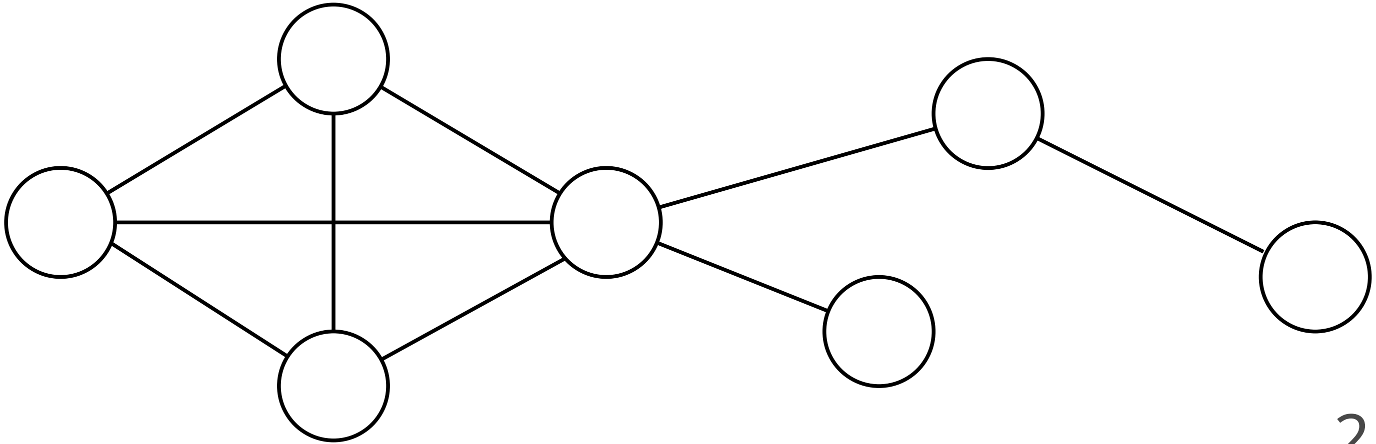
Space-stretch tradeoff in routing revisited

Tolik Zinovyev (Boston University)

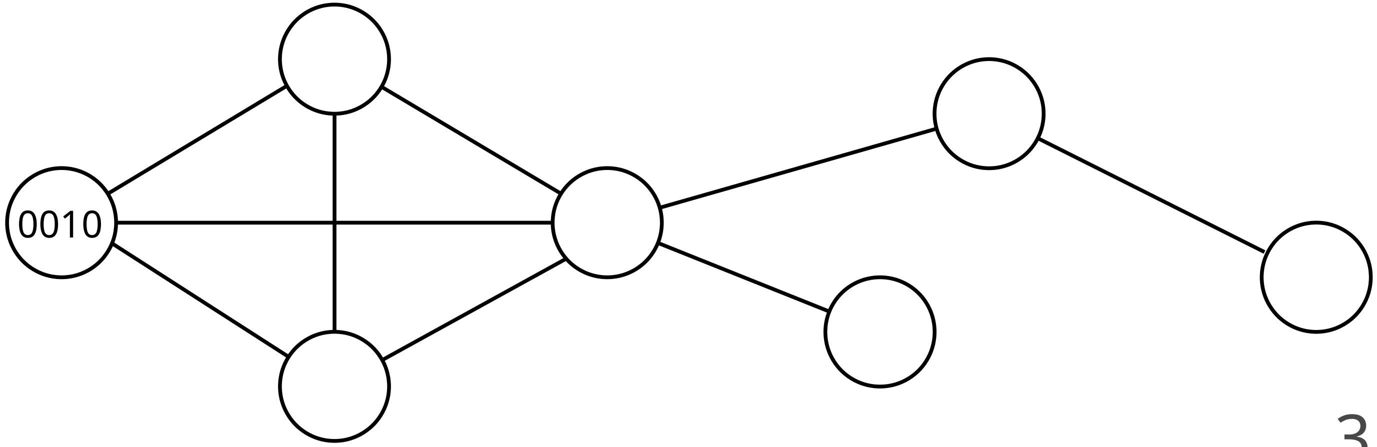
Routing in computer networks, model

- Undirected graph (network)



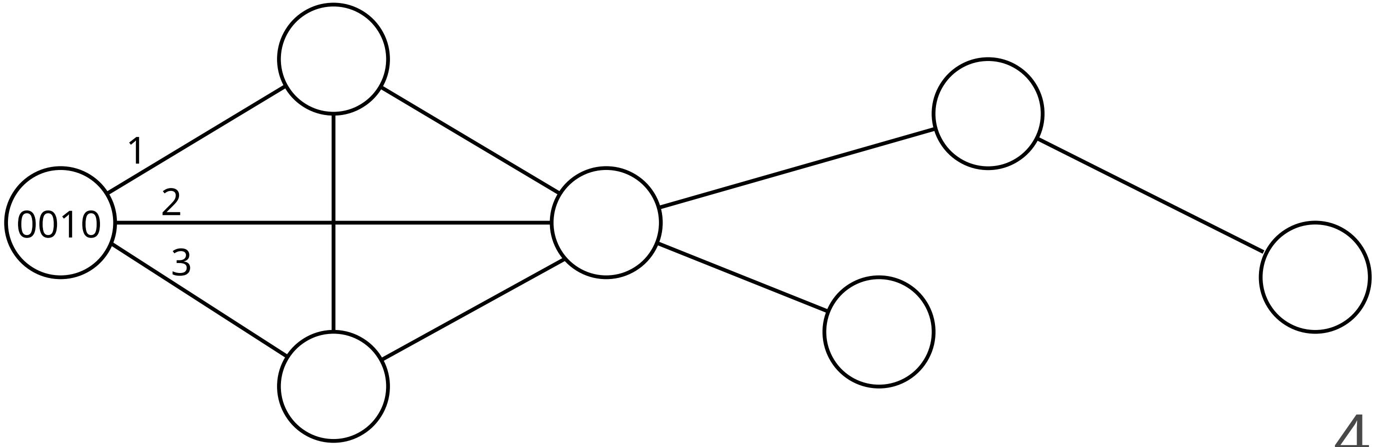
Routing in computer networks, model

- Undirected graph (network)
- Nodes have labels (a binary string)



Routing in computer networks, model

- Undirected graph (network)
- Nodes have labels (a binary string)
- Nodes have ports

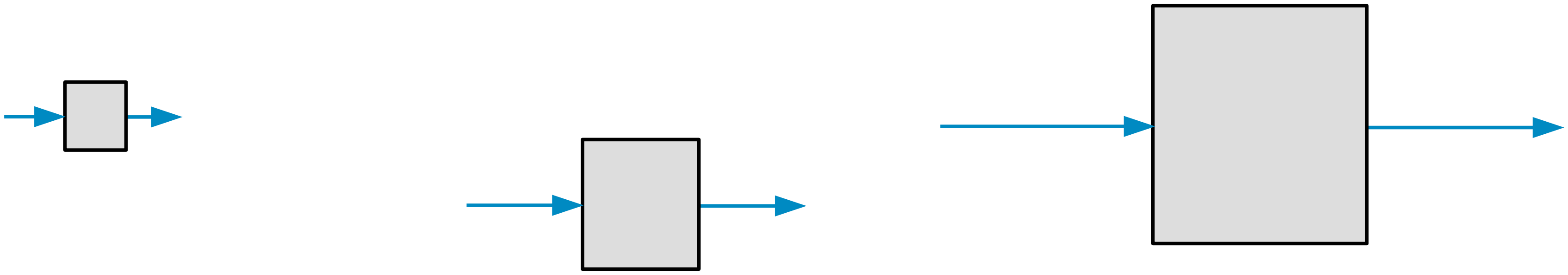


Routing in computer networks, model

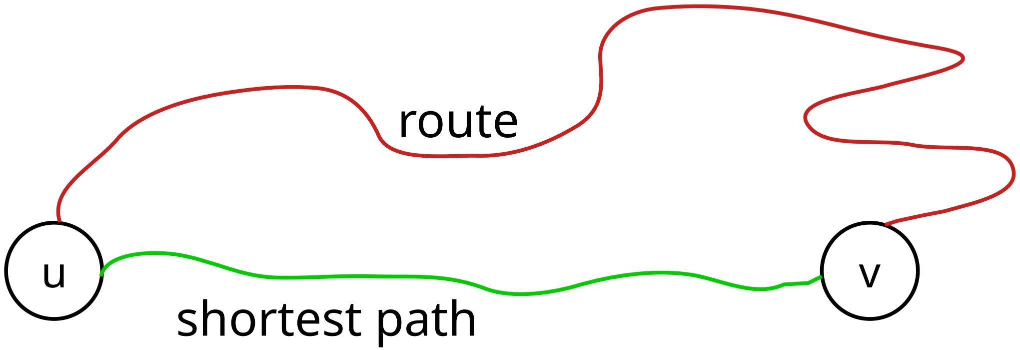
- Undirected graph (network)
- Nodes have labels (a binary string)
- Nodes have ports
- Nodes have routing programs

