Cooperative Graph Analytics for Autonomous Data Centers

Jon Berry (Sandia National Laboratories)
Mike Collins (Christopher Newport University)
Aaron Kearns (U. New Mexico)
Cynthia A. Phillips (Sandia National Laboratories)
Jared Saia (U. New Mexico)
Randy Smith (Sandia National Laboratories

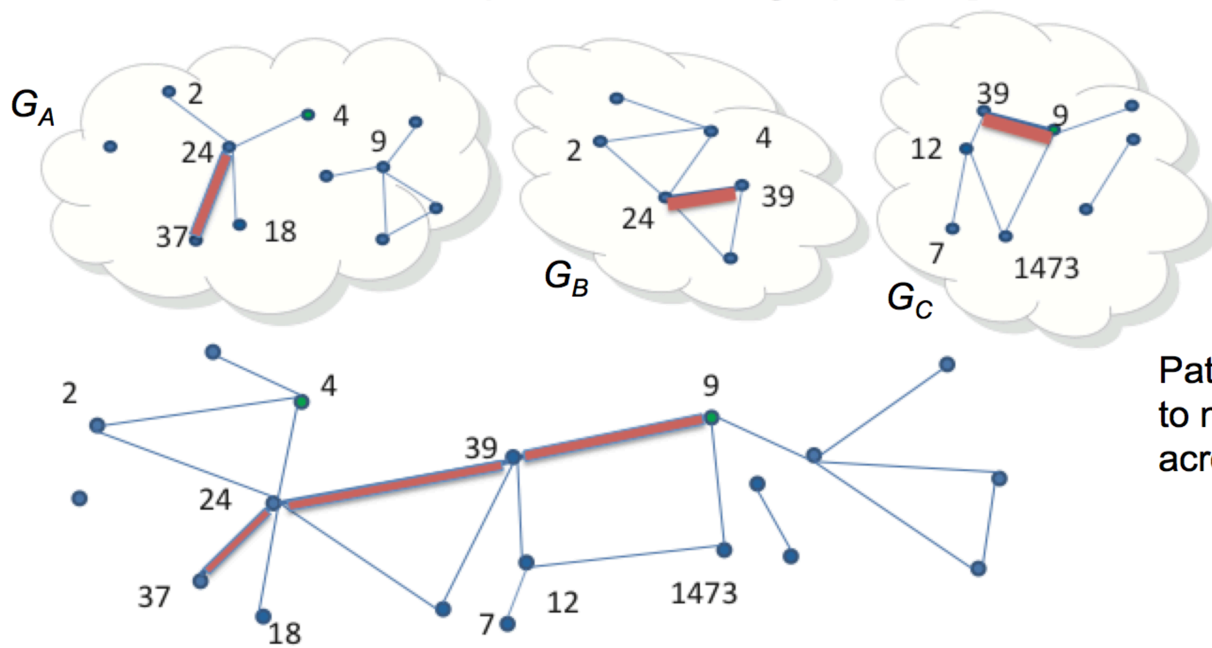
Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

A New Distributed Computing Model

Alice and Bob (or more) independently create social graphs G_A and G_B.

- Alice and Bob each know nothing of the other's graph.
- Shared namespace. Overlap at nodes.

Goal: Cooperate to compute algorithms over G_A union G_B with limited sharing: $O(\log^k n)$ total communication for size n graphs, constant k



s-t connectivity
For 3 centers

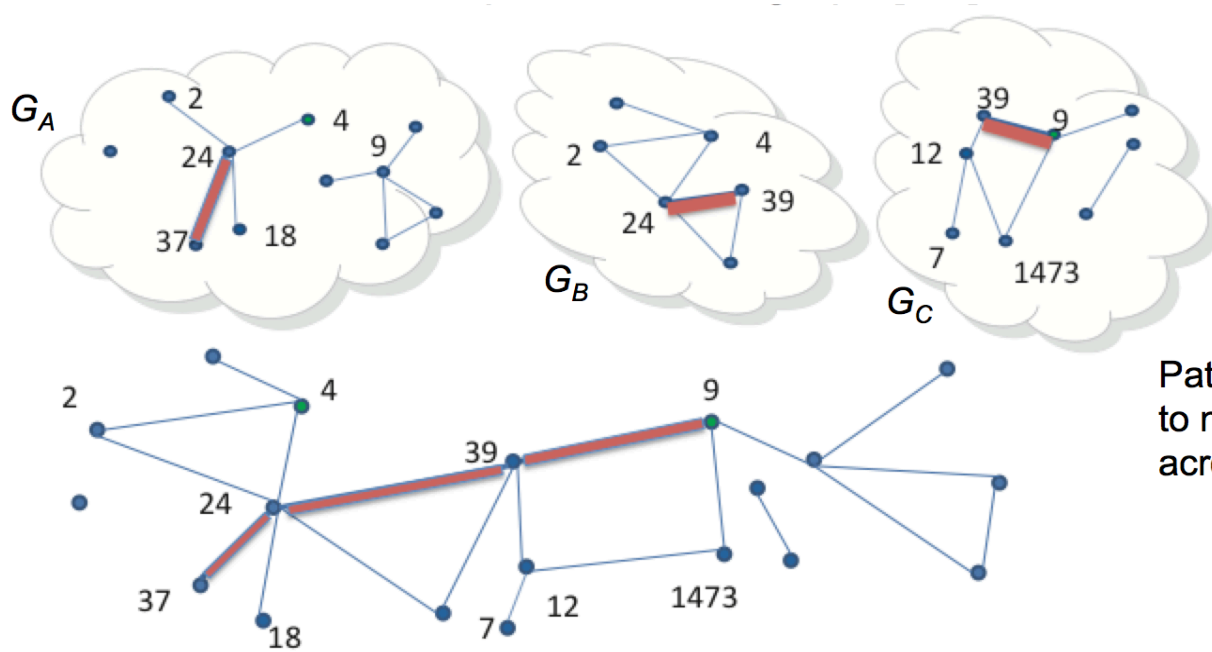
Path from node s=37
to node t=9 is split
across all three centers

Another Limited Sharing Model

Goal: Cooperate to compute algorithms over $G_A \cup G_B (\cup G_C \ldots)$

Alice gets no information beyond answer in honest-but-curious model.

- Secure multiparty computation
  - Few players, large data



s-t connectivity
For 3 centers

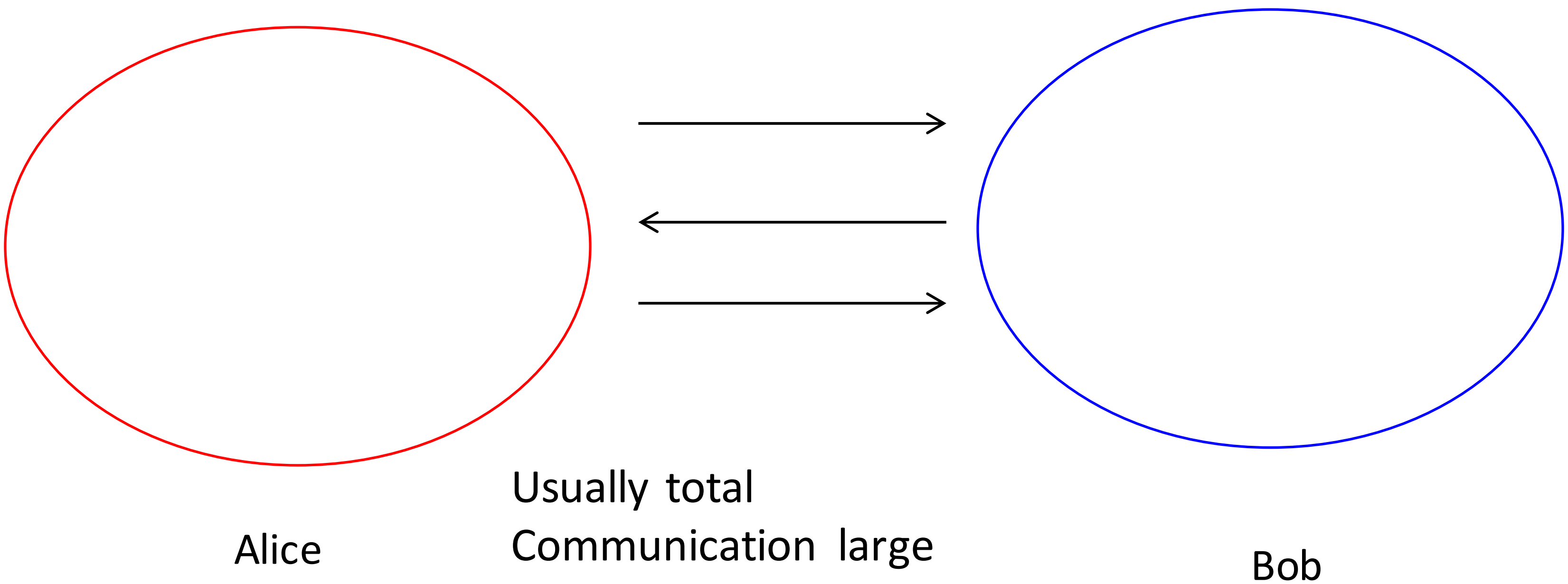
Path from node s=37
to node t=9 is split
across all three centers

