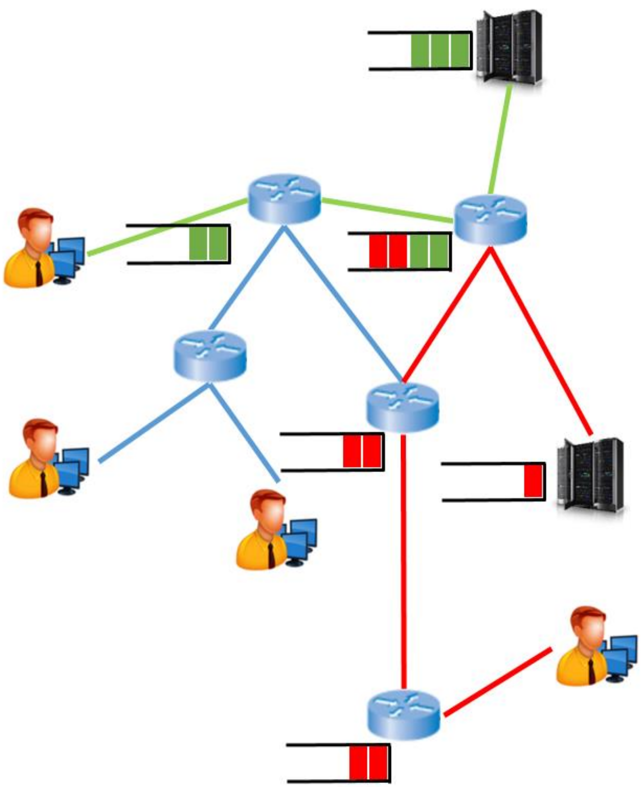
On the Robustness of Distributed Computing Networks

Jianan Zhang (MIT)
Hyang-Won Lee (Konkuk University)
Eytan Modiano (MIT)

- **Background**
  - Distributed computing network
  - Network flow interdiction
- **Computing network model**
  - Communication/computation resource constraints
- **Max flow and min cut**
  - Min communication/computation/joint cut
  - Gap between max flow and min cut
- **Flow interdiction**

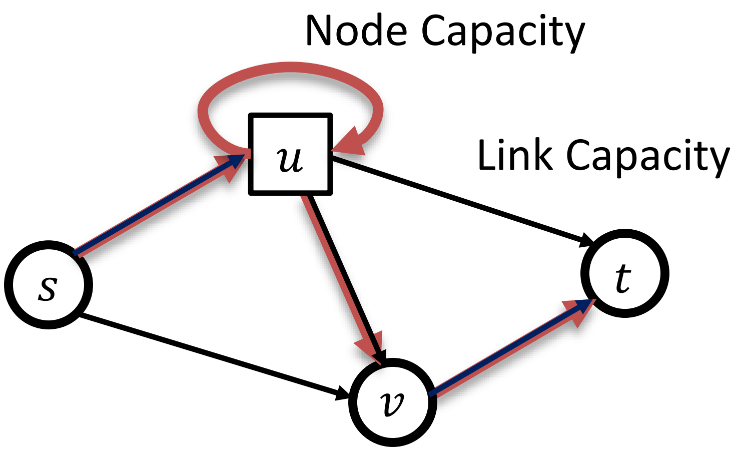
- **Traditional data network: transmitting data packets end-to-end**
  - Objective: maximizing throughput
  - Robustness metric: min-cut, max-flow
- **New network applications require both communication and computation**
  - Examples: cloud/fog computing, virtual reality streaming, content distribution network
- **Computing network failures**
  - Amazon Web Service failure due to power outage/software bug: 4 hour outage in 2017 cost 150 million dollars
  - Intentional attacks (DoS, etc.)