Routing characteristics

Space usage: program size

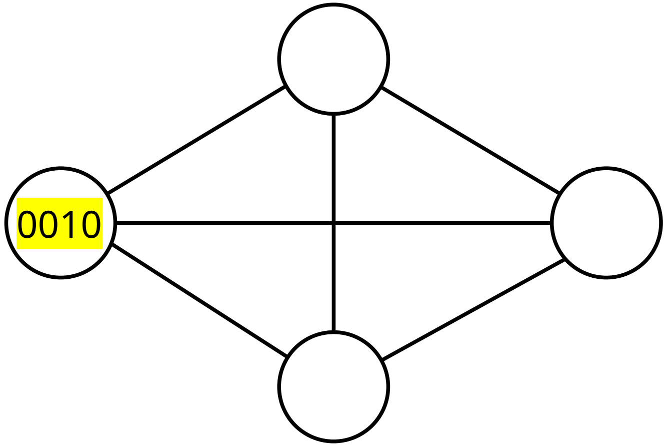


Routing stretch: route length / distance



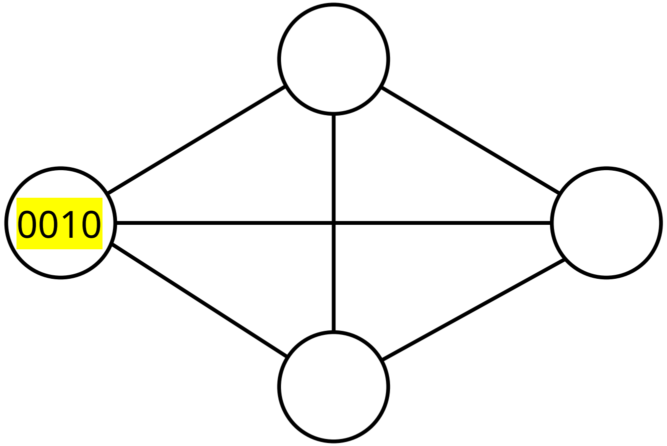
Model, continued

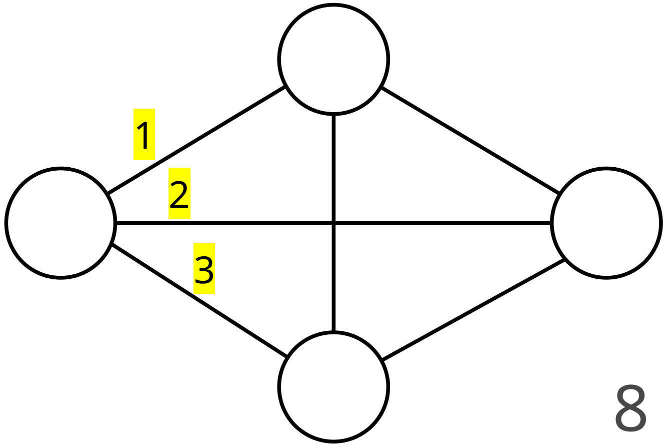
- Adversarial / <u>non-adversarial</u> labels
  - Adversarial: labels given; aka name-independent model
  - Non-adversarial: designer labels; aka labeled model



Model, continued

- Adversarial / non-adversarial labels
  - Adversarial: labels given; aka name-independent model
  - Non-adversarial: designer labels; aka labeled model
- Adversarial / non-adversarial ports
  - Adversarial: ports given
  - Non-adversarial: designer ports





Lower bounds with non-adversarial ports

Work	Stretch	Local memory (bits)	Notes
Gavoille and Perennes (1996)	< 5/3	$\Omega(n \log n)$ on $\Omega(n)$ nodes	Node labels are [n]
Buhrman, Hoepman, Vitányi (1996)	1	$\Omega(n)$ on $\Omega(n)$ nodes	
Gavoille and Gengler (2001)	< 3	$\Omega(n)$ on some node	complex proof

Lower bounds with non-adversarial ports

Work	Stretch	Local memory (bits)	Notes
Gavoille and Perennes (1996)	< 5/3	$\Omega(n \log n)$ on $\Omega(n)$ nodes	Node labels are [n]
Buhrman, Hoepman, Vitányi (1996)	1	$\Omega(n)$ on $\Omega(n)$ nodes	
Gavoille and Gengler (2001)	< 3	$\Omega(n)$ on some node	complex proof
This paper	< 3	$\Omega(n)$ on cn nodes, $\forall 0<c<1$	

Lower bounds with adversarial ports

Work	Stretch	Local memory (bits)	Notes
Peleg and Upfal (1989)	3 5 $s \geq 1$	$\Omega(n^{1/5})$ on some node $\Omega(n^{1/7})$ on some node $\Omega(n^{1/(s+2)})$ on some node	
Thorup and Zwick (2001)	3 5 $< 2k+1$	$\Omega(n^{1/2})$ on some node $\Omega(n^{1/3})$ on some node $\Omega(n^{1/k})$ on some node	Does not work in standard model; relies on girth conjecture

Lower bounds with adversarial ports

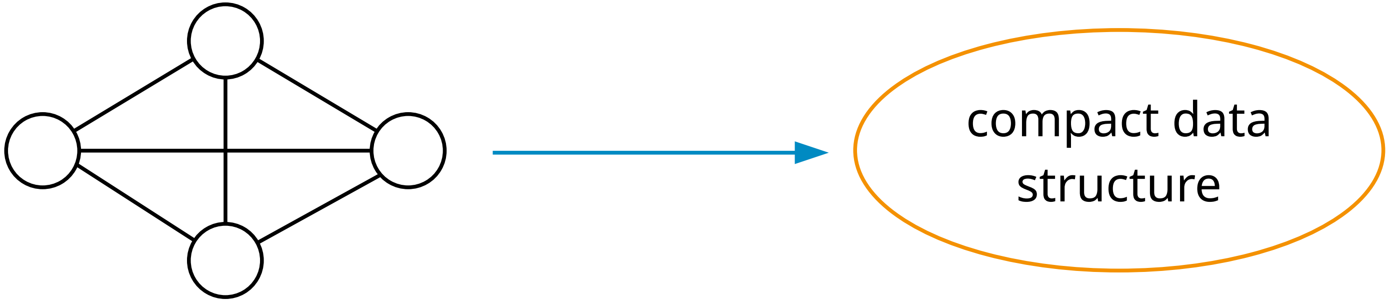
Work	Stretch	Local memory (bits)	Notes
Peleg and Upfal (1989)	3 5 $s \geq 1$	$\Omega(n^{1/5})$ on some node $\Omega(n^{1/7})$ on some node $\Omega(n^{1/(s+2)})$ on some node	
Thorup and Zwick (2001)	3 5 $< 2k+1$	$\Omega(n^{1/2})$ on some node $\Omega(n^{1/3})$ on some node $\Omega(n^{1/k})$ on some node	Does not work in standard model; relies on girth conjecture

Lower bounds with adversarial ports

Work	Stretch	Local memory (bits)	Notes
Peleg and Upfal (1989)	3 5 $s \geq 1$	$\Omega(n^{1/5})$ on some node $\Omega(n^{1/7})$ on some node $\Omega(n^{1/(s+2)})$ on some node	
Thorup and Zwick (2001)	3 5 $< 2k+1$	$\Omega(n^{1/2})$ on some node $\Omega(n^{1/3})$ on some node $\Omega(n^{1/k})$ on some node	Does not work in standard model; relies on girth conjecture
This paper	$< 2k+1$	$\Omega(n^{1/k} \log n)$ on some node	relies on girth conjecture

Previous proof does not work

- Mikkel Thorup's and Uri Zwick's proof relies on a reduction from approximate distance oracles

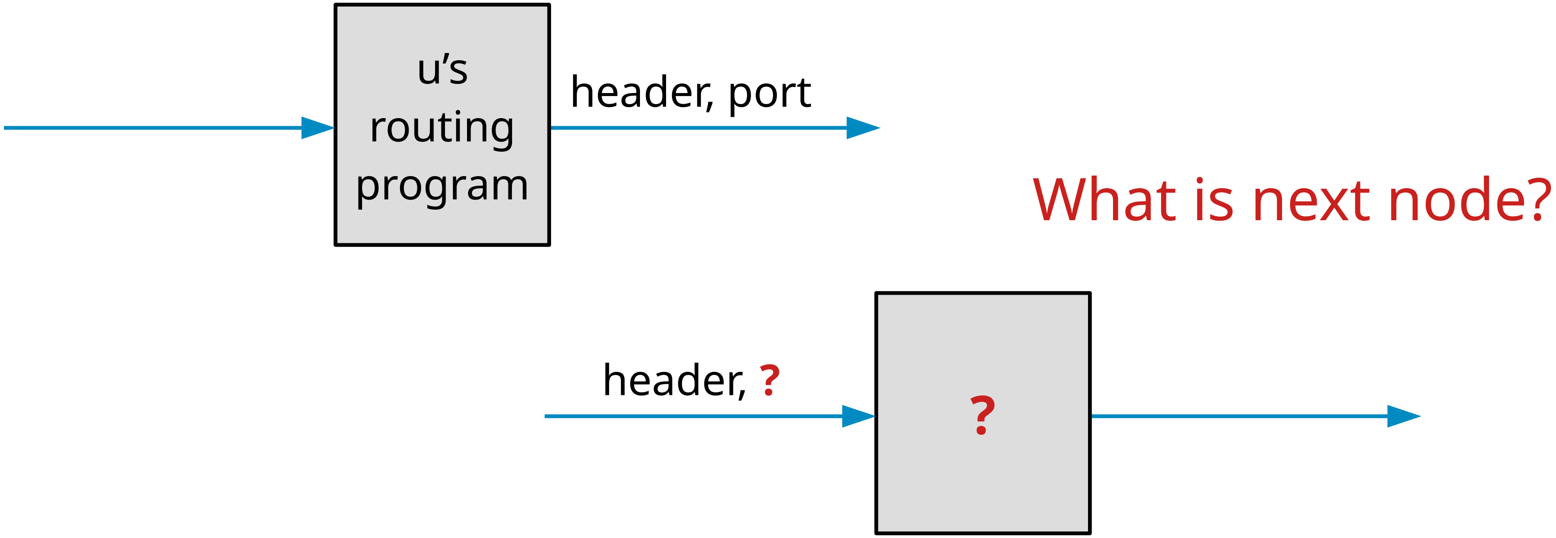


- Distance oracles with stretch <2k+1 require $\Omega(n^{1+1/k})$ bits of storage [1]