Motivation

- Company mergers
- National security: connect-the-dots for counterterrorism
- Nodes are people
  - Exploit structure of social networks

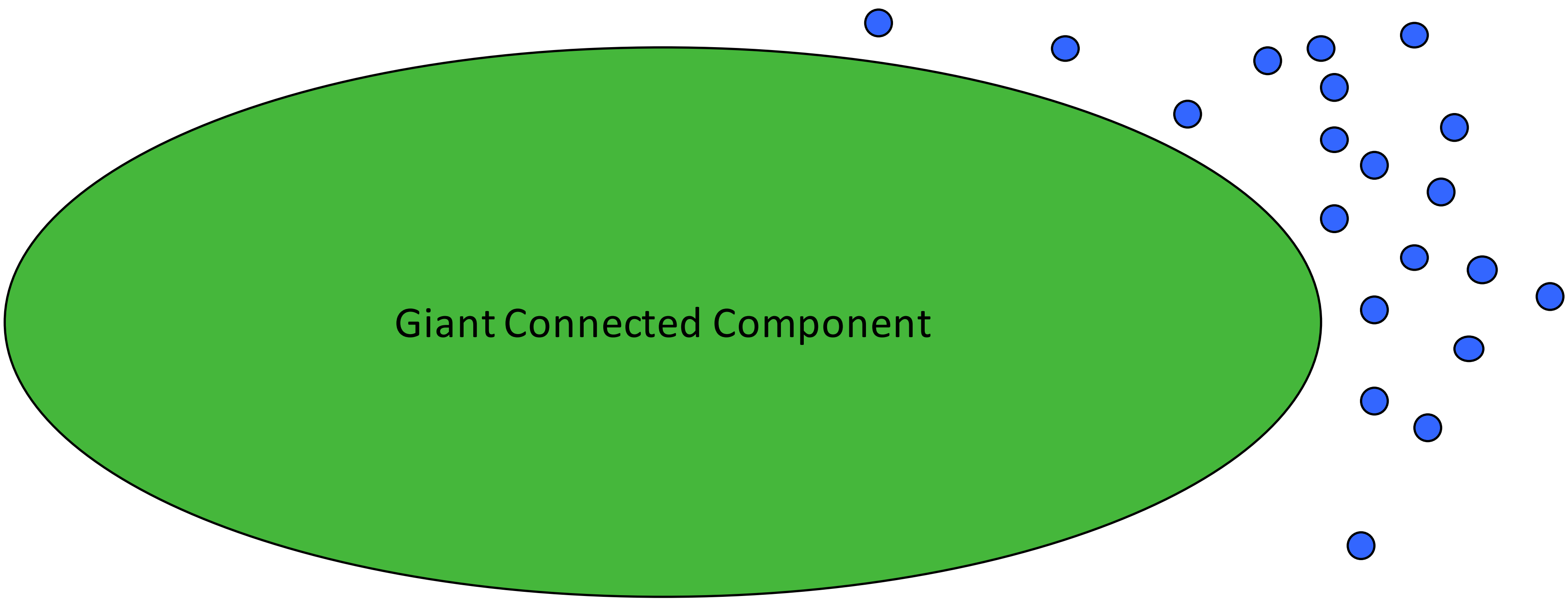
Result: Low-Communication s-t Connectivity

- s-t connectivity for social graphs: $O(\log^2 n)$ bits for n-node graphs
- $\Omega(n \log n)$ lower bound for general graphs (Hajnal, Maass, Turàn)
  - Edges partitioned, 2 parties



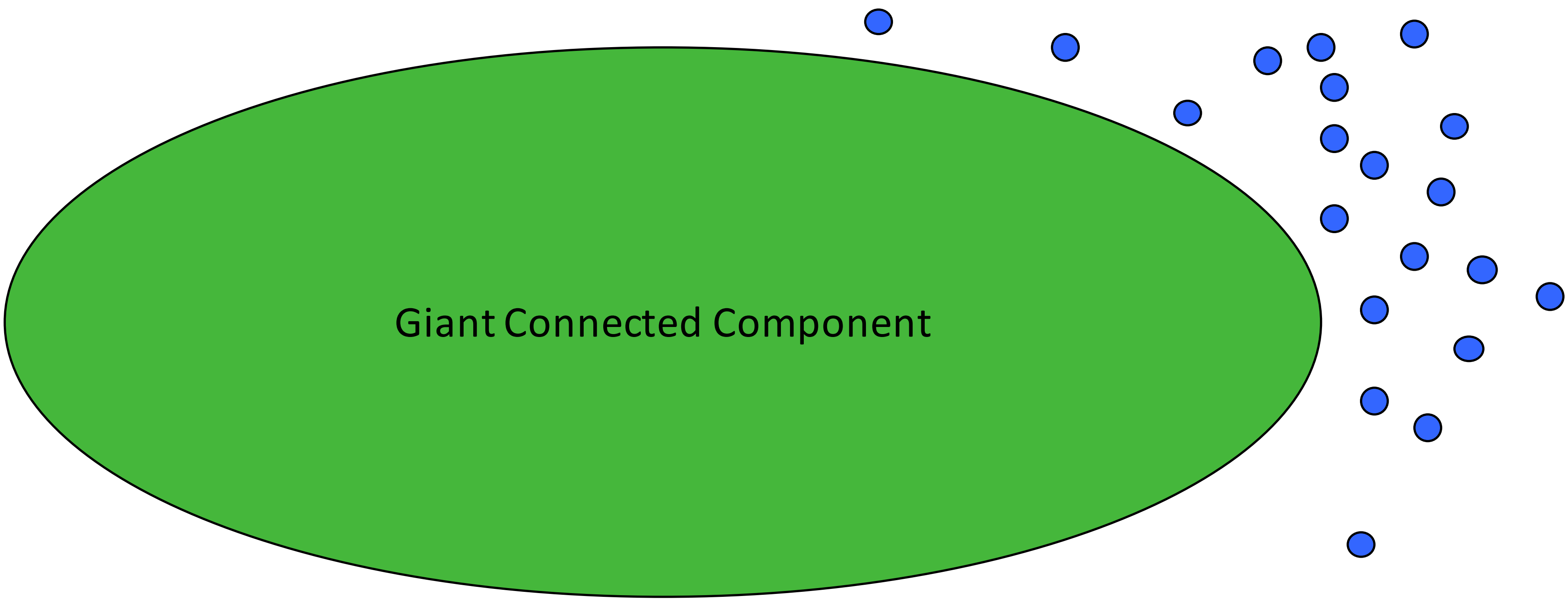
Social Network Structure

- Social networks have a giant component: second smallest component of size O(log n)



