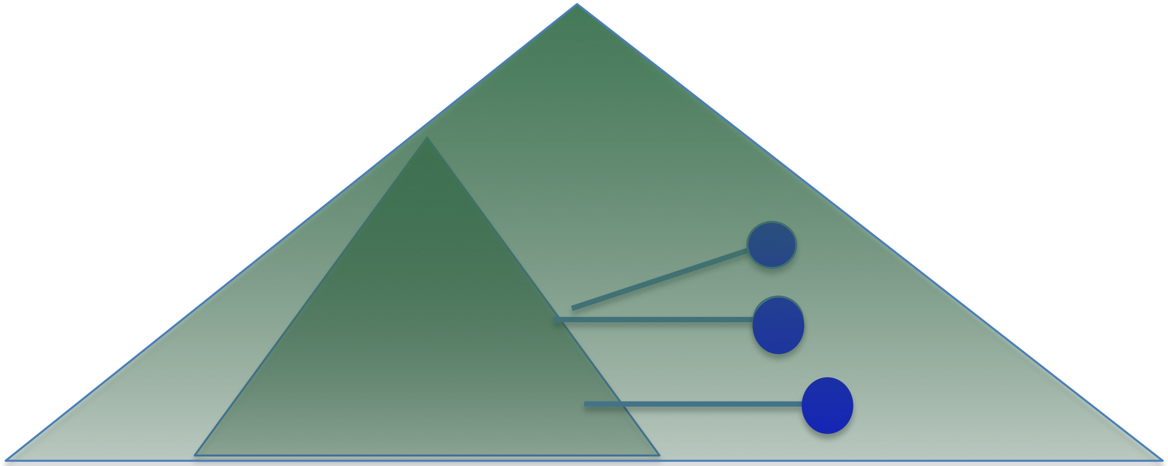
How do you WAIT asynchronously ??

Use FindAnys to estimate cutset size K

For each message sent to T, trigger new phase with prob ~2/K.



Progress



The next Expand either finds a high degree node or the number of outgoing edges to nodes with low degree is reduced by half

Analysis

- Cost per phase O(n log n) words, each of O(log n)-bits and there are O($\sqrt{n\ log\ n}$) phases
- For a total of O($(n \log n)^{3/2}$) words.

Asynchronous minimum spanning tree

1. Construct a spanning tree

2. Using it, root can synchronize the merging of fragments by level number, as in GHS

Can be done with $\tilde{O}(n \log n)$ messages.

A simpler model for understanding lower bounds?

- One-way communication, a Coordinator
- Each node sends to Coordinator, X bits
- Coordinator outputs spanning tree or connected/not connected
- With public randomness, this can be done with connectivity streaming method with

X= $O(\log^3 n)$ bits (Monte Carlo) and this is tight (Nelson, Yu 2018)

Coordinator—Public randomness =streaming technique

Public randomness is used to specify log n hash functions.

For each hash function:
Each node sends the XOR of its neighbors' names which hash to [1,2^i] range for i=0,.., 2lg n

For i=1,2, , c lg n, Coordinator uses i^{th} hash function result to find outgoing edges to compute i^{th} tier of Boruvka tree.

To verify whp that XOR is indeed an edge name, extra info must be sent by each node.

(Ahn,Guha,McGregor, SODA 2012)

Coordinator—Private randomness

Observe: Given a set of n^c elements, we can deterministically encode each element

using $O(k \log n)$ bits so that

the XOR of any subset of k elements is unique.

Sending to Coordinator:

Let k=$\sim\sqrt{n}$

- Each node sends the O(log n) bit name of each incident edge with probability ($\log^2 n$)/k

- Each node v encodes each of its incident edges {u,v} using k log n bits and sends the XOR of these encodings, XOR(v)

Coordinator:

1) Uses sampled edges to create connected components.

Note: remaining components have small cuts w.h.p.