PREVIOUS WORK

- **Network robustness**
  - Max-Flow and min-cut [Dantzig, Fulkerson, 1956]
- **Network interdiction**
  - Minimizing max flow by removing links within a budget [Wood 1993; Phillips 1993; Burch et al. 2003]
- **Failure models**
  - Cross layer network robustness: WDM network [Medhi, Tipper 2000; Modiano, Narula-Tam 2001; Hu 2003; Lee, Modiano 2011]
  - Shared risk group model [Medhi 1994; Coudert et al. 2007]

- **Graph $G(V, E)$ represents a computing network**
  - Flow is processed at computation nodes $V_c \subseteq V$
  - Computation constraints at nodes V_c
  - Communication constraints at links E

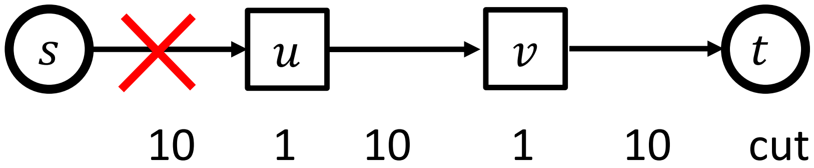


Computation node $V_c = \{u\}$
Communication links
$E = \{(s,u),(u,t),(s,v),(v,t)\}$

- **Links have communications capacity**
- **Nodes have computation capacity**
- **Robustness: Flow reduction due to node/link removals**

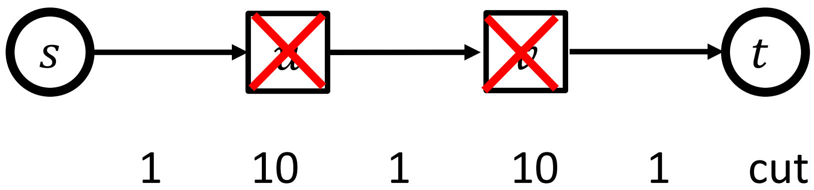
Robustness metric – Cut

- **Communication cut**



cut = 10, flow = 2

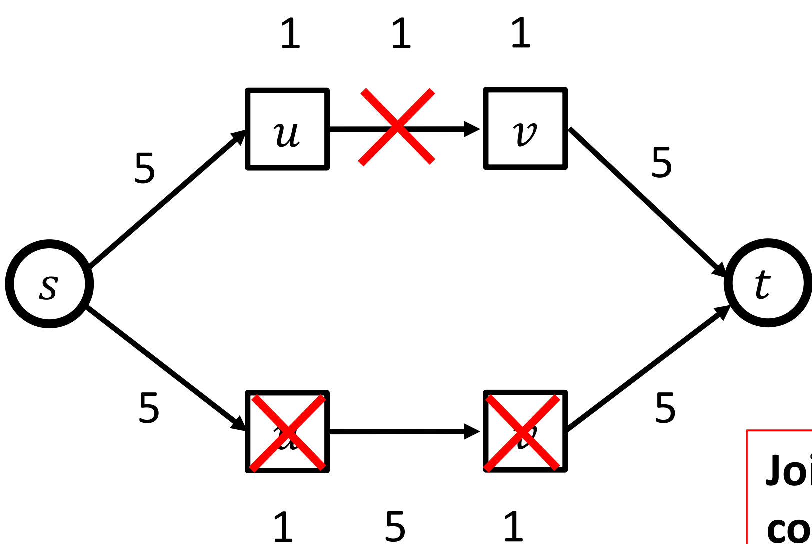
- **Computation cut**



cut = 20, flow = 1

Arbitrary gap between max-flow and min-cut

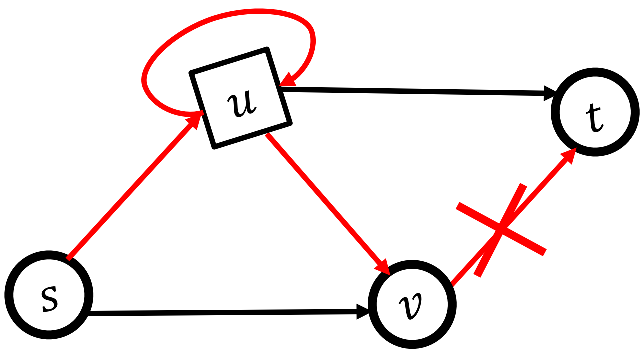
- **Joint communication and computation cut**