[1]: Mikkel Thorup and Uri Zwick. Approximate distance oracles. 2001

Previous proof does not work

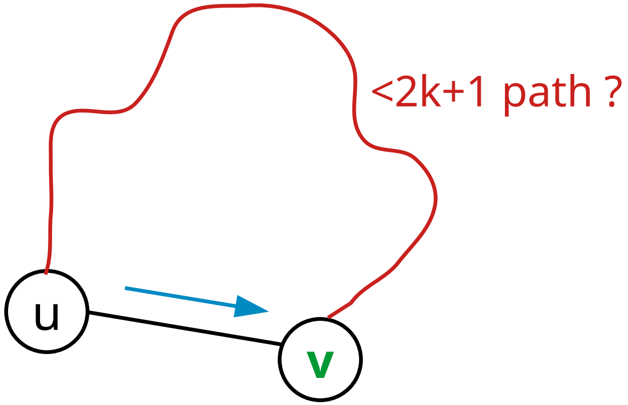
- Reduction in [2]:
  - Given a routing scheme with small size, construct small distance oracle
  - Distance oracle simulates routing and counts hops



[2]: Mikkel Thorup and Uri Zwick. Compact routing schemes. 2001

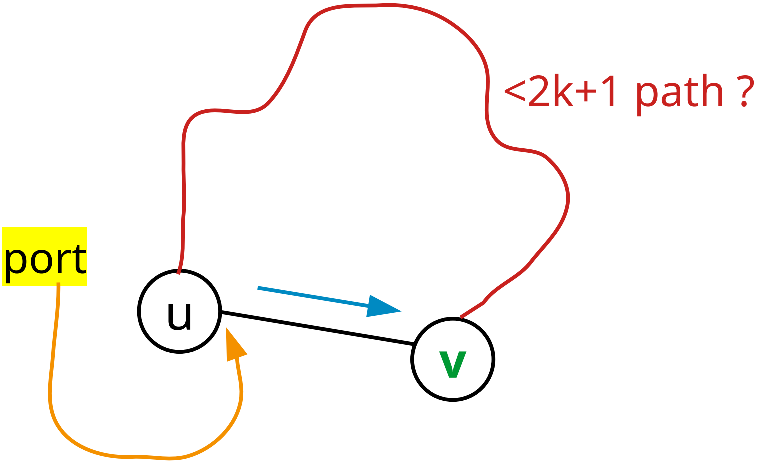
New proof

- Works in the standard model
- Borrows inspiration from Thorup's and Zwick's proof by using graphs with large girth
- Property: routing to a neighbor with stretch <2k+1 in graph with girth 2k+2 traverses edge toward neighbor



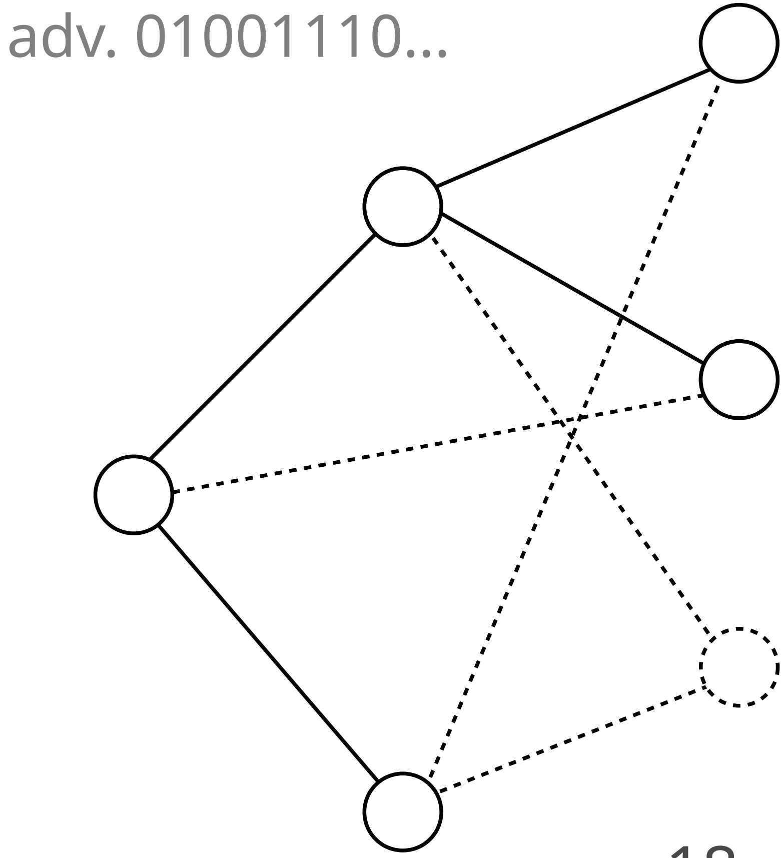
New proof

- Routing scheme must know port toward neighbor!
- Make extractor program
  - input: routing scheme, node labels, advice string
  - output: port assignment for the whole graph



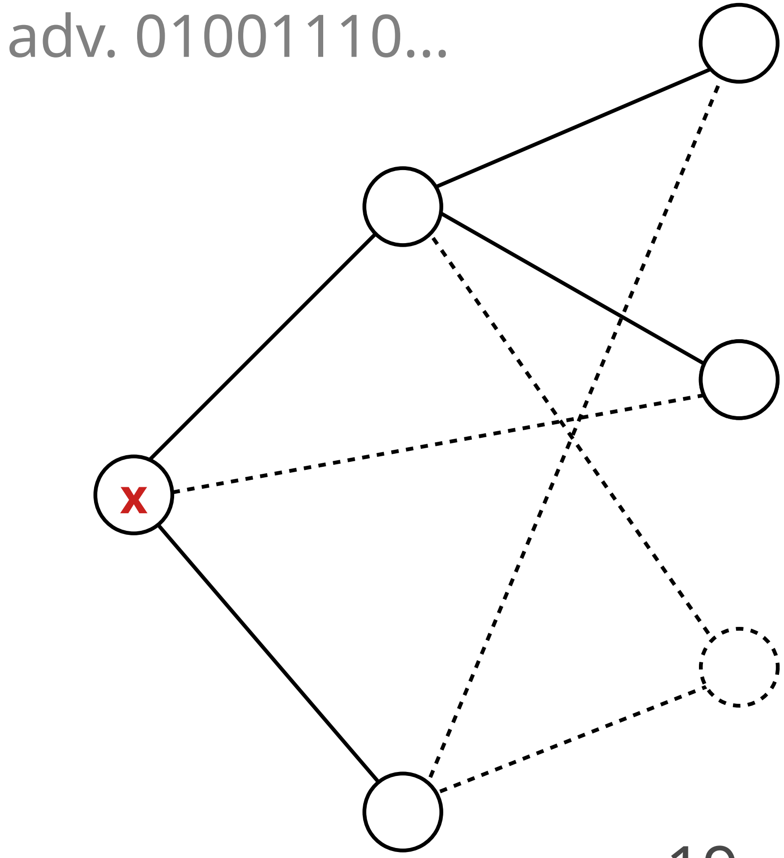
Extractor program

- Start with unknown port assignment



Extractor program

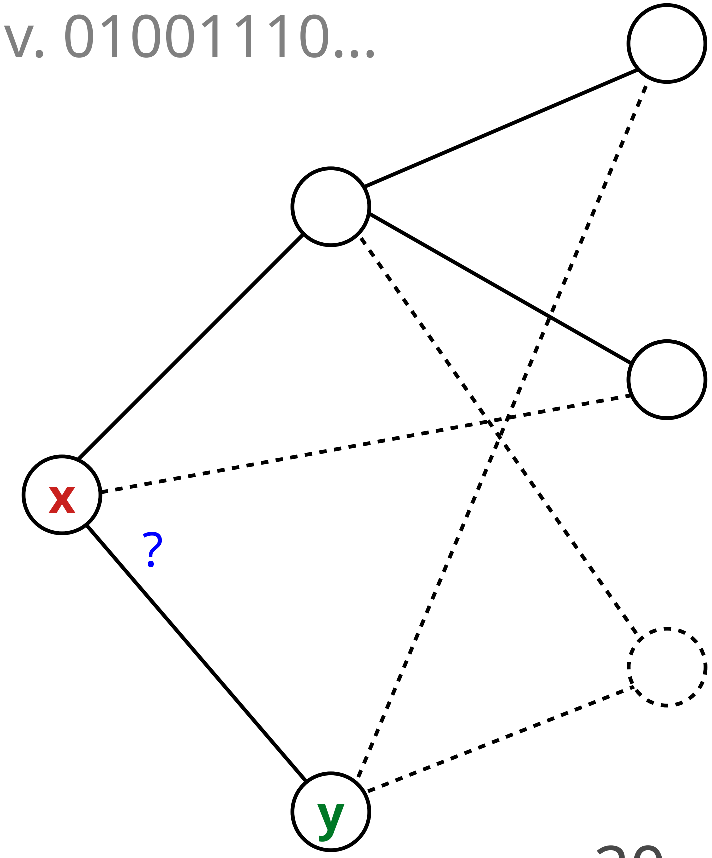
- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment



Extractor program

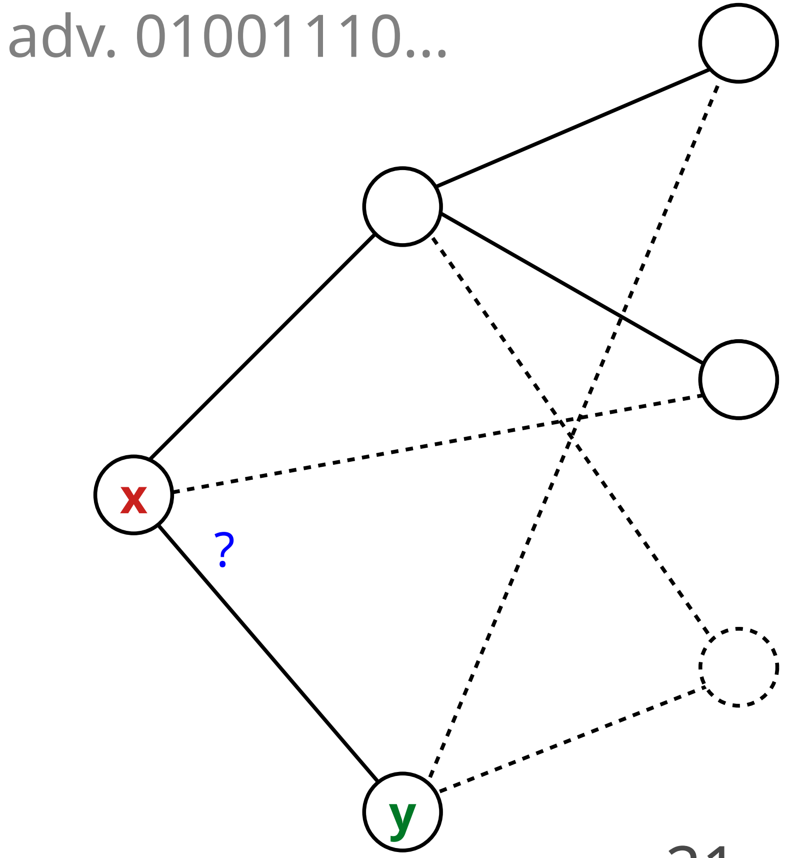
- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port

adv. 01001110...



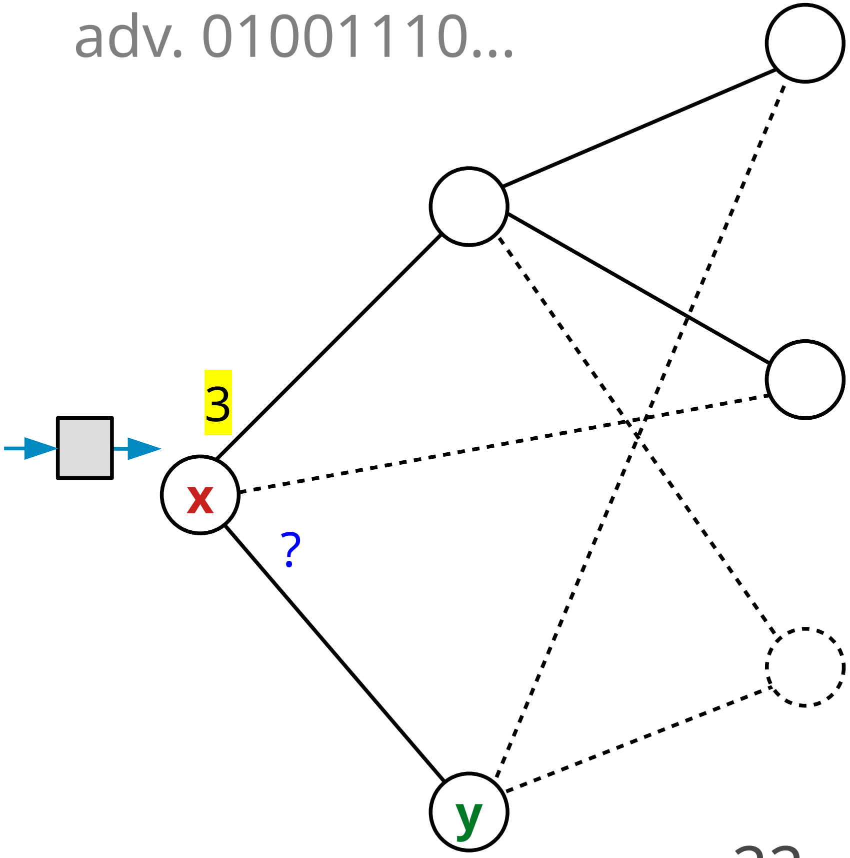
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



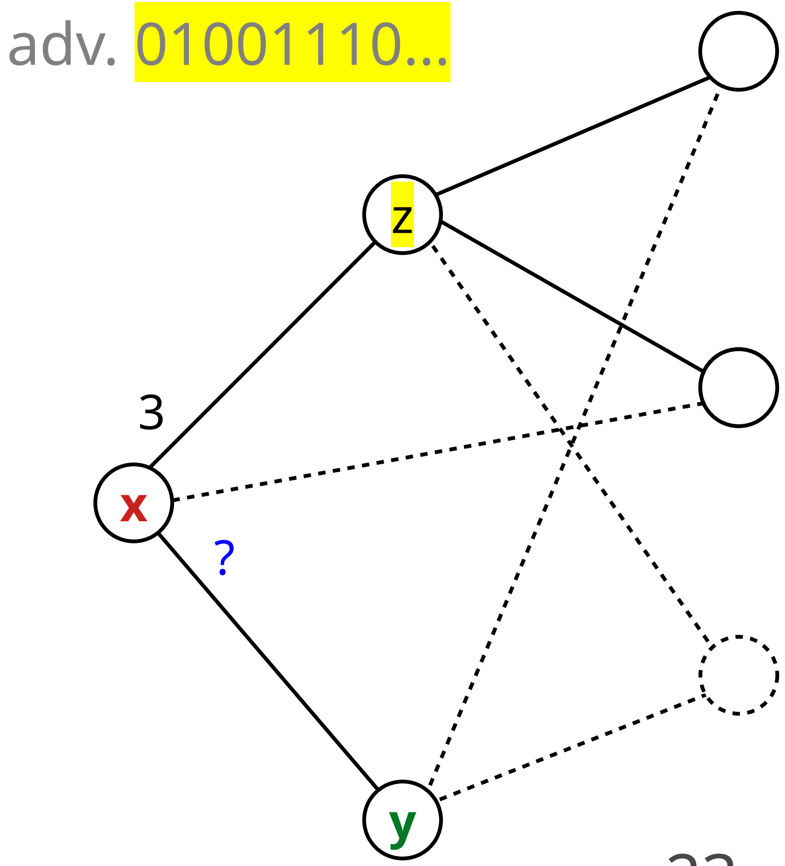
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



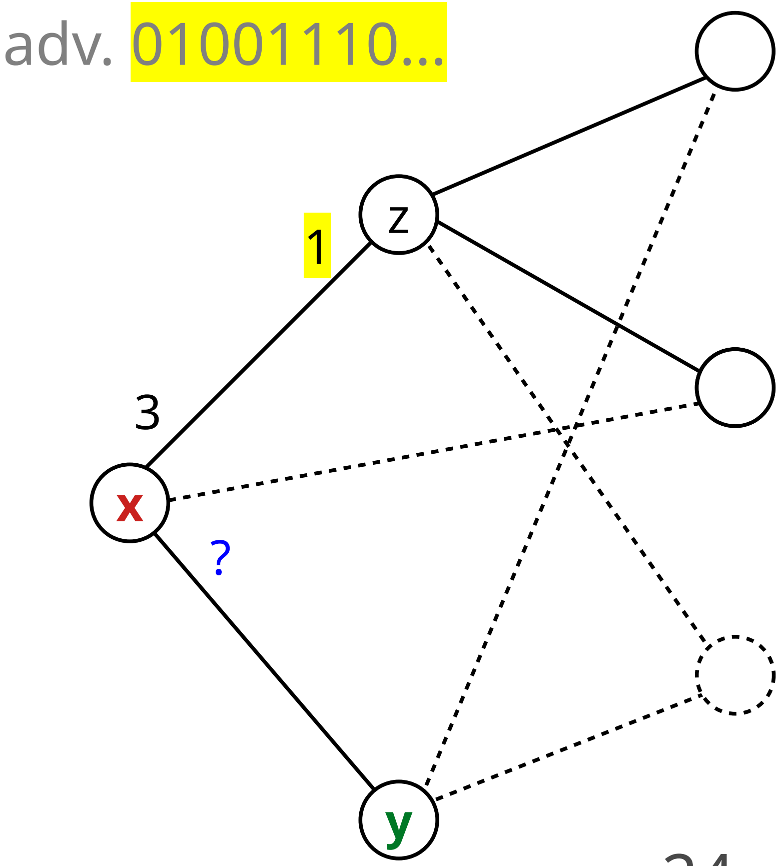
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



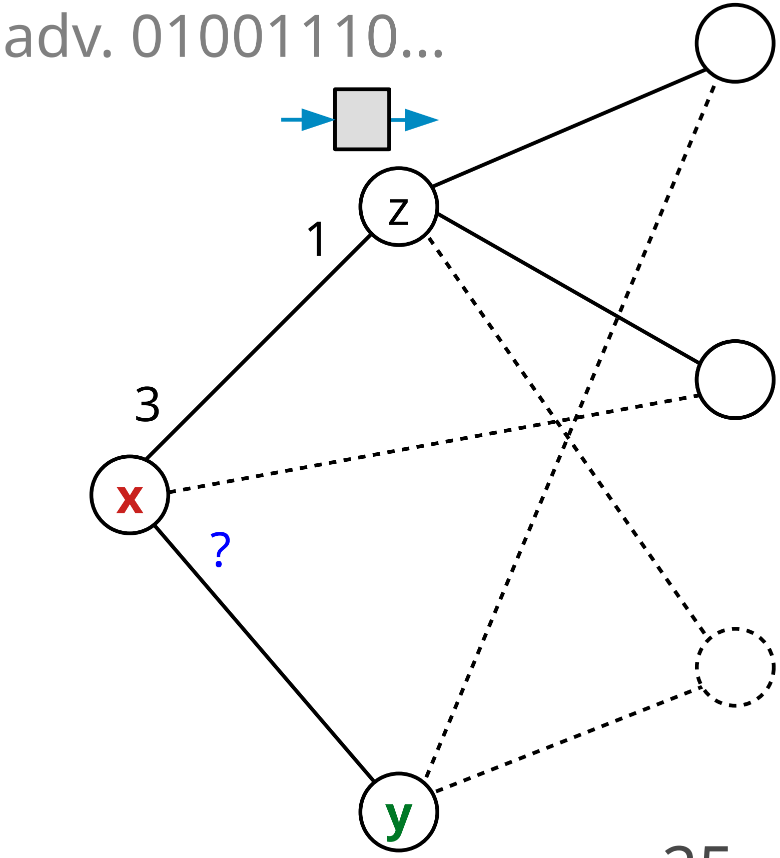
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