2) For each component C, uses

XOR of coded strings from nodes in C

to determine all edges between the components

Optimizing for k

- This gives X= $\tilde{O}(n^{1/2})$ bits.
- (Holm, K, Thorup, Zwick)

Time- communication tradeoffs

In the synchronous model for MST:

- Õ(n) bits, Õ(diameter(MST)) time

  (Mashreghi, K, ICDCN 2017)

- Õ(min{$n^{3/2}$,m}) bits, Õ(diameter(G)+ $\sqrt{n}$) time

  Ghaffari, Kuhn DISC 2018

Open problems

1. Lower bound even for det algs for broadcast tree in general KT_1 model

   a) Simpler: In the one-way communication model, anything better than $\Omega(\log^3 n)$ for private randomness?

   b) Lower bounds in KT_0 and KT_1 for IDs $=\{1,2,\ldots,n\}$

Open problems (cont'd)

2. Synch vs asynch: Is there really a separation?

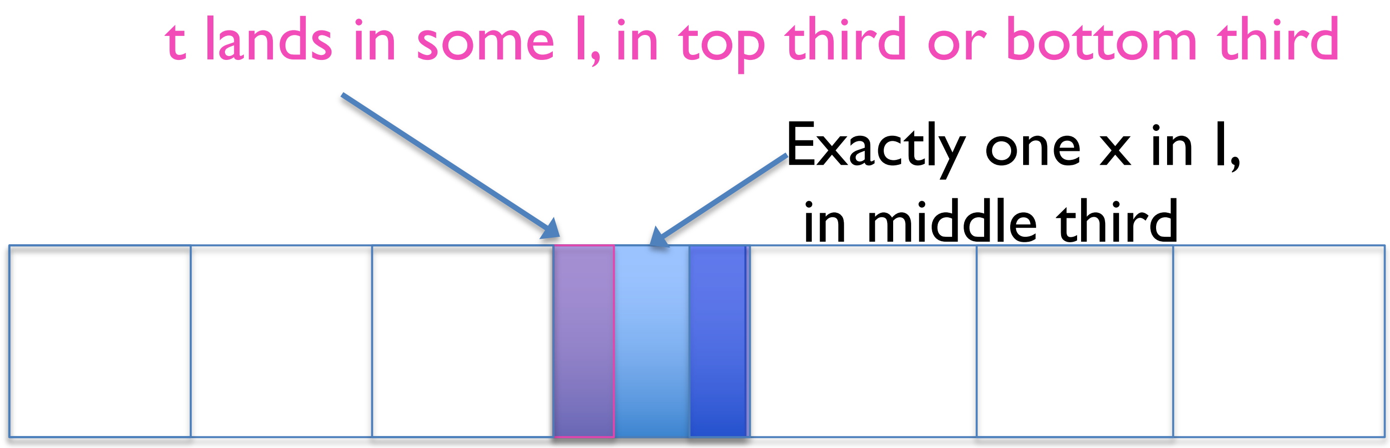
3. Can optimal time Õ(D+sqrt(n)) and Õ(n) bits be achieved? Or lower bounds on time-bit trade-offs proved?

4. Building a shortest path tree in o(m) bits?


THANK YOU. Any questions?

Why it works

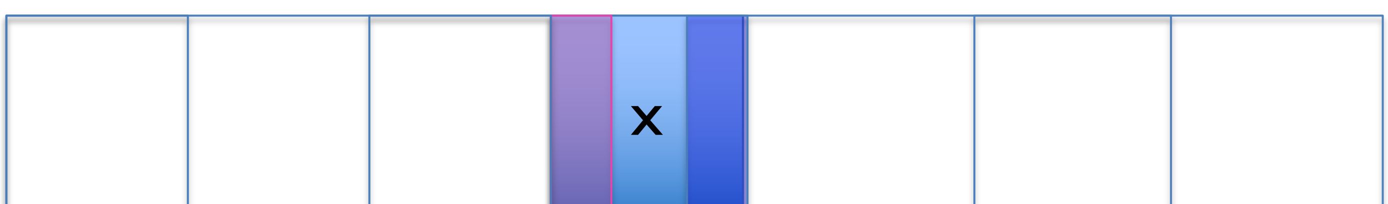
- Let S be a subset of edges
- Imagine 2|S| equal sized intervals.



Why it works

- Let S be a subset of edges
- Imagine $2|S|$ equal sized intervals.