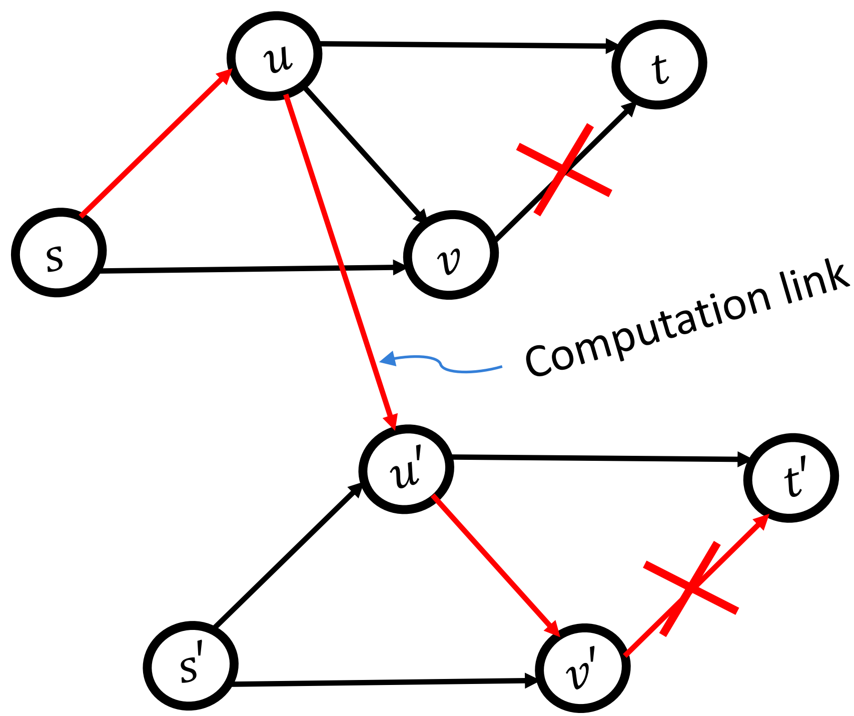
cut = 1 + 1 + 1 = 3
flow = 1 + 2 = 3

Joint cut is smaller than either communication or computation cut (gap at most factor of two)

- **Layered graph representation for a computing network**
  - Two layers: duplicate graph and add links between computation nodes
    - Link Capacity = computation node capacity
  - Flow must traverse computation link and depart at lower layer (t')
  - One-to-one mapping between an $s - t$ flow in the computing network and an $s - t'$ flow in the layered graph
  - One link cut in the computing network corresponds to two copies of the link cut in the layered graph



Flow on physical link (u, v) is the sum of flows on (u, v) and (u', v') in the layered graph – must obey capacity constraint



- **Assumptions**
  - No flow scaling due to computation
  - Normalization: each unit flow requires unit bandwidth for transmission and unit computation resource for processing

- **Linear program (based on the layered graph, polynomial time)**

layered graph $\tilde{G} = (\tilde{V}, \tilde{E})$

$$\max \quad f_{t's}$$

$$\text{s.t.} \quad \sum_{u \in \tilde{V}:(u,v) \in \tilde{E}} f_{uv} - \sum_{w \in \tilde{V}:(v,w) \in \tilde{E}} f_{vw} = 0, \forall v \in \tilde{V},$$

flow conservation

$$f_{ww'} \leq \mu_w, \quad \forall w \in V,$$

computation capacity constraint

$$f_{uv} + f_{u'v'} \leq \mu_{uv}, \quad \forall (u,v) \in E,$$

communication capacity constraint

$$f_{uv} \geq 0, f_{u'v'} \geq 0, \quad \forall (u,v) \in E,$$

$$f_{ww'} \geq 0, \quad \forall w \in V.$$

- **Complexity**

min computation cut	Polynomial time solvable
min communication cut	NP-hard (exact cover by 3-sets)
min joint cut	NP-hard (exact cover by 3-sets)