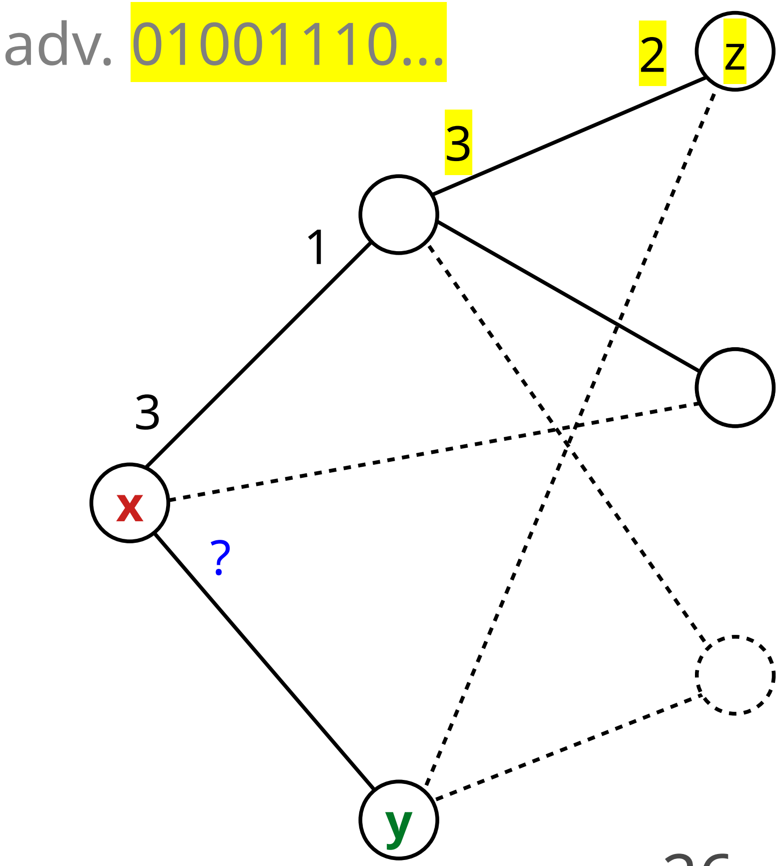
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



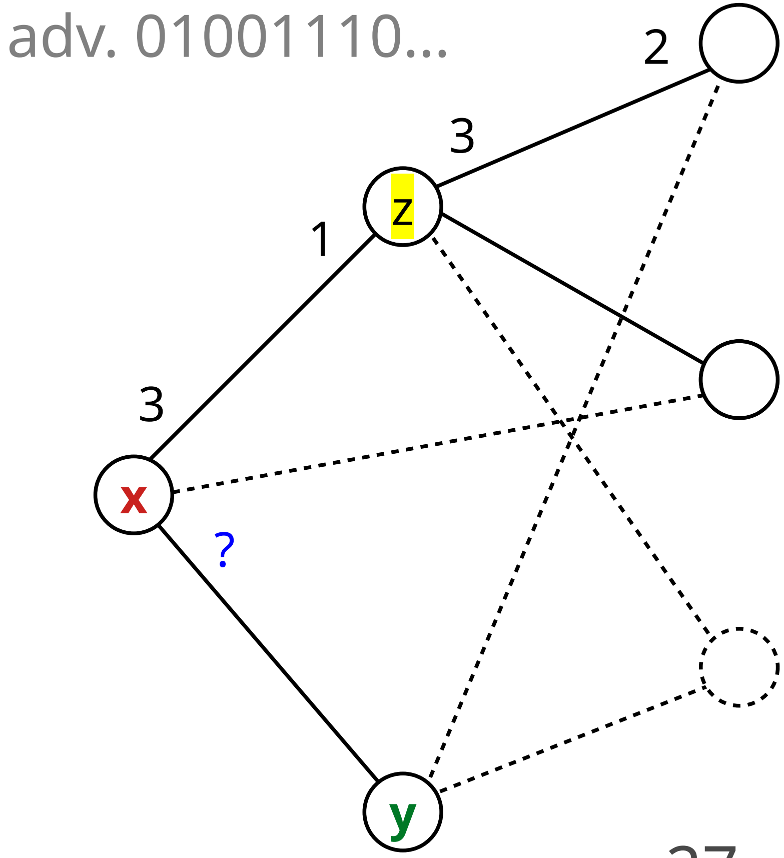
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



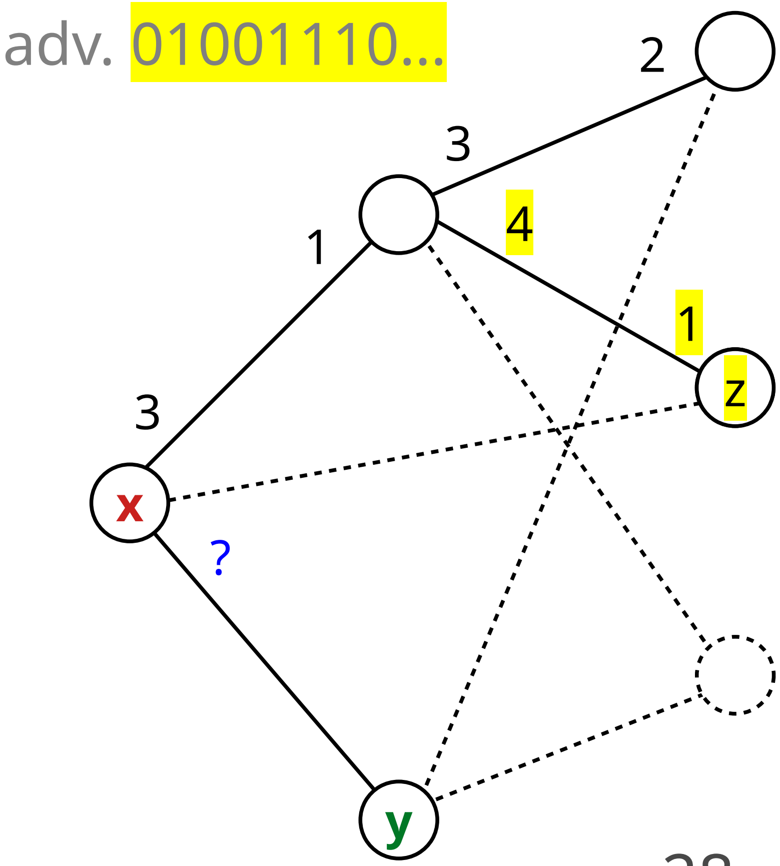
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



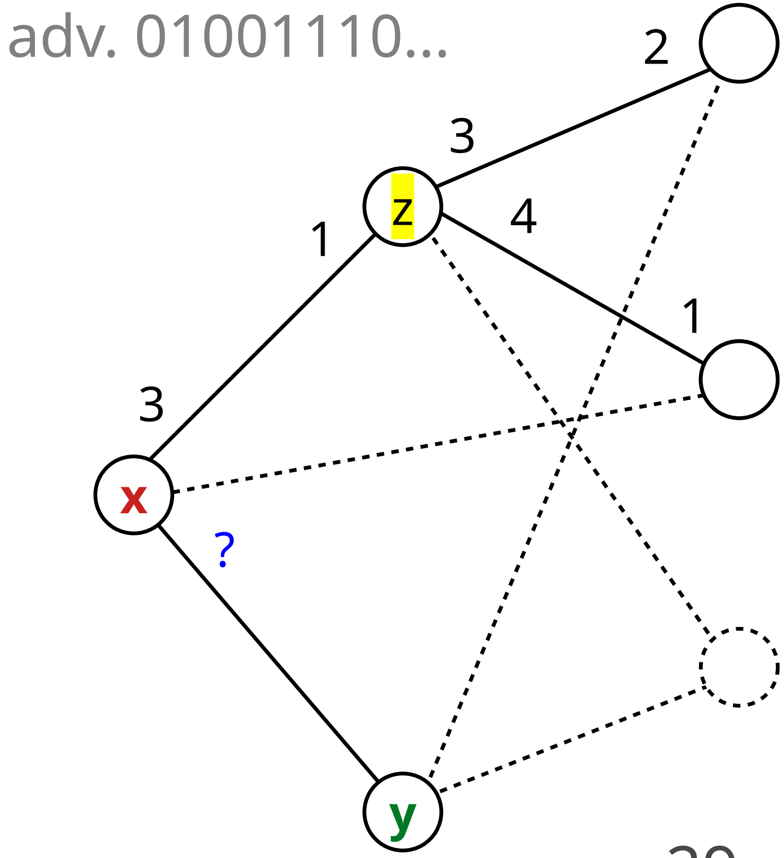
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