Succeeds when parity of elements $< x$ is even and t is $> x$
Or parity of elements $< x$ is odd and t is $< x$



To summarize:

Spanning tree:

- Build in O(n log n) messages and time
- Use findany to repair an ST in expected O(n) messages, O(log n) local memory

Minimum spanning tree

Build in O(n log n/log log n) messages and time

Use findmin to repair an MST in expected O(nlogn/loglog n) messages, Olog (n) local memory

Open problems and discussion

- Is $\Omega(m)$ communication required for building tree in asynchronous model?
- Prove separation of communication cost of findmin from findany
- Time v. communication tradeoffs
- Deterministic?
- Practical implementation: distances not exactly symmetric, topology, etc.

Findmin weight edge leaving

Two Tests to see if there is an edge leaving a comp:

1. Constant prob. Test out: 1 broadcast, 1 1-bit echo
2. High prob. **HP-**Test out*:* 1 broadcast, 1 *log n*-bit echo

Findmin weight edge leaving

1. In parallel, with 1 Broadcast-and-echo
   do log n-wise search on weights with
   TestOut ‘s to find smallest wt interval

2. Verify its minimality with HP TestOut

3. Repeat if wrong or recurse on that interval

Testout

randomly choose a function

F: edges$\rightarrow\{0,1\}$

s.t. for a nonempty set S,

w. constant prob. >0,

an ODD number of S's elements map to 1.

Simple F (Thorup)

F has two parts,

- a 2-wise indep.hash function

$$h: U \rightarrow U$$

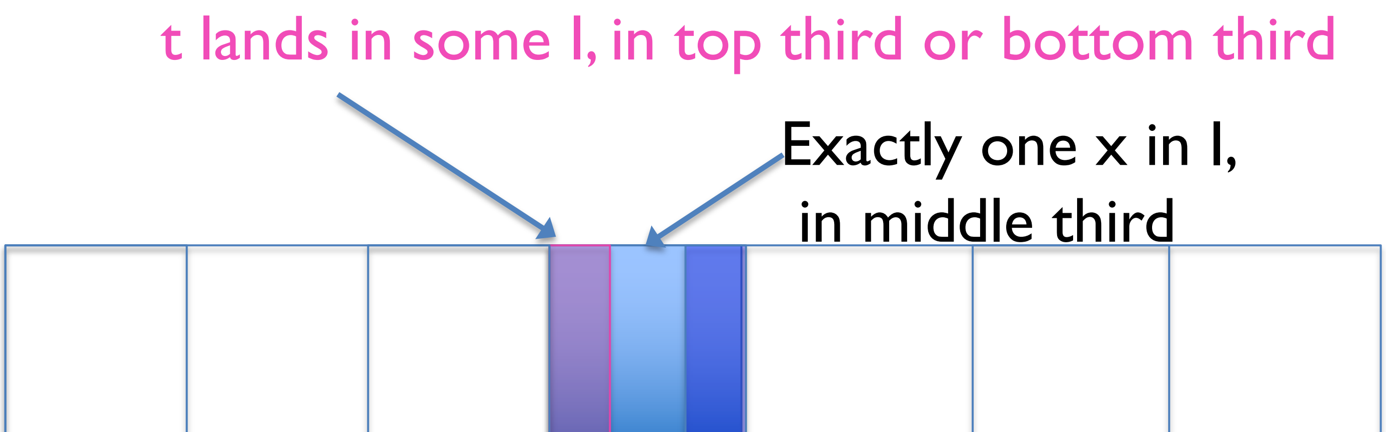
- t, a random element of U

$\boxed{F(s)=1 \text{ iff } h(s) < t}$

F can be described in O(log |U|) bits.

Why it works

- h hashes U→U, S subset of U
- Imagine 2|S| equal sized intervals.



HP TestOut

- Repeat Testout in parallel O(log n) times to get prob error $1/n^c$?
- Would need to send clog n hash functions

Or use deterministic amplification

Or…

Test 2: Set equality method to get HP-TestOut

A tree T is maximal iff

the set Out(T) of edges leaving all nodes in T

= the set In(T) of edges entering all nodes in T

Test equality of two sets using polynomial ID-testing O(log n) bits of communication

$f(x)=\Pi_{a \text{ in Out}} (x-a)$

$g(x)=\Pi_{b \text{ in In}} (x-b)$

Does $f(x)=g(x)$?