- **Integer programming for min joint cut**
  - Node potential approach

$$\min \quad \sum_{(u,v)\in E} \mu_{uv} y_{uv} + \sum_{w\in V} \mu_w y_w$$

$$\begin{aligned} \text{s.t.} \quad & p_v - p_u + y_{uv} \geq 0, \quad \forall (u,v) \in E, \\ & p_{v'} - p_{u'} + y_{uv} \geq 0, \quad \forall (u,v) \in E, \\ & p_{w'} - p_w + y_w \geq 0, \quad \forall w \in V, \\ & p_s - p_{t'} \geq 1, \\ & y_{uv} \in \{0,1\}, \quad \forall (u,v) \in E, \\ & y_w \in \{0,1\}, \quad \forall w \in V. \end{aligned}$$

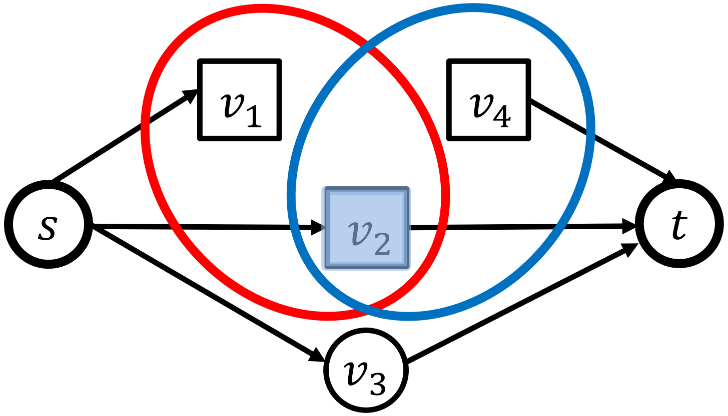
$y_{uv} = 1$: removing communication capacity at link (u, v).

$y_w = 1$: removing computation capacity at node w.

Compute the min communication cut by setting $y_w = 0, \ \forall w \in V$.

Compute the min computation cut by setting $y_{uv} = 0, \ \forall (u,v) \in E$.

- **Linear-time exact algorithm for computing the min computation cut**



Compute the set of computation nodes that can be reached from s. $\{v_1, v_2\}$

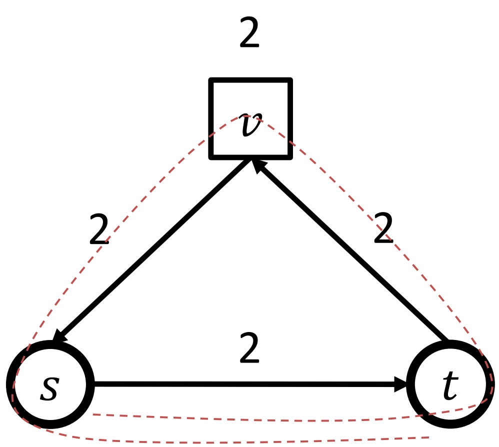
Compute the set of computation nodes that can reach t. $\{v_2, v_4\}$

Union of the two sets. $\{v_2\}$

- **Approximation algorithm for computing the min communication cut and the min joint cut**
  - Compute the min cut in the layered graph
  - Map the min cut to the original graph

- **Performance: 2-approximation**
  - The value of the cut computed by the approximation algorithm is at most twice the value of the min cut
  - Intuition: Cutting two different links in the layered graph "costs" twice as much as cutting the same link in both layers

- **Theorem: Min joint cut is at most twice the max flow**
  - Proof using the layered graph
- **Example: Min joint cut can be twice as large as the max flow**

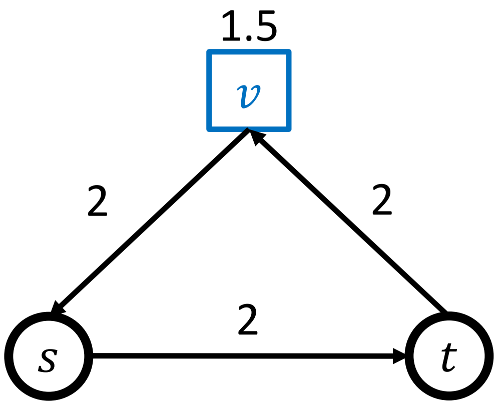


Max flow = 1:
The flow has to traverse link (s, t) twice.