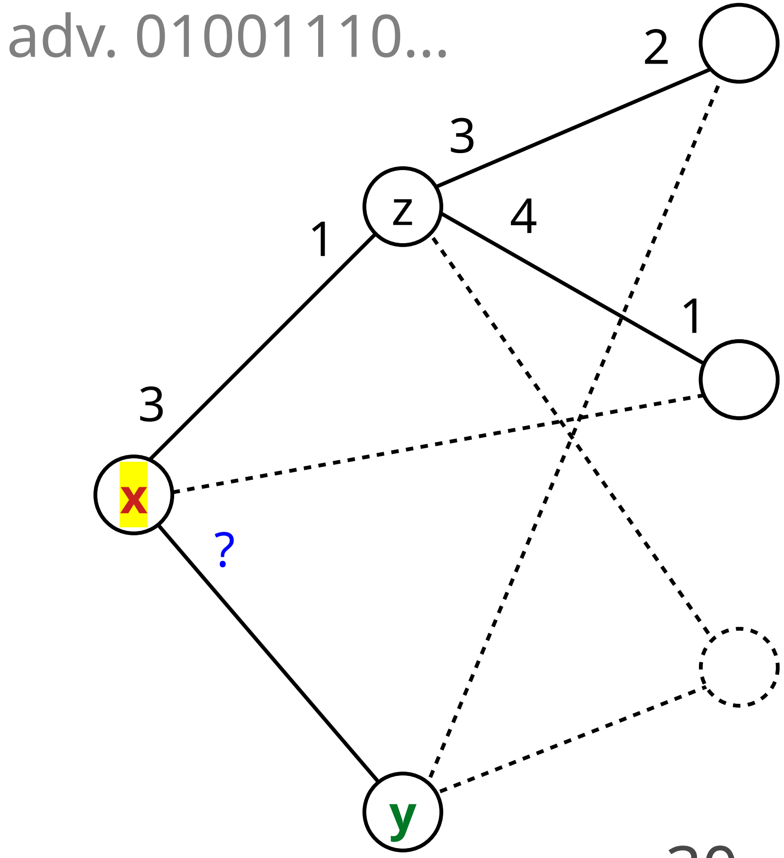
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**



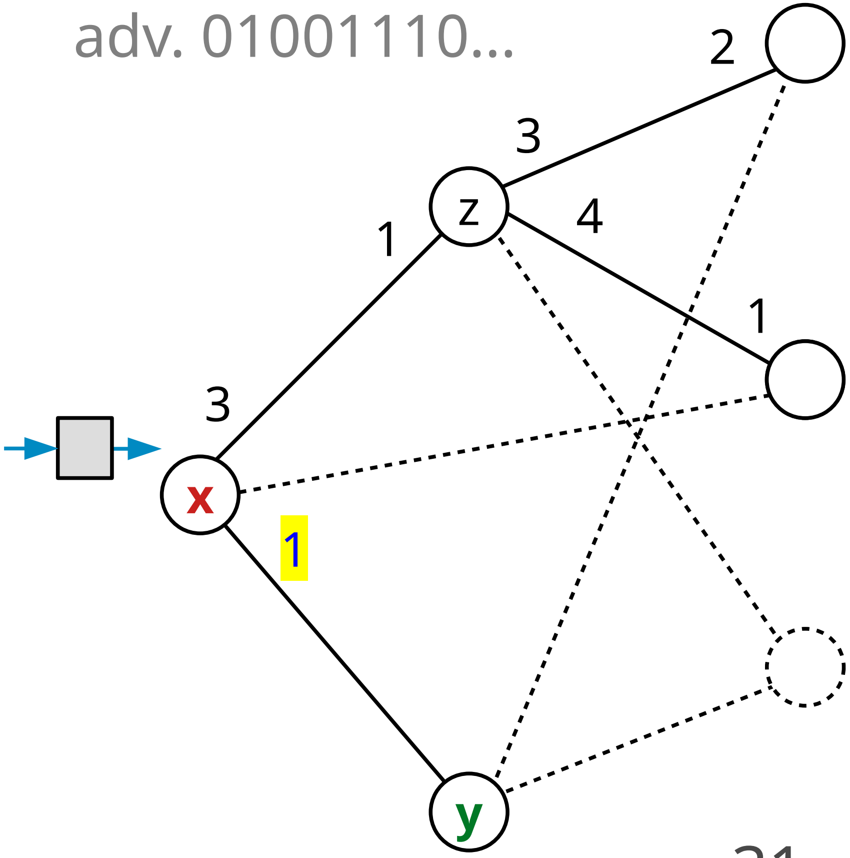
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex x with incomplete port assignment
  - Take neighbor y with unknown outgoing port
  - Simulate routing from x to y



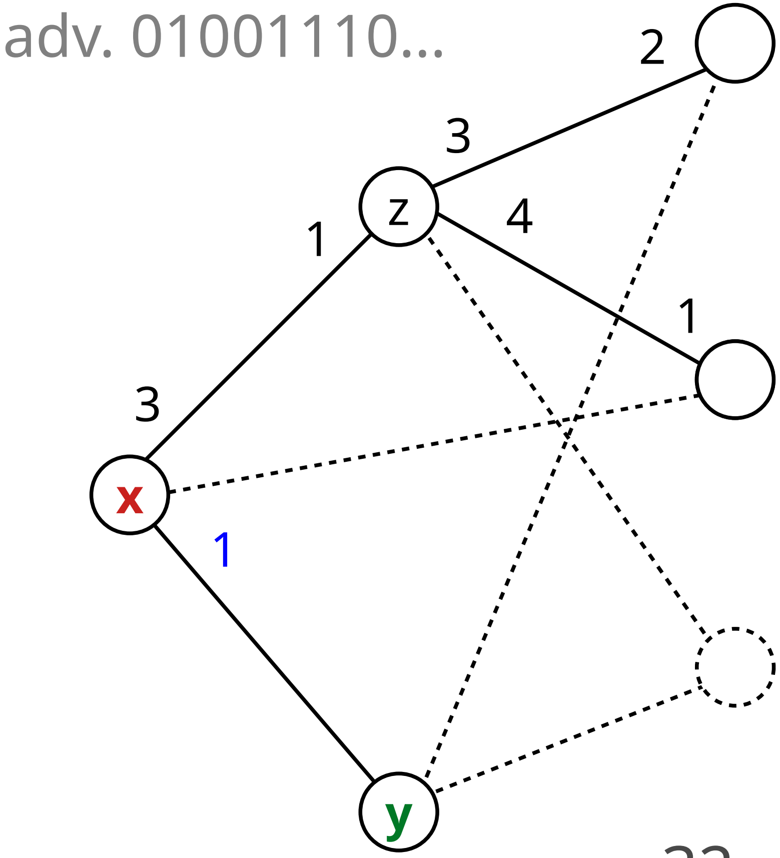
Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**
  - As last step, learn port at **x** toward **y**



Extractor program

- Start with unknown port assignment
- Repeat:
  - Take vertex **x** with incomplete port assignment
  - Take neighbor **y** with unknown outgoing port
  - Simulate routing from **x** to **y**
  - As last step, learn port at **x** toward **y**
- 1 in 4k ports learned from routing scheme



Results

- **Thm**: if graph with n vertices, m edges, girth 2k+2 exists, then routing with stretch $<2k+1$ requires $\Omega(m/n \log(m/n))$ bits at some node

Results

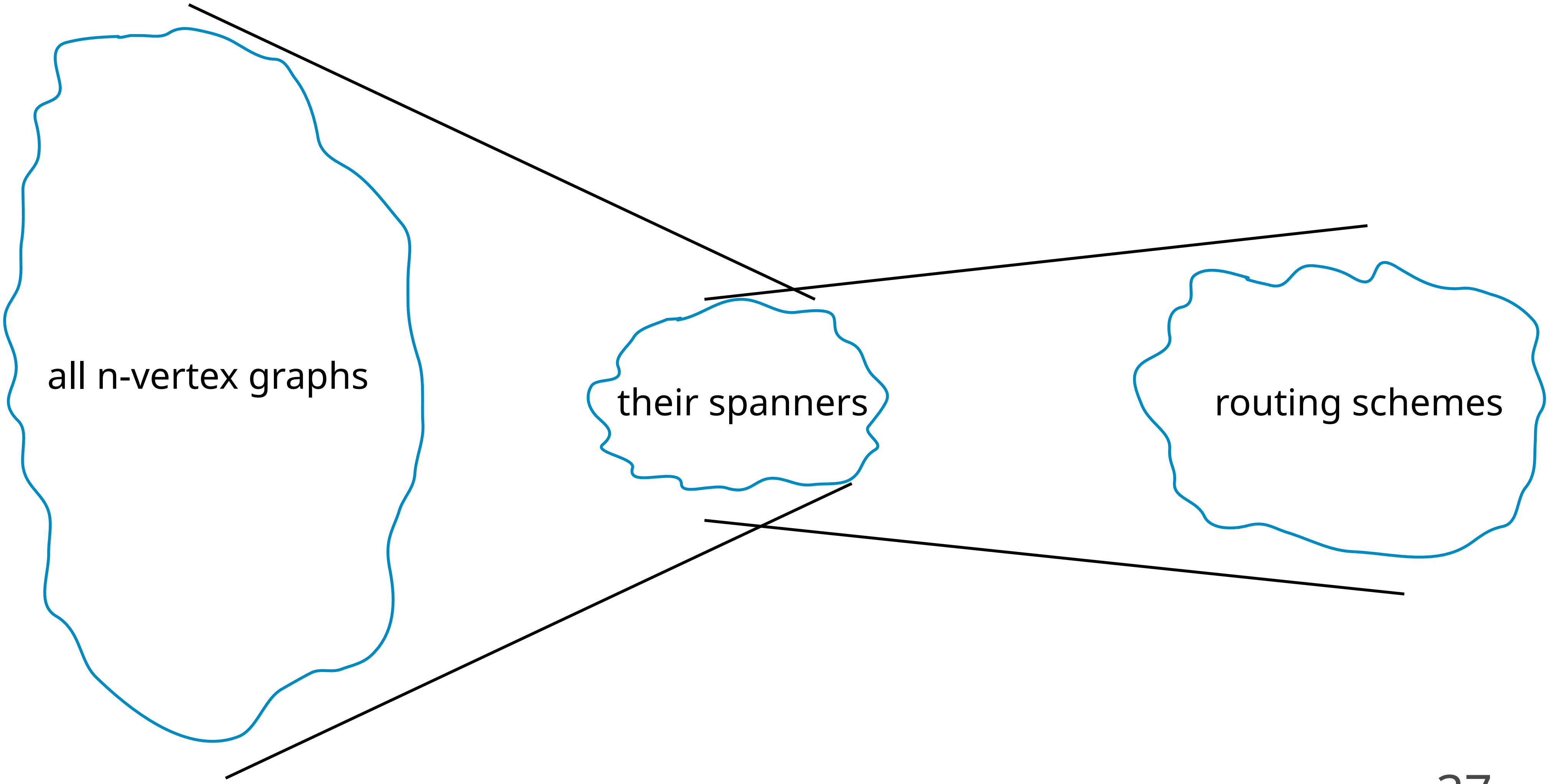
- **Thm**: if graph with n vertices, m edges, girth 2k+2 exists, then routing with stretch $<2k+1$ requires $\Omega(m/n \log(m/n))$ bits at some node
- **Cor**: assuming girth conjecture by Paul Erdős, routing with stretch $<2k+1$ requires $\Omega(n^{1/k} \log n)$ bits at some node