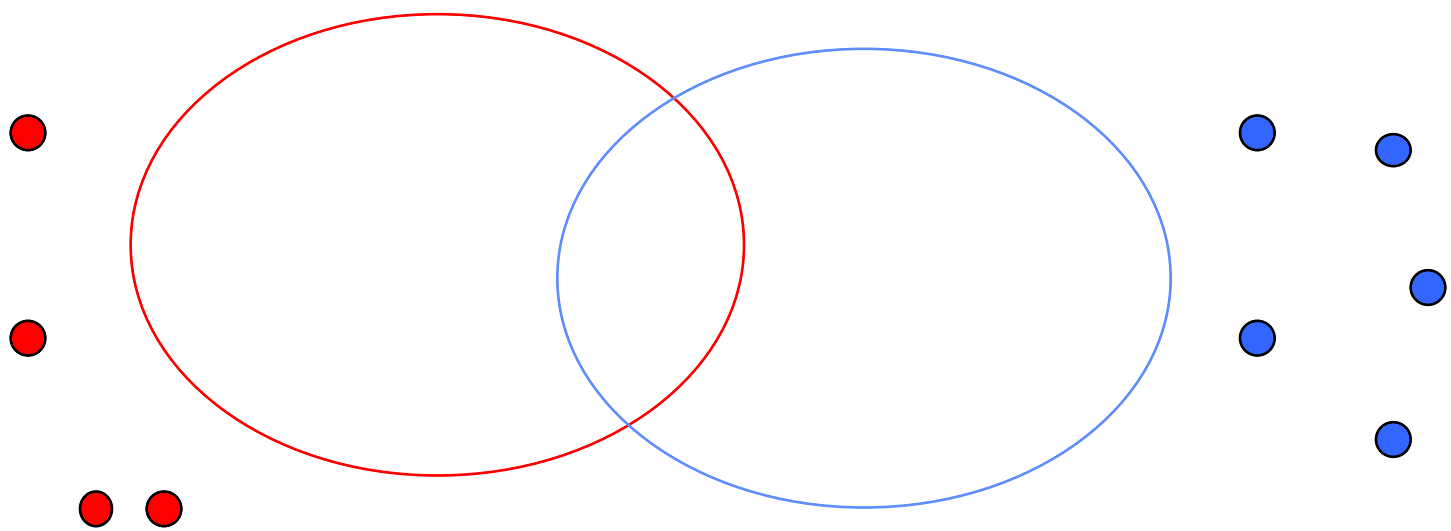
Social Network Structure

- Normal connection growth (Easley and Kleinberg)
- Observed in social networks (long distance phone call, linkedin, etc)
- Theoretically in Chung-Lu graphs with power law exponent between 1+ε and 3.47



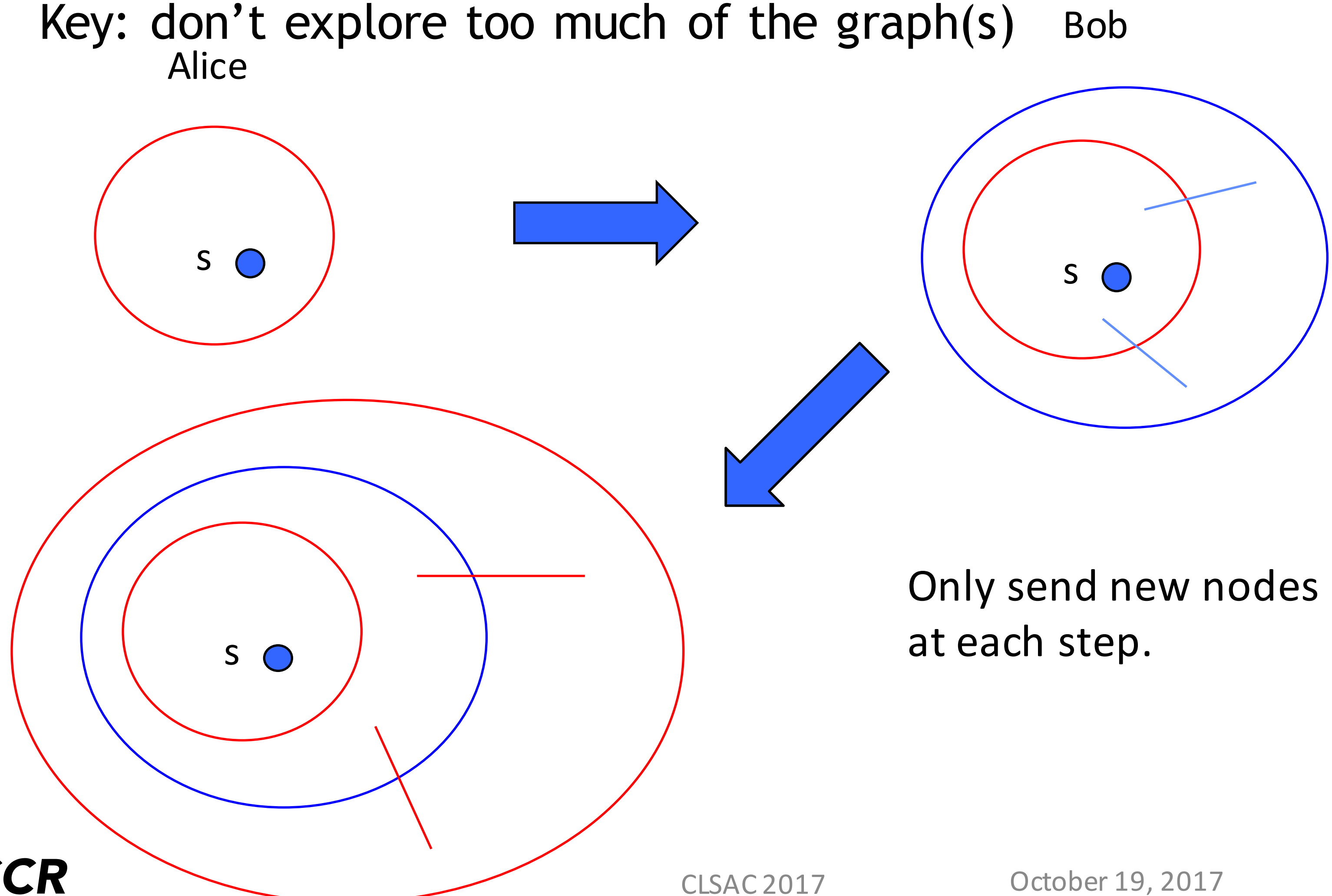
Assumptions

- Alice's graph G_A and Bob's graph G_B both have giant components
- These giant components intersect
  - Can verify with $O(\log^2 n)$ communication with high probability if intersect by a constant fraction (say 1%)



Shell Expansion

- Like breadth-first-search, “layer” is connected piece in G_A or G_B
- Key: don’t explore too much of the graph(s)



Only send new nodes at each step.