Min joint cut = 2
Min communication cut = 2
Min computation cut = 2

- **Note: Added cycle needed for processing at node v**
  - A flow will traverse the same link at most twice: Once before processing and once after processing

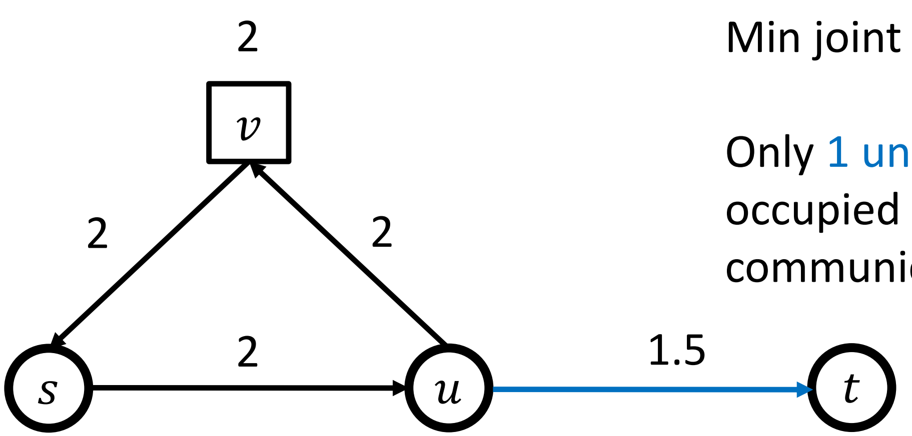
- **In the standard flow model, edges in min cuts are saturated by the max flow. This does not hold in a computing network.**
- **Unsaturated node in min joint cut**



Min joint cut: node v

Only 1 unit computation resource is occupied by the max flow. 0.5 unit computation resource remains idle.

- **Unsaturated link in min joint cut**



Min joint cut: link (u, t)

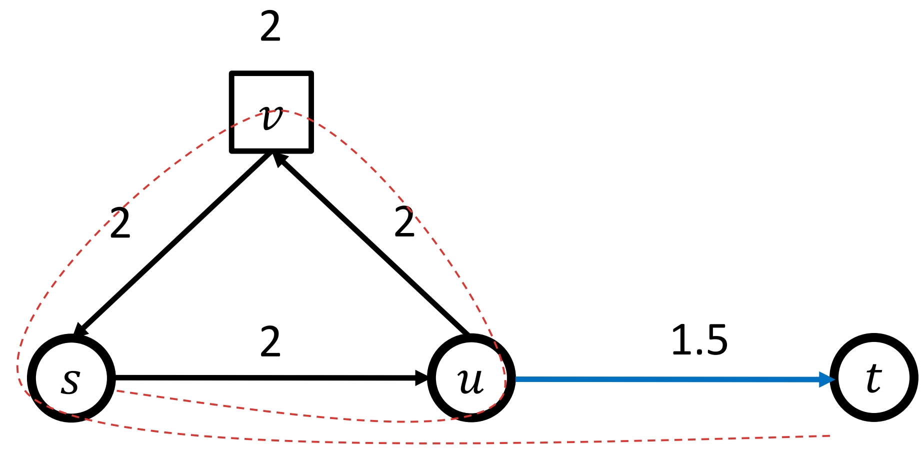
Only 1 unit communication resource is occupied by the max flow. 0.5 unit communication resource remains idle.