(Schwartz-Zippel):

Set $x=$random α in $Z_{p,}$ p be a prime, $p > n^{c+2}$

compute over Z_p:

$\Pr(f(\alpha)-g(\alpha))=0$ if In $\neq$ Out

$= \Pr[\alpha= \text{root}]$

$= \#\text{roots}/p< n^2 / p = 1/n^c$

Implementation (1 broadcast-and-echo)

- Leader broadcasts α
- Each node v computes $f_v(\alpha)$ and $g_v(\alpha)$ for its incident edges
- Starting at the leaves, pass to parent

  $f_v(\alpha) * \pi_c\, f_c(\alpha)$, c child of v

  g is computed similarly
- Leader (root) computes f-g.
  - If 0, Testout is true, else false.

Open problem and discussion

- Is $\Omega(m)$ communication required for **building MST in asynchronous model?**
- Lower bound for findmin
- Time v. communication tradeoffs
- Applications to map reduce etc.

- Thank you

Algorithms for MST

Gallager, Humblet and Spira (JACM 1983)

- *asynchronous*
- works if starting with any number of nodes wake
- time $O(n^2)$, or $O(n \log n)$ if all nodes wake at start

$\tilde{O}(\sqrt{n}+\text{diameter})$ time in a synchronous model is possible with $\tilde{O}(m)$ communication. (Pandurangan, Robinson, Scquizzato, STOC 2017)

Classic Result for MST: 1983: O(m + nlogn) bits

Gallager, Humblet and Spira (JACM)

- asynchronous
- works if starting with any number of nodes wake
- time $O(n^2)$, or $O(n \log n)$ if all nodes wake at start

$\tilde{O}(\sqrt{n}+\text{diameter})$ time in a synchronous model is possible with $\tilde{O}(m)$ communication. (Pandurangan, Robinson, Scquizzato, STOC 2017)

Coordinator—Public randomness

Public randomness is used to specify log n pairwise independent hash functions.

For each hash function:
Each node sends the XOR of its neighbors' names which hash to [1,2^i] range for i=0,.., 2lg n

For i=1,2, , lg n, Coordinator uses i^{th} hash function result to find outgoing edges to compute i^{th} tier of Boruvka tree.

To verify whp that XOR is indeed an edge name, extra info must be sent by each node.

Coordinator—Private randomness

Observe: Given a set of n^c elements, we can encode each element using $\tilde{O}$(k) bits so that the XOR of any subset of k elements is unique.

Coordinator—private randomness

k=$\sqrt{n}$

- Each node sends the name of each incident edge with probability (log n)/k

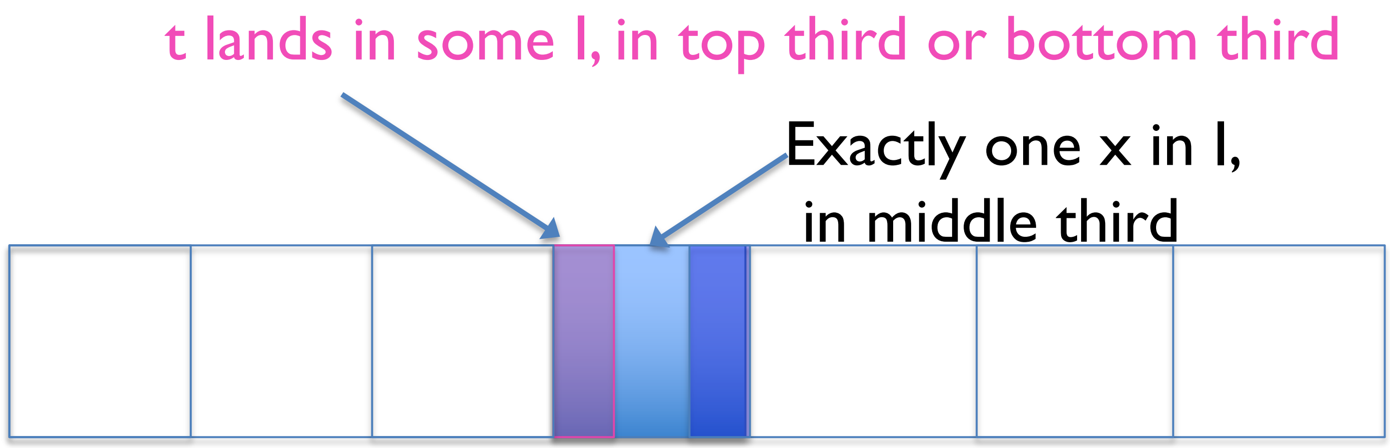
- Each node v encodes each of its edges {u,v} as a