Low-Sharing s-t Connectivity Algorithm

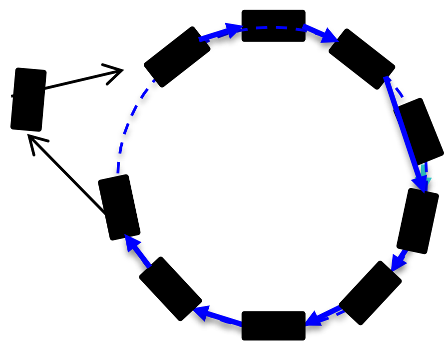
- Alice and Bob agree on a value γ (polylog in n)
  - Algorithm is correct iff γ at least size of 2^{nd} largest component
- Do shell expansion (BFS) from both s and t
- Stopping criteria:
  1. s shell merges with t shell (yes)
  2. No new nodes added in some step (no)
  3. Shells merge with giant component of G_A or G_B (yes)
  4. Shell size exceeds γ. Stop before sending. (yes)

- With a good guess, $\gamma = O(\log n)$, so $O(\log^2 n)$ bits communicated

More Than Two Centers

- Do shell expansion in a loop
- Center that adds a node removes it when it comes back (so each center sees it once)

Query processor



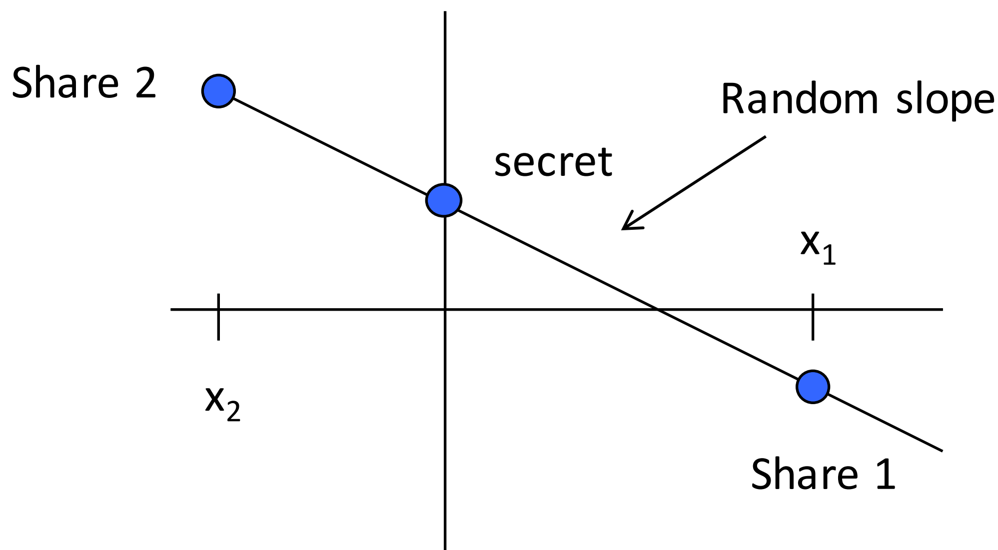
- The query processor starts both the s and t shells (containing only the one node if necessary
- Looks like the 2-processor protocol with all the other processors merged.

Secure Multiparty Computation Version

- Alice and Bob can determine that a path connects s and t without revealing anything about: the path, nodes seen by either party
- Similar to a model used by Brickell and Shmatikov
  - They assume known node names (shared customer lists)
- Secure multiparty computation
  - Usually many parties, small data (circuits, oblivious RAM)
    - Millionaire's problem
    - Beet farmers
  - We have small number of parties, large data

Tool #1

- Secret sharing
  - Secrets are in a finite field
  - Use a polynomial of degree d to encode a value, d+1 shares
    - All shares reveal secret, d reveals nothing
    - Solution is y intercept, secrets are polynomials at other x
- Key: Given a share of x (called $[x]_i$) and a share of y (called $[y]_i$), can get a share of the sum by adding shares: $[x+y]_i = x_i + y_i$



Tool #2: Secure MUX

$$\mathrm{MUX}(c,a,b) = \begin{cases} a, & c \neq 0, \\ b, & \text{otherwise.} \end{cases}$$

- Need to be able to securely compute shares of MUX(c,a,b), given shares of a,b,c
- Information-theoretically secure protocols if at least 3 centers (Ben-or, Goldwasser, Wigderson)
- For 2 centers need Yao's garbled circuits (crytographic)
- This is expensive, requires communication

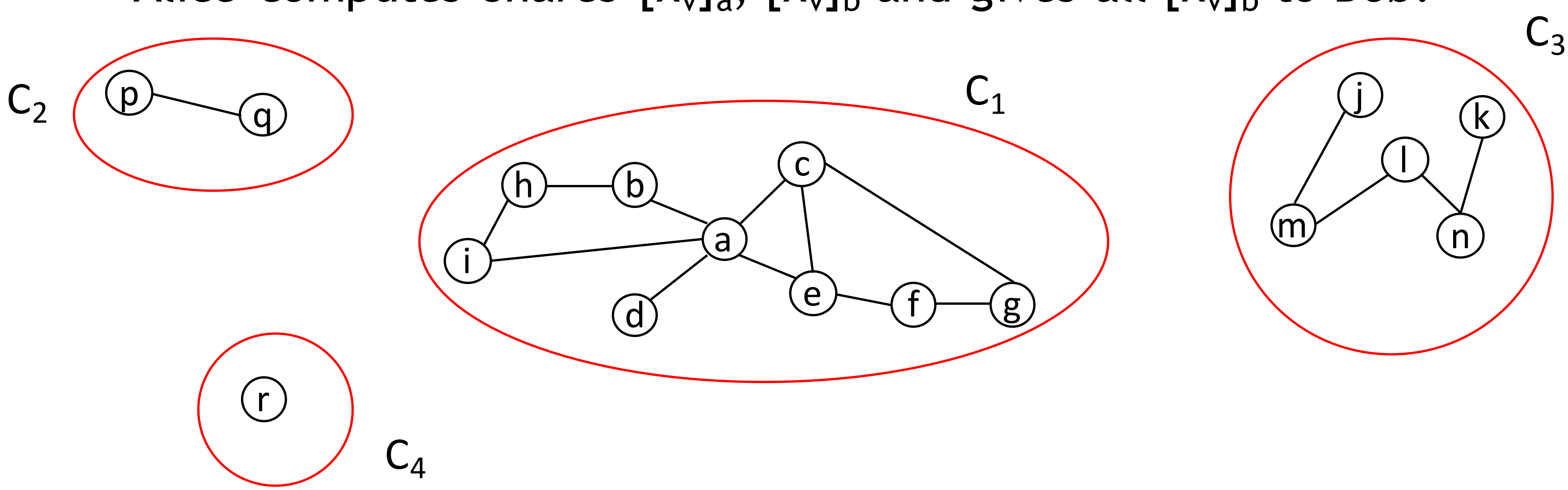
Algorithm Overview

- Alice (Bob) computes connected components on her (his) graph
- Secret share component names for each node (both Bob and Alice)
- Secret-shared shell expansion from s
- For each node compute secret-shared binary variable:
  - P(v) is 1 if node v in same component as s, else 0
- In end reveal P(t) by combining secret shares

- Can do this with hidden names except for s and t.

First Version: Shared Node Names

- Alice computes connected components
- x_v is component label for node v
  - $x_b=1$, $x_p=2$, $x_j=3$, $x_r = 4$
- Alice computes shares $[x_v]_a$, $[x_v]_b$ and gives all $[x_v]_b$ to Bob.



- Bob does the same. His node labels are y_v, shares $[y_v]_a$, $[y_v]_b$. He gives $[y_v]_a$ to Alice.

Computing in Secret-Shared World

- Each data center has its part of the secret label for every node in every center
- Addition, subtraction, multiplication by a constant are local
- Comparison, multiplication of shares requires communication to all

Alice has: $[x_1]_a$, $[x_2]_a$, … $[x_{na}]_a$, $[y_1]_a$, $[y_2]_a$, …. , $[y_{nb}]_a$, $[z_1]_a$, $[z_2]_a$, …, $[z_{nc}]_a$ for three parties (Alice, Bob, Carol)

Constraint on Component Labels

- Let P be a large prime, $P > n^2$ (n is # nodes). Field is integers mod P.

- Pick an $M > n$ such that $M^2 < P$. Require $1 < x_v < M$ for Alice. Bob's labels are tM for some $1 < t < M$.