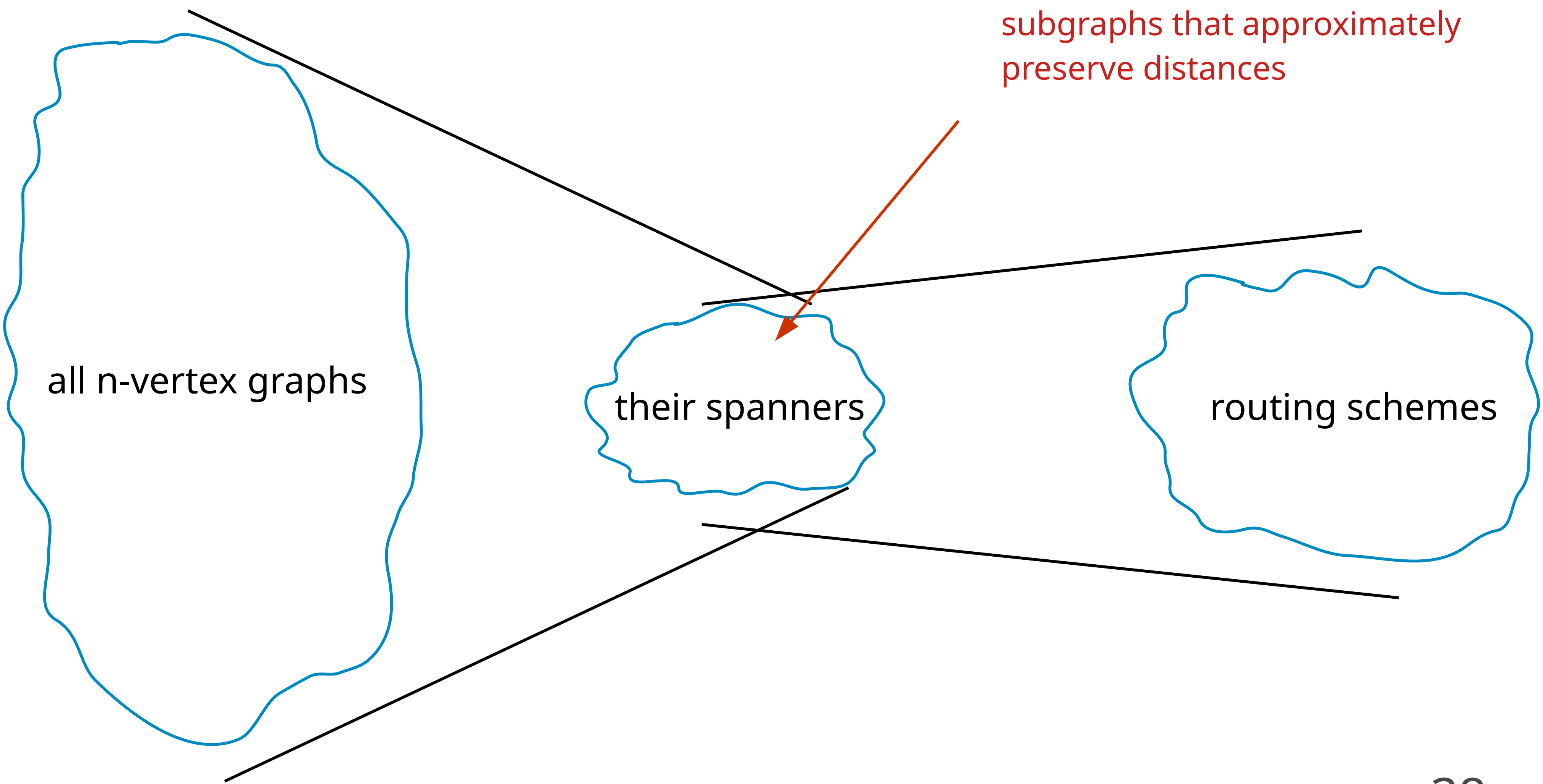
Results

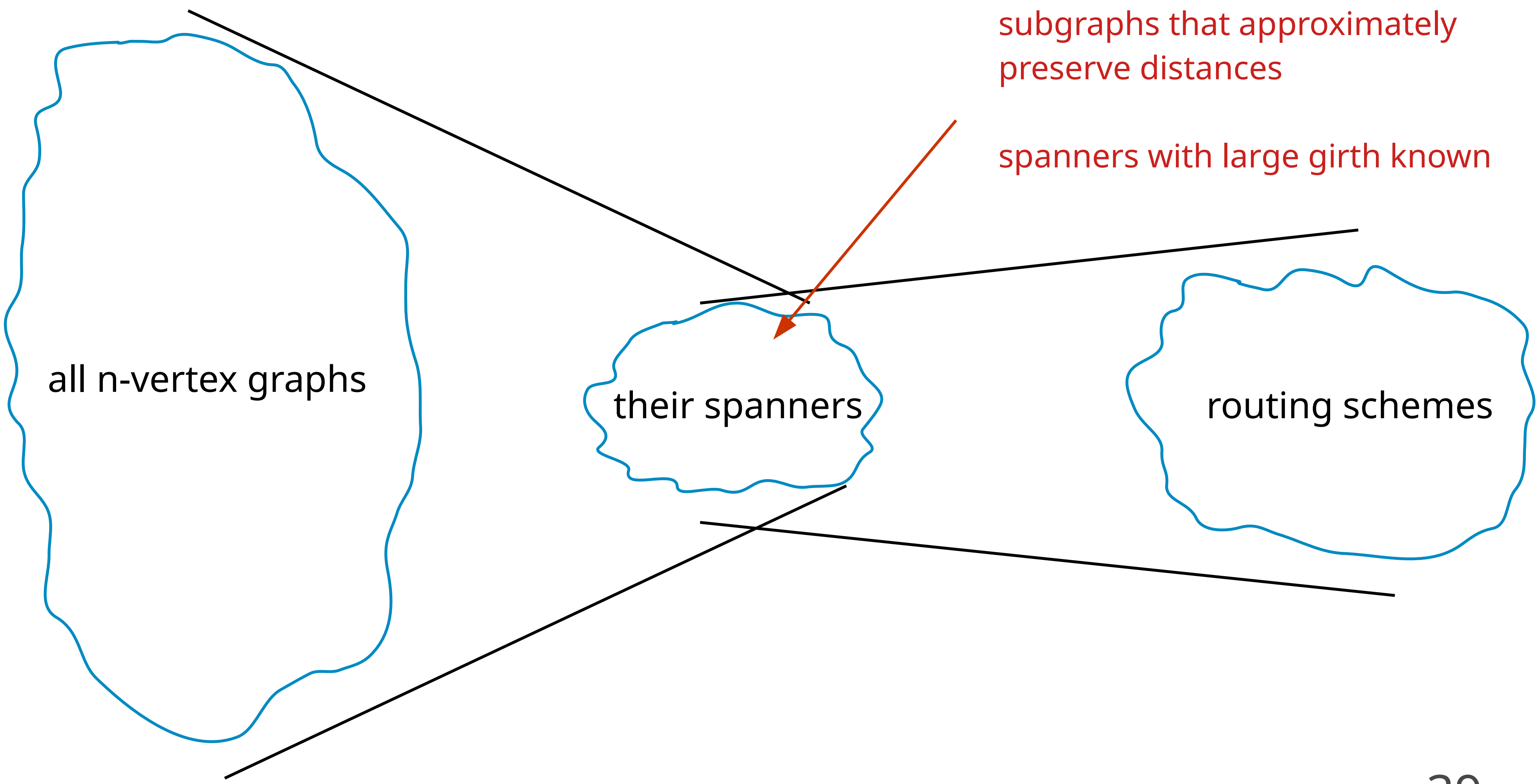
- **Thm**: if graph with n vertices, m edges, girth 2k+2 exists, then routing with stretch $<2k+1$ requires $\Omega(m/n \log(m/n))$ bits at some node
- **Cor**: assuming girth conjecture by Paul Erdős, routing with stretch $<2k+1$ requires $\Omega(n^{1/k} \log n)$ bits at some node
- Girth conjecture by Paul Erdős: there exists a graph with
  - n nodes
  - $\Omega(n^{1+1/k})$ edges
  - girth 2k+2
- Proven for k=1,2,3,5; weaker results for other k