  O(k log n) bit string and sends the XOR of these
encodings.

- Coordinator uses sampled edges to merge components then takes the $XOR_v(XOR(v))$ to determine the edges in the (small) cuts between them.

Why it works

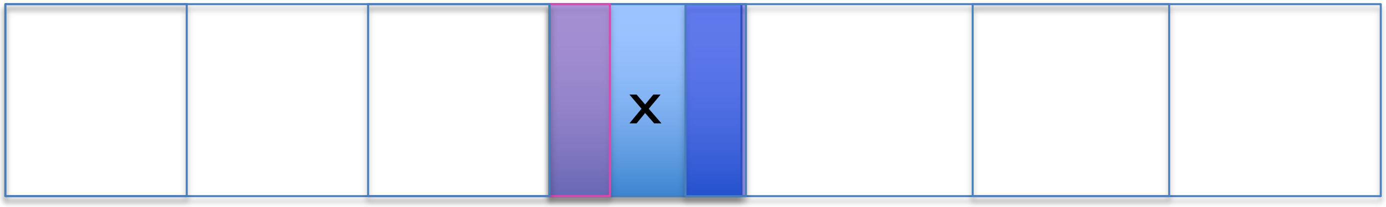
- Let S be a subset of edges
- Imagine 2|S| equal sized intervals.



Why it works

- Let S be a subset of edges
- Imagine $2|S|$ equal sized intervals.

Succeeds when parity of elements< x is even and t is > x
Or parity of elements < x is odd and t is < x



To summarize:

Spanning tree:

- Build in O(n log n) messages and time
- Use findany to repair an ST in expected O(n) messages, O(log n) local memory

Minimum spanning tree

Build in O(n log n/log log n) messages and time

Use findmin to repair an MST in expected O(nlogn/loglog n) messages, Olog (n) local memory

Open problems and discussion

- Is $\Omega(m)$ communication required for building tree in asynchronous model?
- Prove separation of communication cost of findmin from findany
- Time v. communication tradeoffs
- Deterministic?
- Practical implementation: distances not exactly symmetric, topology, etc.

Findmin weight edge leaving

Two Tests to see if there is an edge leaving a comp:

1. Constant prob**.** Test out: 1 broadcast, 1 1-bit echo
2. High prob. **HP-**Test out*:* 1 broadcast, 1 *log n*-bit echo

Findmin weight edge leaving

1. In parallel, with 1 Broadcast-and-echo
   do log n-wise search on weights with
   TestOut 's to find smallest wt interval

2. Verify its minimality with HP TestOut

3. Repeat if wrong or recurse on that interval

Testout

randomly choose a function

F: edges $\rightarrow$ {0,1}

s.t. for a nonempty set S,

w. constant prob. >0,

an ODD number of S's elements map to 1.

Simple F (Thorup)