Key: Alice's labels are different order(s) of magnitude from Bob's:

  – Alice's components: 1,2,3
  – Bob's components: 1000, 2000, 3000

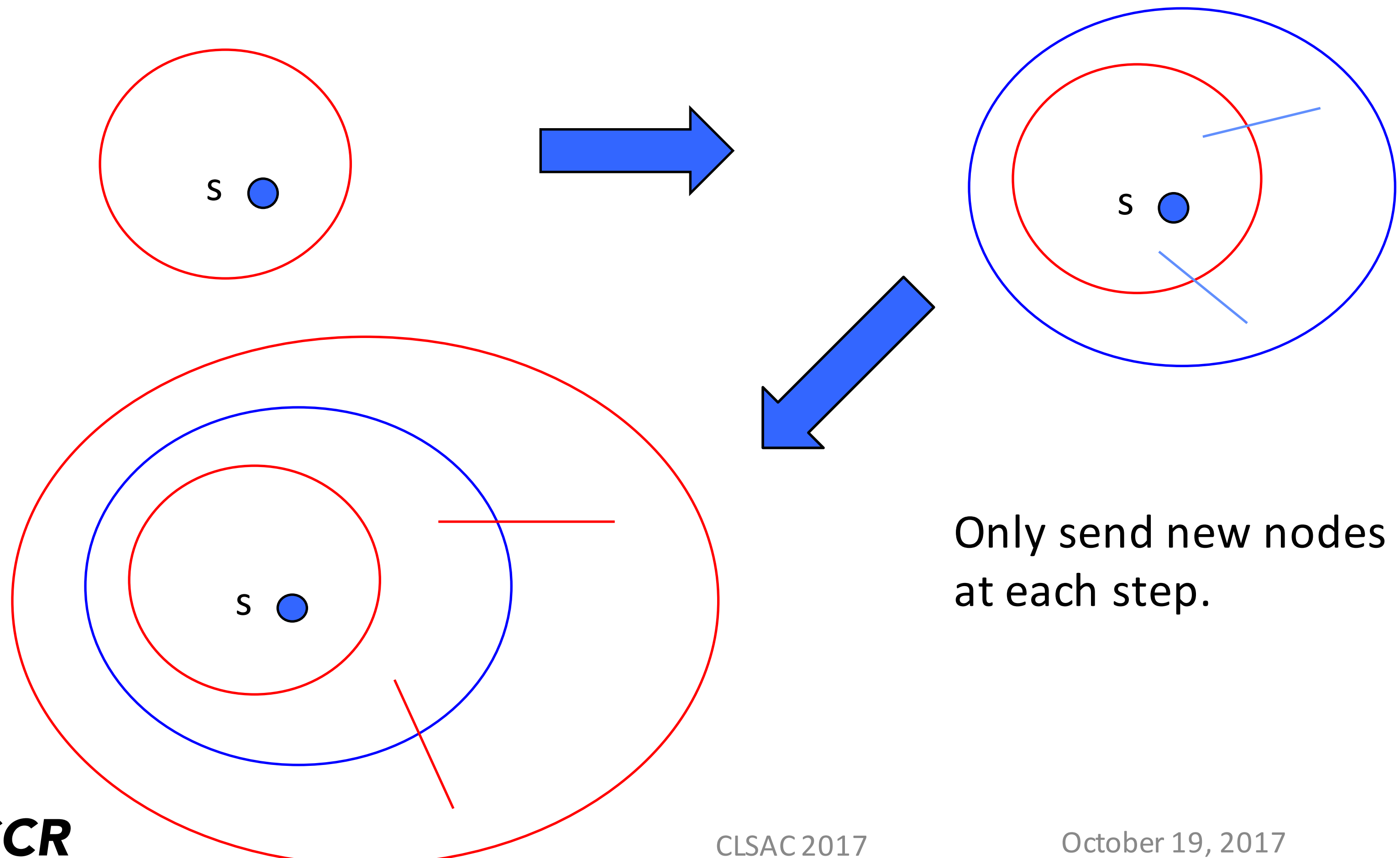
- No zero labels

Shell Expansion

- Like breadth-first-search, “layer” is connected piece in G_A or G_B
- Key: don’t explore too much of the graph(s)

Bob

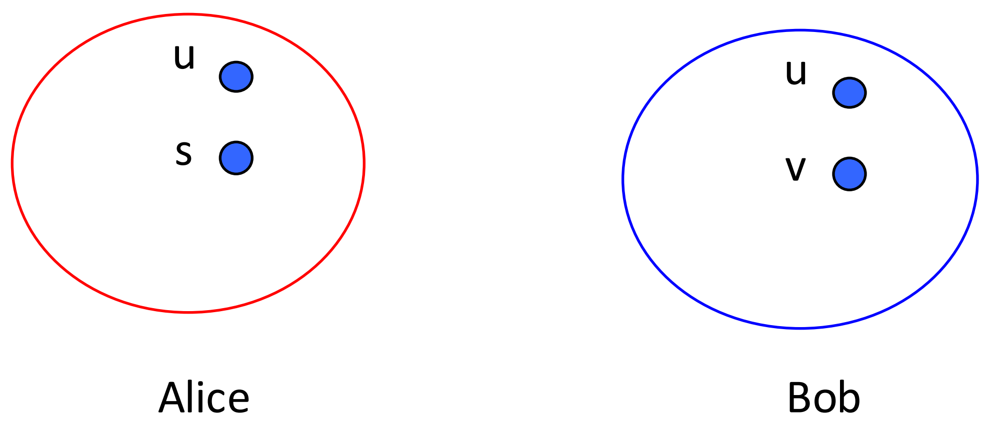
Alice



Only send new nodes at each step.

Propagating Connectivity Information

- P_v is a binary variable set to 0 iff there exists a node u such that $x_u = x_s$ and $y_u = y_v$.



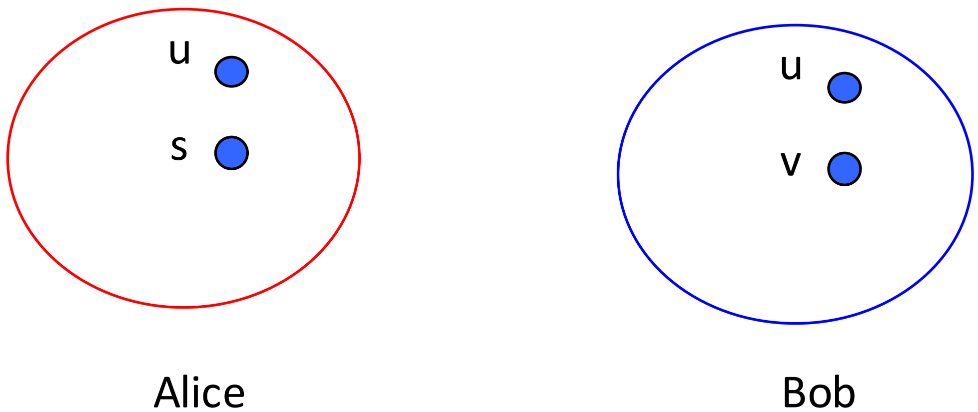
Algorithm 1 OddStep

```
P_v = 1
for node u do
    P_v ← MUX((x_s − x_u + y_u − y_v), P_v, 0)
end for
```

$$P_v = 1$$
for node u **do**
$$P_v \leftarrow \mathrm{MUX}((x_s - x_u + y_u - y_v), P_v, 0)$$
end for

Propagating Connectivity Information

- Pv is a binary variable set to 0 iff there exists a node u such that $x_u = x_s$ and $y_u = y_v$.