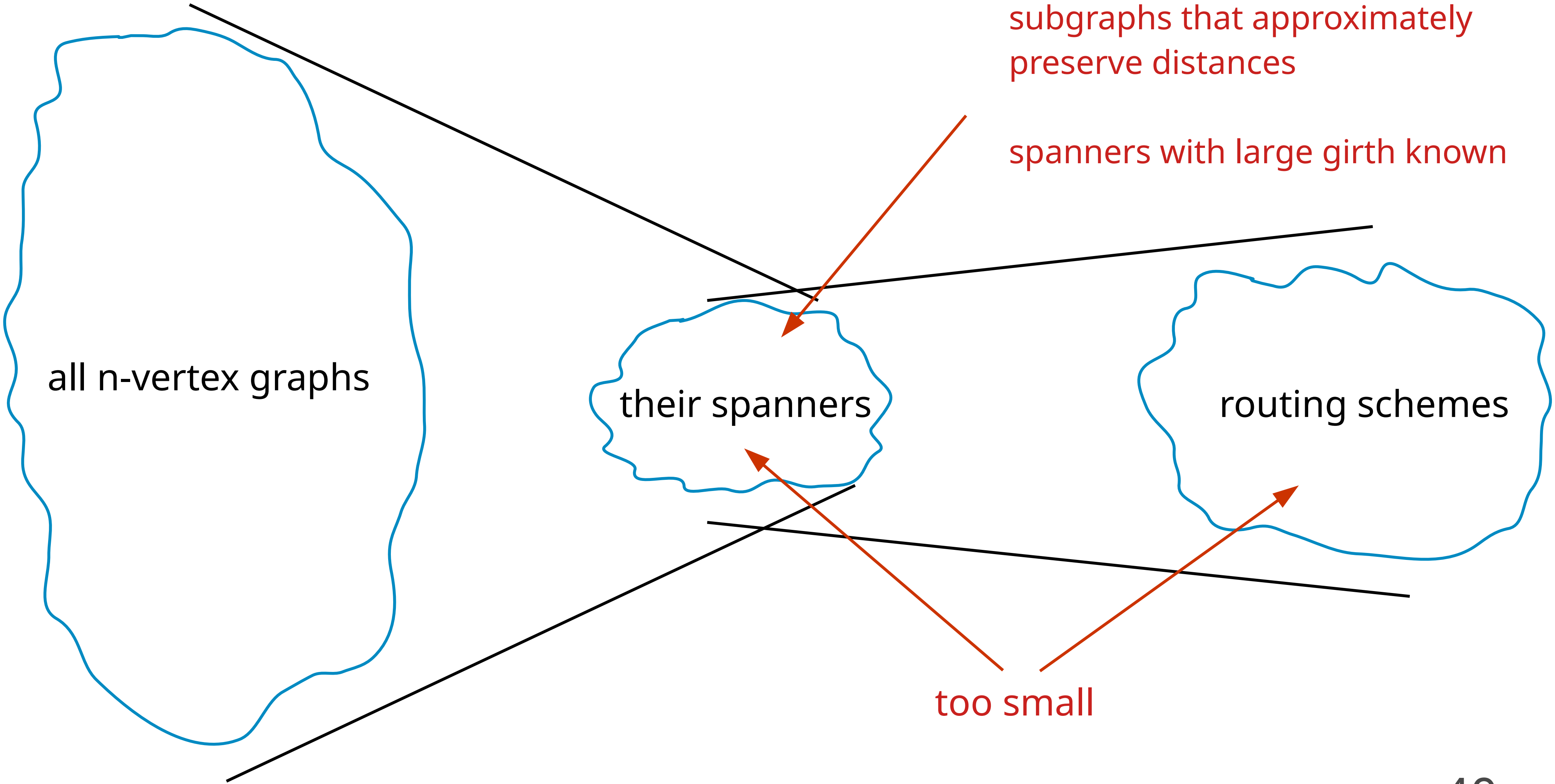
Girth conjecture requirement

- All known approaches: prove that many routing schemes are necessary (including last proof)
- If $2^{\Omega(n^{(1+1/k)} \log n)}$ routing schemes are needed to satisfy all n-vertex graphs with stretch $<2k+1$, then there exists a graph with $\Omega(n^{1+1/k})$ edges and girth $2k+2$


all n-vertex graphs
their spanners
routing schemes


subgraphs that approximately
preserve distances
all n-vertex graphs
their spanners
routing schemes


subgraphs that approximately preserve distances
spanners with large girth known
all n-vertex graphs
their spanners
routing schemes


subgraphs that approximately preserve distances
spanners with large girth known
all n-vertex graphs
their spanners
routing schemes
too small

Open problems

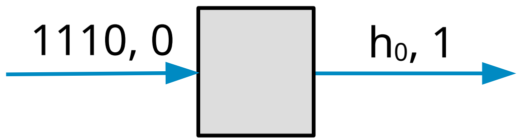
1. Can we overcome girth conjecture in labeled model?
    - (in name-independent model we can [1])

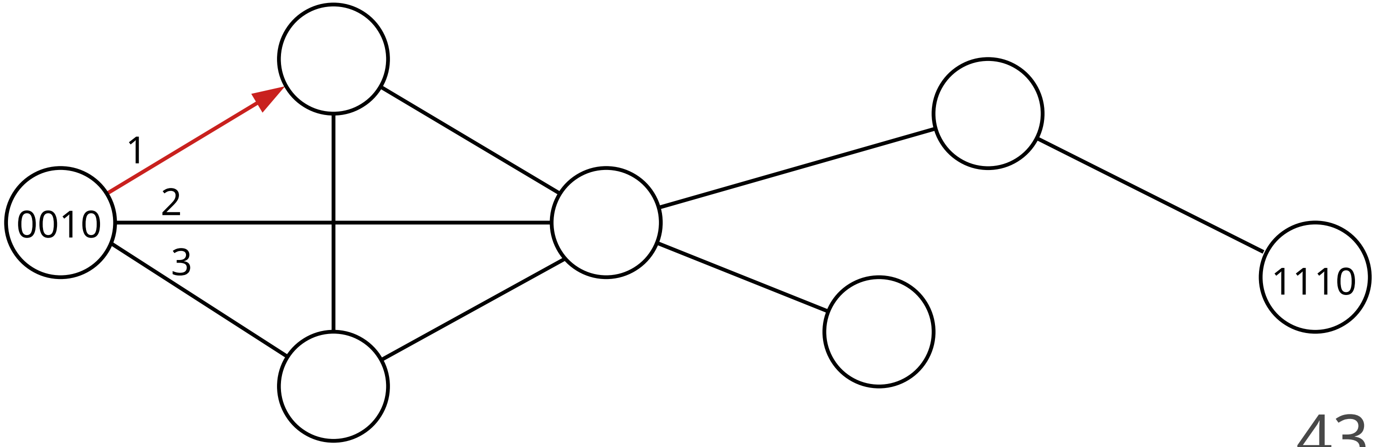
2. Unconditional (non-adversarial ports) lower bound for stretch ≥ 3?

[1]: Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Lower bounds. 2006

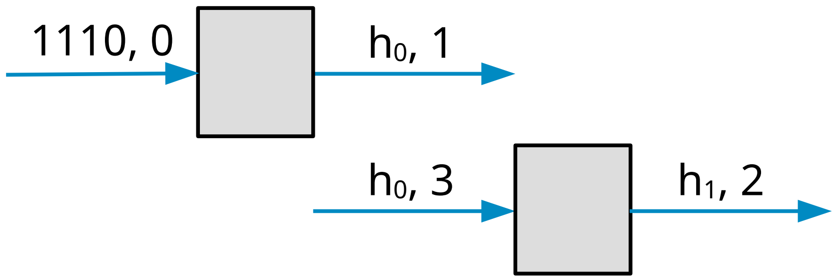
Extra slides

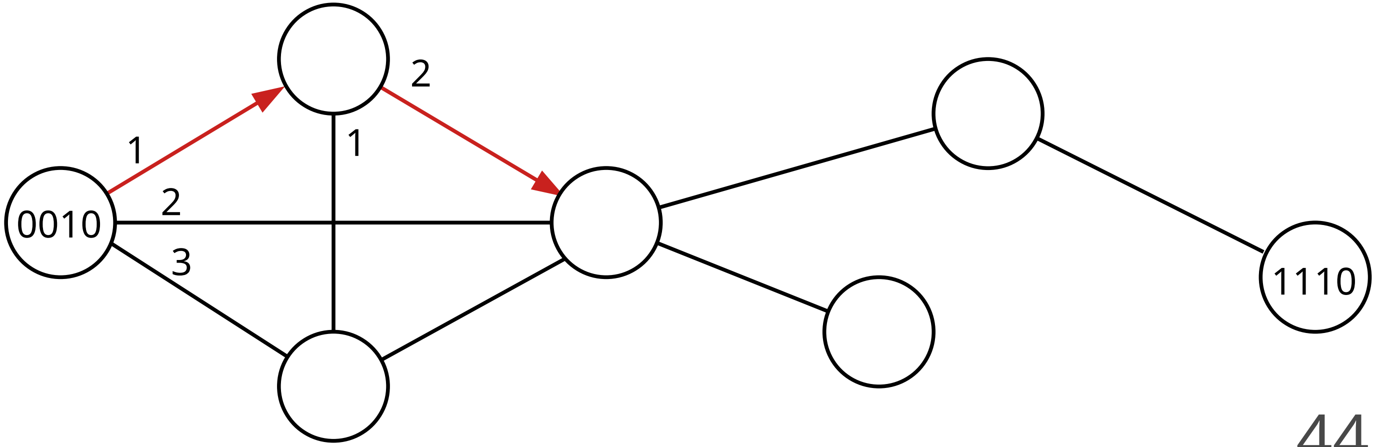
Routing in computer networks, example





Routing in computer networks, example





Routing in computer networks, example