F has two parts,

- a 2-wise indep.hash function

$$h: U \rightarrow U$$

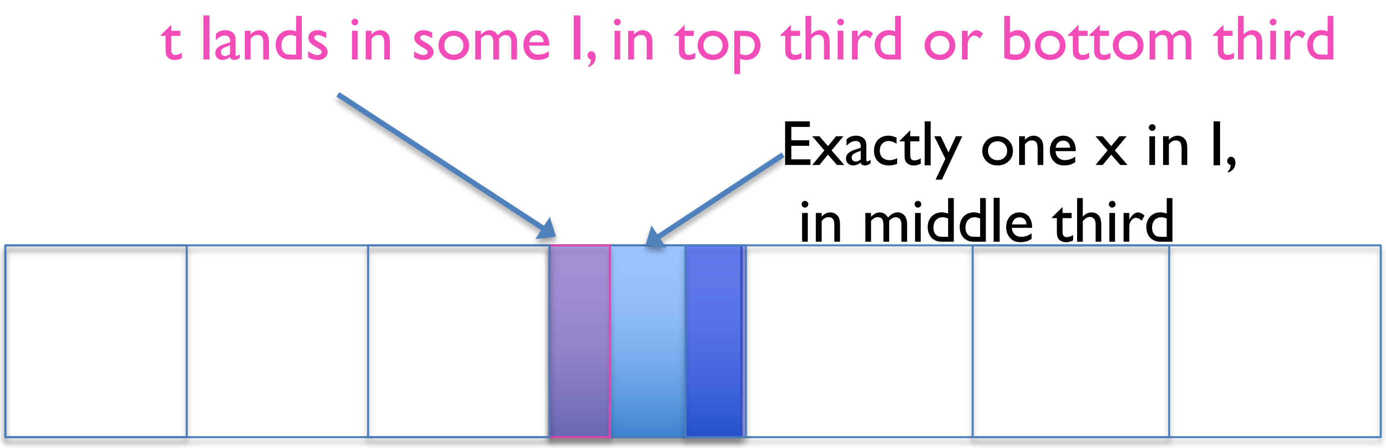
- t, a random element of U

$\boxed{F(s)=1 \text{ iff } h(s) < t}$

F can be described in O(log |U|) bits.

Why it works

- h hashes U→U, S subset of U
- Imagine 2|S| equal sized intervals.



HP TestOut

- Repeat Testout in parallel O(log n) times to get prob error $1/n^c$?
- Would need to send clog n hash functions

Or use deterministic amplification

Or…

Test 2: Set equality method to get HP-TestOut

A tree T is maximal iff

the set Out(T) of edges leaving all nodes in T

= the set In(T) of edges entering all nodes in T

Test equality of two sets using polynomial ID-testing
O(log n) bits of communication

$f(x)=\Pi_{a \text{ in Out}} (x-a)$

$g(x)=\Pi_{b \text{ in In}} (x-b)$

Does $f(x)=g(x)$?

(Schwartz-Zippel):

Set $x=$random α in $Z_{p,}$ p be a prime, $p > n^{c+2}$

compute over Z_p:

$\Pr(f(\alpha)-g(\alpha))=0$ if In $\neq$ Out

$= \Pr[\alpha= \text{root}]$

$= \#\text{roots}/p< n^2 / p = 1/n^c$

Implementation (1 broadcast-and-echo)

- Leader broadcasts α
- Each node v computes $f_v(\alpha)$ and $g_v(\alpha)$ for its incident edges
- Starting at the leaves, pass to parent

  $f_v(\alpha) * \pi_c f_c(\alpha)$, c child of v

  g is computed similarly
- Leader (root) computes f-g.
  - If 0, Testout is true, else false.

Open problem and discussion

- Is $\Omega(m)$ communication required for **building MST in asynchronous model?**
- Lower bound for findmin
- Time v. communication tradeoffs
- Applications to map reduce etc.

- Thank you

Algorithms for MST

Gallager, Humblet and Spira (JACM 1983)

- *asynchronous*
- works if starting with any number of nodes wake
- time $O(n^2)$, or $O(n \log n)$ if all nodes wake at start

$\tilde{O}(\sqrt{n}+\text{diameter})$ time in a synchronous model is possible with $\tilde{O}(m)$ communication. (Pandurangan, Robinson, Scquizzato, STOC 2017)

Classic Result for MST: 1983: O(m + nlogn) bits

Gallager, Humblet and Spira (JACM)

- asynchronous
- works if starting with any number of nodes wake
- time $O(n^2)$, or $O(n \log n)$ if all nodes wake at start

$\tilde{O}(\sqrt{n}+\text{diameter})$ time in a synchronous model is possible with $\tilde{O}(m)$ communication. (Pandurangan, Robinson, Scquizzato, STOC 2017)


Thanks!