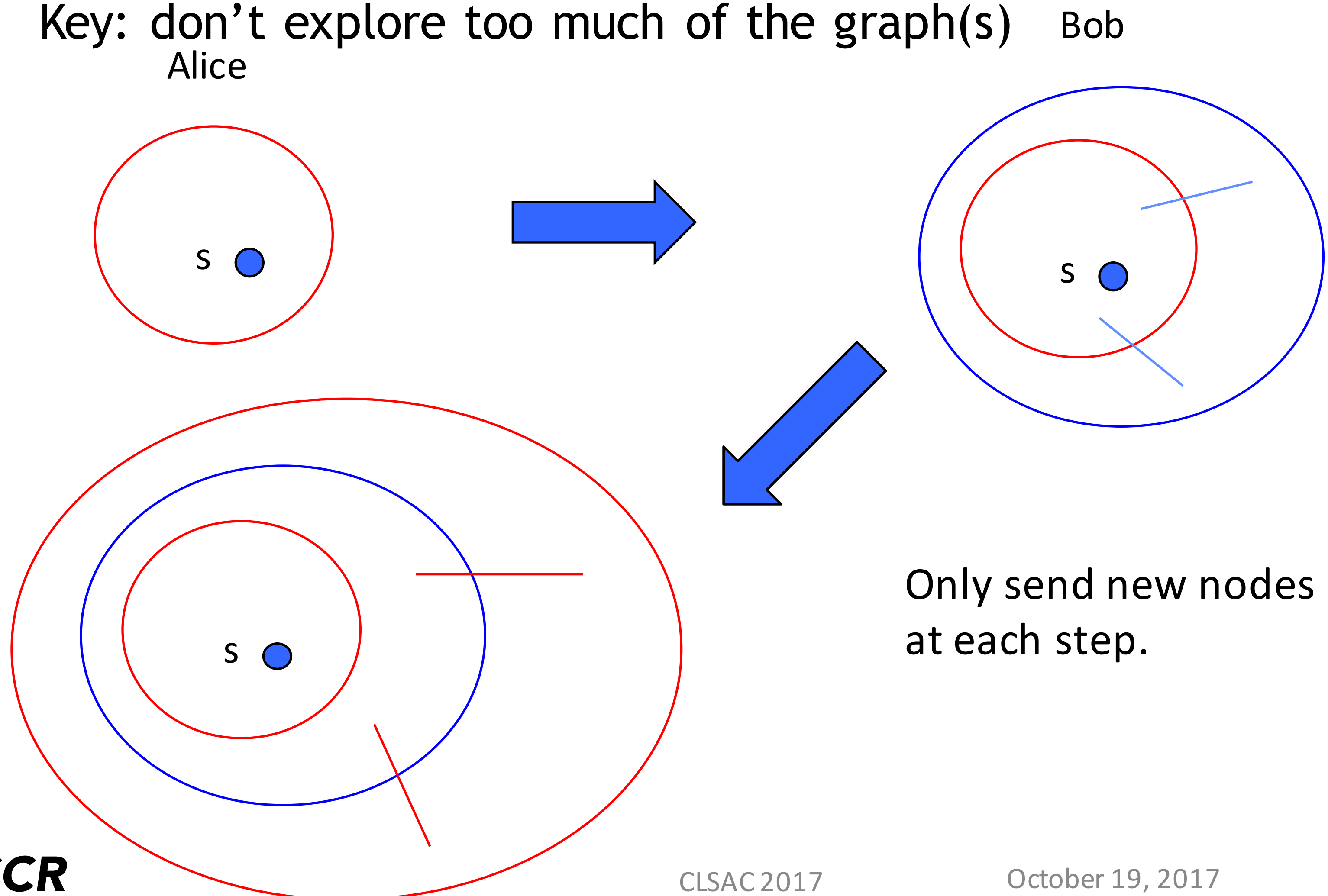


- Update the y_v, to show connectivity to s

$$y_v \leftarrow \mathrm{MUX}(P_v, y_v, y_s)$$

Shell Expansion

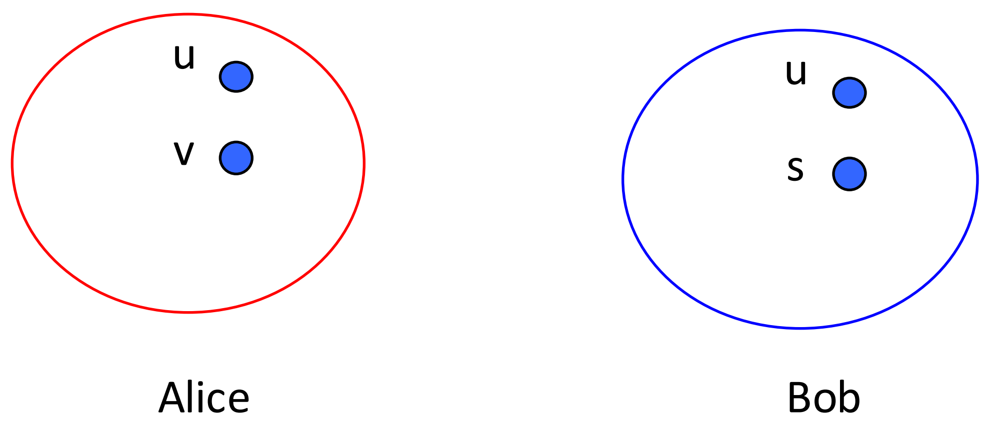
- Like breadth-first-search, "layer" is connected piece in G_A or G_B
- Key: don't explore too much of the graph(s)



Only send new nodes at each step.

Propagating Other Way Too

- Pv is a binary variable set to 0 iff there exists a node u such that $x_u = x_s$ and $y_u = y_v$.

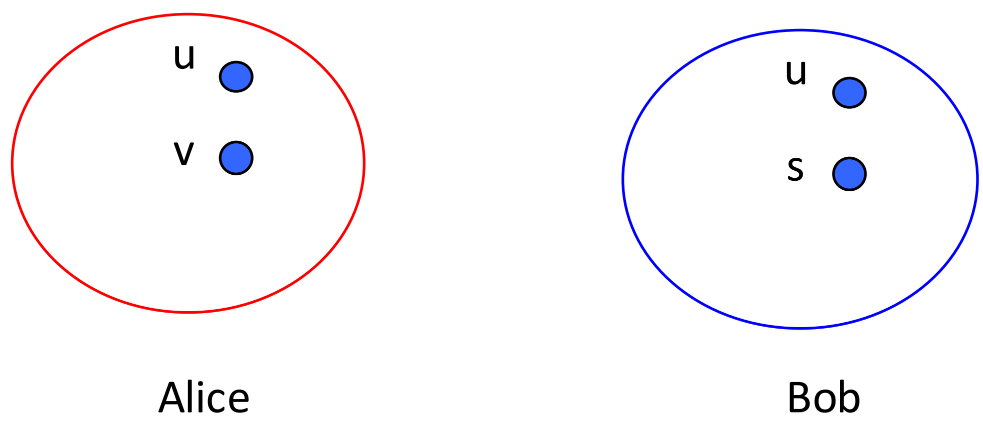


Algorithm 2 EvenStep

$P_v = 1$
for node u **do**
  $P_v \leftarrow \text{MUX}((y_s - y_u + x_u - x_v), P_v, 0)$
end for

Propagating Connectivity Information

- Pv is a binary variable set to 0 iff there exists a node u such that $x_u=x_s$ and $y_u=y_v$.

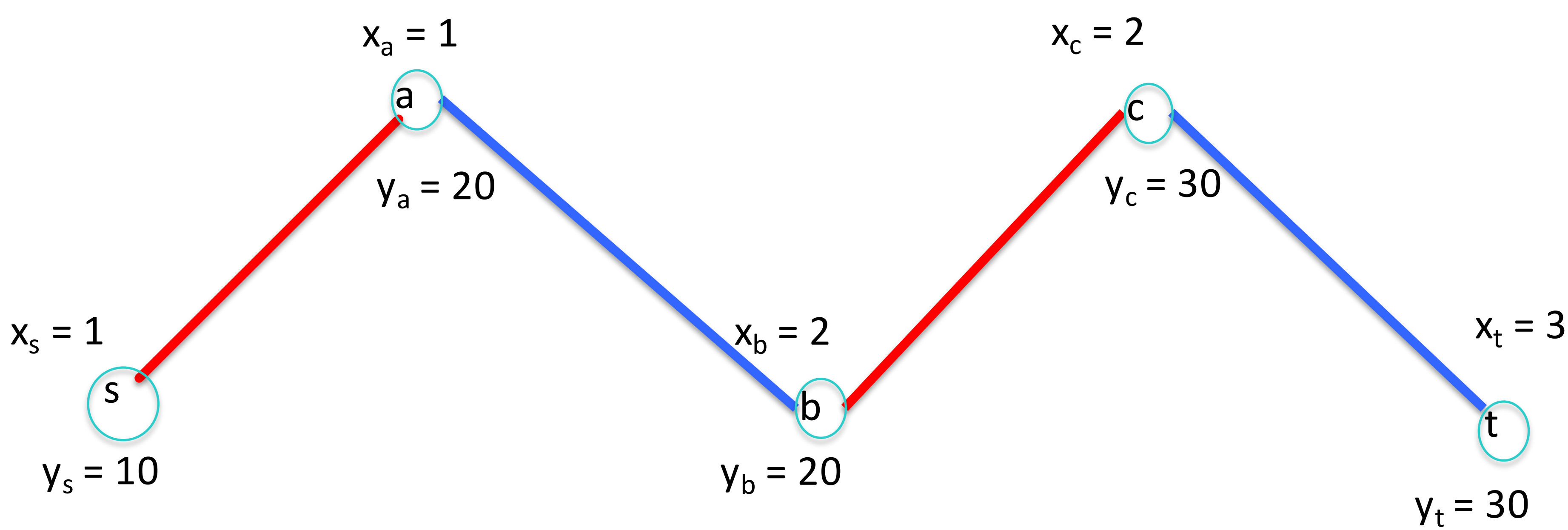


- Update the x_v, to show connectivity to s

$$x_v \leftarrow MUX(P_v,\ x_v,\ x_s)$$

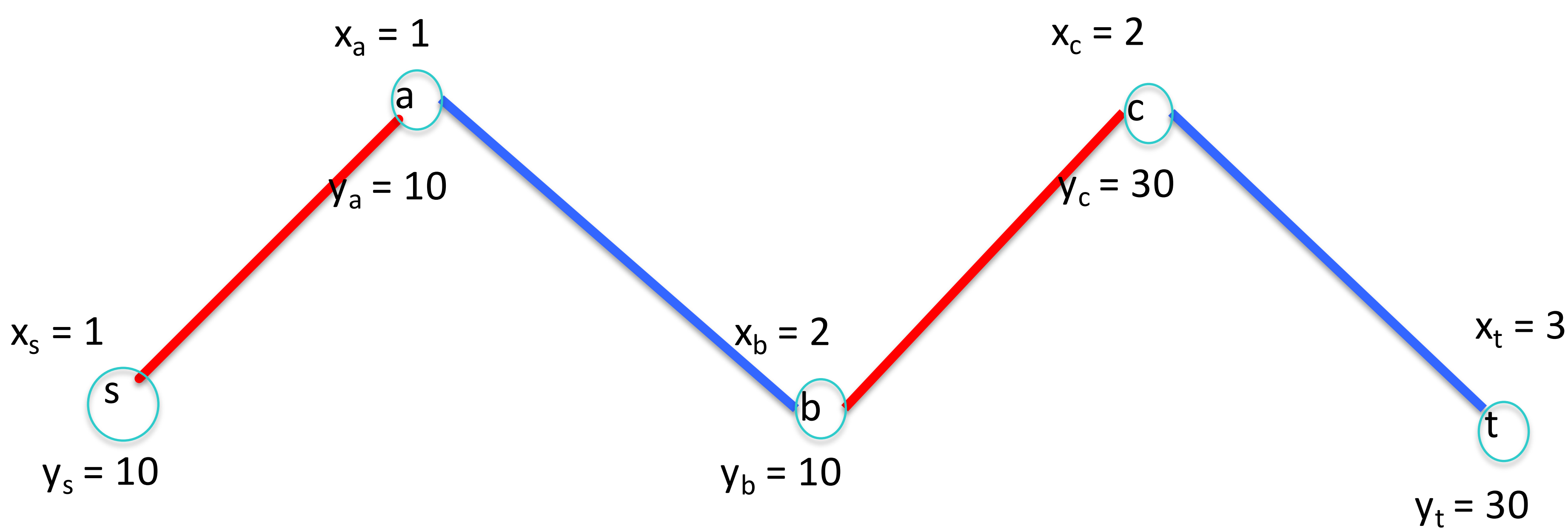
Example

- Here are the labels at the start:



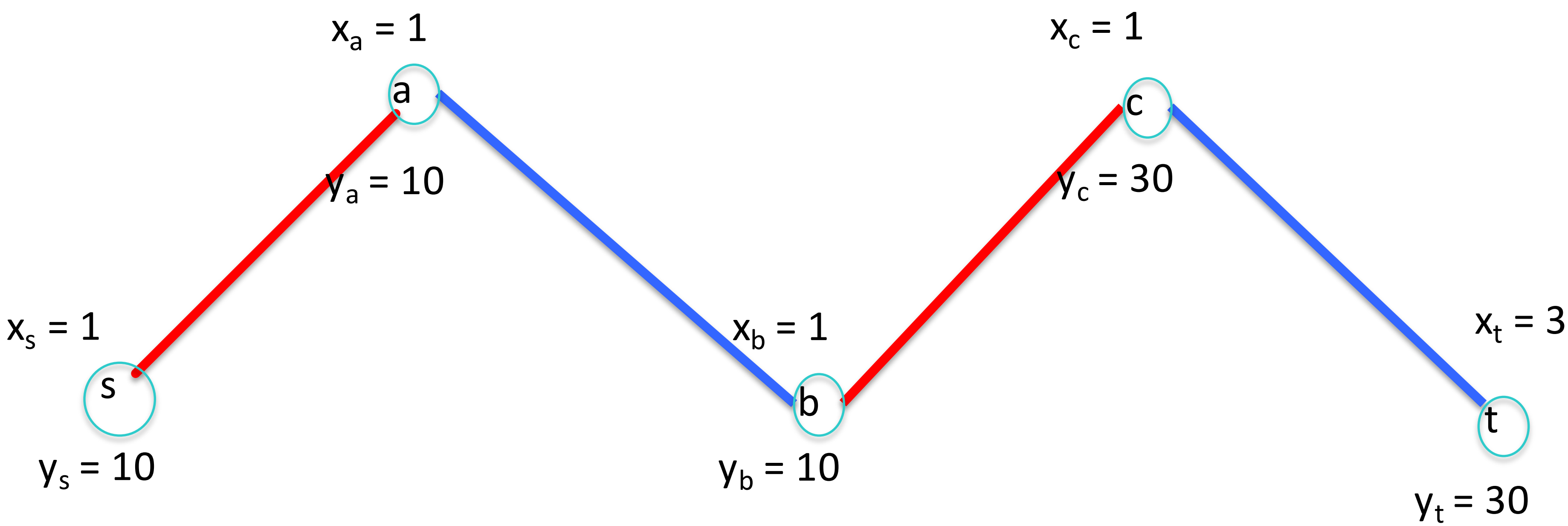
- $P_a = 0$ because $x_s - x_a + y_a - y_a = 0$ (u = a)
- $P_b = 0$ because $x_s - x_a + y_a - y_b = 0$ (u = a)
- So y_a and y_b are set to y_s

Example



- $P_b = 0$ because $y_s - y_b + x_b - x_b = 0$ (u = b)
- $P_c = 0$ because $y_s - y_b + x_b - x_c = 0$ (u = b)
- So x_b and x_c are set to x_s

Example



- The next step sets $y_t = 10 = y_s$
- From that point on $P_t = 0$
- After enough steps, compare shares to decode P_t.
- Enough steps: diameter (at most n-1), or j if only care about paths of length at most j

Hiding Names

- Arrays of names and labels
  - Arbitrary, except s, t are first

Dummy node (→ β, δ)

Alice

s	t	c	b	q	a	e	α	β	δ	Names
x_s	x_t	x_c	x_b	x_q	x_a	x_e	x_α	x_β	x_δ	Labels

Bob

s	t	a	q	g	e	h	b	ζ	μ	Names
y_s	y_t	y_a	y_q	y_g	y_e	y_h	y_b	y_ζ	y_μ	Labels

Secret-Shared Permutation

- Secret-shared y' array effectively permutes Bob's labels to match

Alice

s	t	c	b	q	a	e	α	β	δ	Names
y_s	y_t	0	y_b	y_q	y_a	y_e	0	0	0	Bob's Permuted Labels y'

Bob

s	t	a	q	g	e	h	b	ζ	μ	Names
y_s	y_t	y_a	y_q	y_g	y_e	y_h	y_b	y_ζ	y_μ	Labels

Secret Names

- Compute using MUX (just comparisons of unknown objects)
- Then use y' instead of y in previous algorithm

for j **do**
  $y'_j \leftarrow 0$
  for i **do**
    $y'_j \leftarrow y'_j + \text{MUX}(\hat{x}_j - \hat{y}_i,\ 0,\ y_i)$
  end for
end for

Secret-shared names

Then the parties compute shares of P_k as

$P_k \leftarrow 1$
for j **do**
  $P_k \leftarrow \text{MUX}(x_s - x_j + y'_j - y_k,\ P_k,\ 0)$
end for

Complexity: One Shell Expansion

- Setting the P_v indicator variables requires n^2 MUX computations
  - Must do for all values of u and v
  - n is an upper bound on the nodes for Alice, Bob

Algorithm 1 OddStep

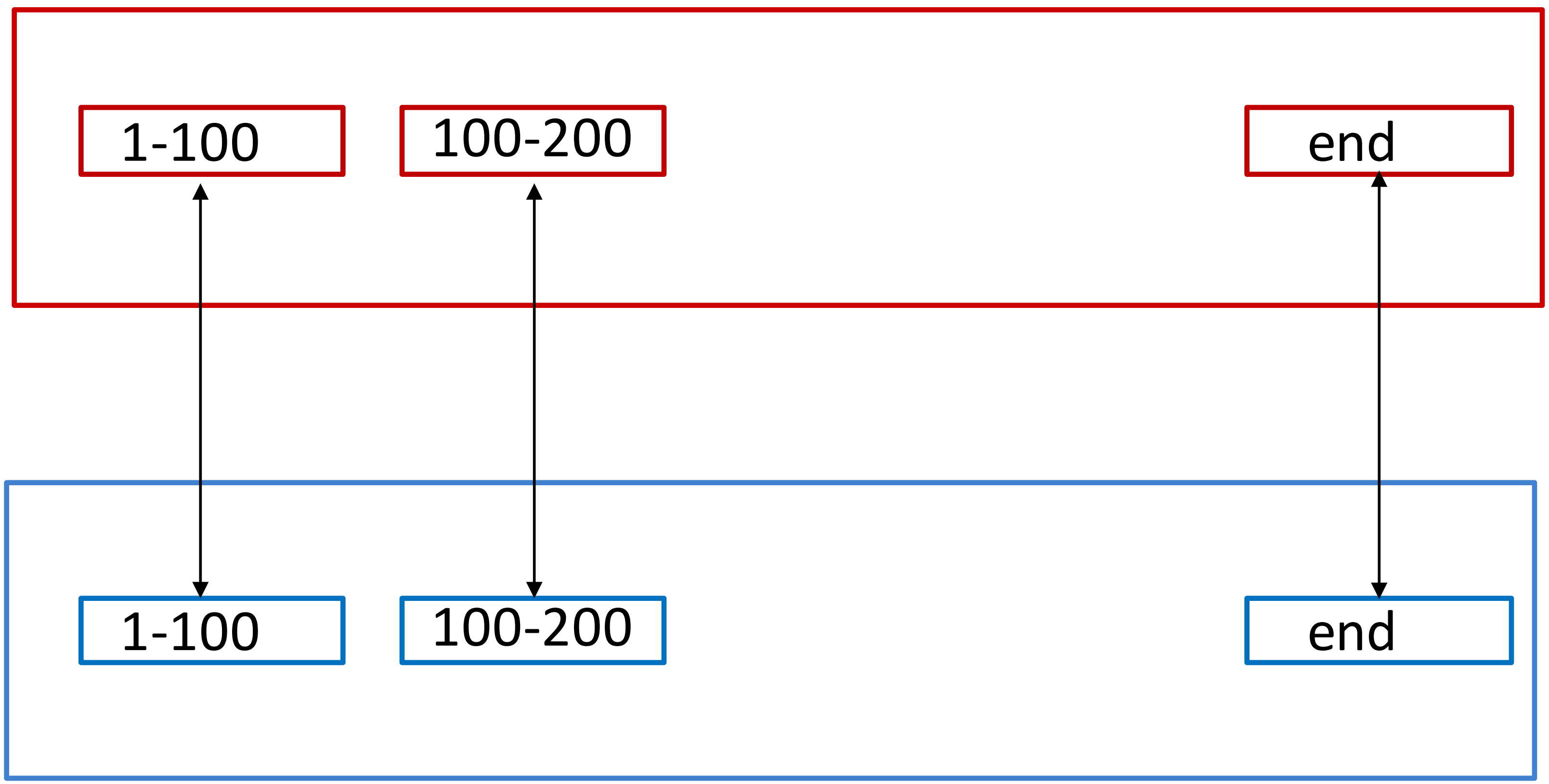
```
1: P_v = 1
2: for node u do
3:     P_v ← MUX((x_s − x_u + y_u − y_v), P_v, 0)
4: end for
```

- Updating the labels requires n MUX computations:

$$y_v \leftarrow \text{MUX}(P_v, y_v, y_s)$$

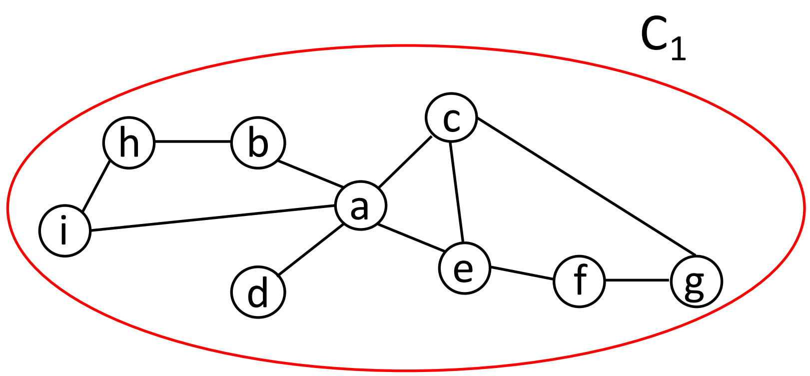
Computation of P_v in Parallel

- Every P_v can be computed independently (n MUXs each)
- Get label shares locally at each center (after updates)



Complexity: Shell Expansion Rounds

- Number of rounds of shell expansion
  - Worst case is n
  - If a shortest (hop-based) path of more than d is not interesting
    - Stop at d rounds
    - Could still report “yes” for longer shortest paths
  - Can check for t connected to s after each shell expansion



Communication Complexity of Operations

$$\mathrm{MUX}(c, a, b) = \begin{cases} a, & c \neq 0, \\ b, & \text{otherwise.} \end{cases}$$

- Comparison of a shared value to 0 [Nishide and Ohta]
  - Deterministic in 8 rounds, 81ℓ total communication, where ℓ is the number of bits in the prime used to create the secret field
  - Randomized in 4 rounds, 12k total communication, error probability $\frac{1}{2^k}$
  - Comparison bit d = 0 if c=0 and 1 otherwise

Communication Complexity of Operations

$$\mathrm{MUX}(c,a,b) = \begin{cases} a, & c \neq 0, \\ b, & \text{otherwise.} \end{cases}$$

- Comparison bit d = 0 if c=0 and 1 otherwise
- Output = da + (1-d)b
- Multiplication: send one share to all, receive one share from all
  - Total communication per machine: 2(# centers – 1)ℓ
    - Ben-Or, Goldwasser, Wigderson

Write Up

- Initial idea:

- J. Berry, M. Collins, Aaron Kearns, C. Phillips, J. Saia, R. Smith, "Cooperative computing for autonomous data centers," Proceedings of the IEEE International Parallel and Distributed Processing Symposium, May 2015.