- **Minimizing the max flow by removing communication and computation resources within a given budget (B)**
  - Remove any combination of communication and computation resources
  - Interdiction cost can be either equal to the removed capacity, or independent of the removed capacity (arbitrary)
  - Interdiction type: either removes the entire link/computation resource at a node at a fixed cost (binary), or removes a fractional capacity at a fractional cost (partial)

	interdiction cost = removed capacity	**arbitrary cost**
binary interdiction	NP-hard	NP-hard
partial interdiction	?	NP-hard

- Special cases of traditional interdiction problems that are NP hard

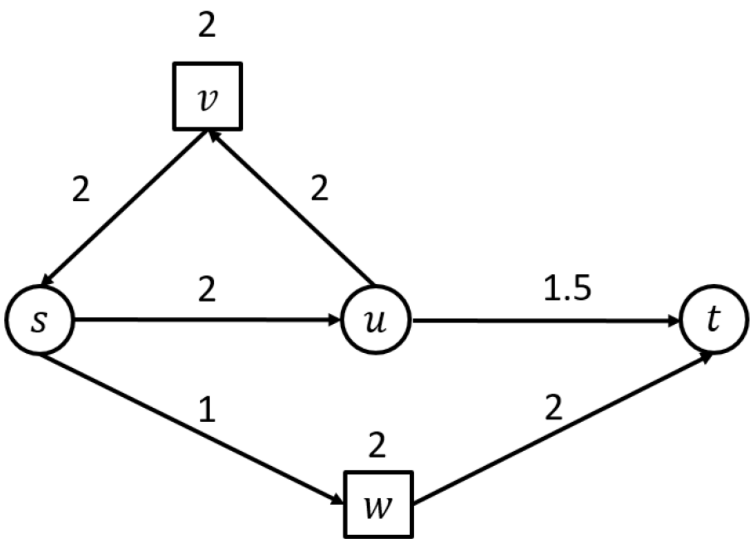
- **In traditional model, the amount of max-flow decrease equals the amount of removed capacity in the min cut**
  - In a computing network, since links or nodes in the min cut are not saturated by the max flow, attacking the min cut may not be optimal
  - The optimal attack strategy depends on the budget
- **Budget B ≤ 1**
  - Optimal strategy: interdict link (s,u)
  - Max-flow will be (2-B)/2
  - Max-flow decreases at rate 1/2
- **B ≥ 1**
  - Optimal strategy: interdict link (u,t)
  - Max-flow will be 1.5-B
  - Max-flow decreases at rate 1

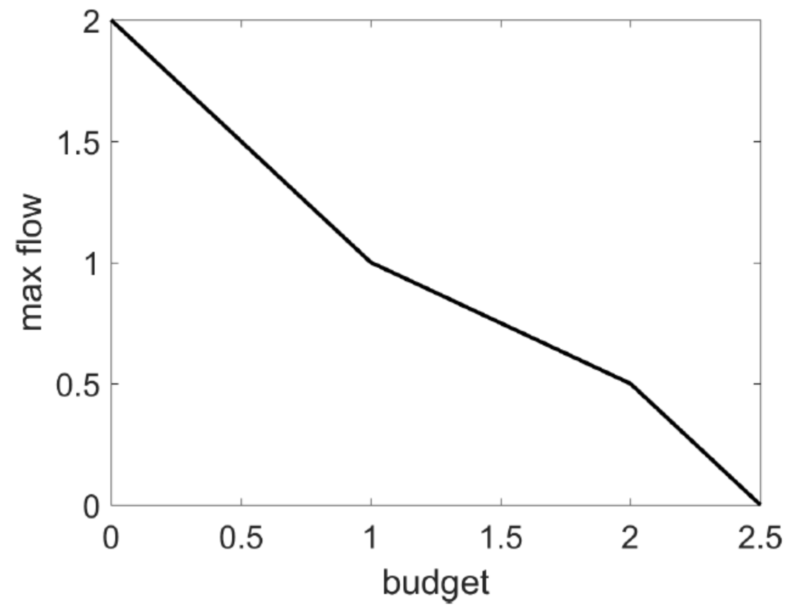


- **Observations:**
  - 1) The strategy changes as a function of the budget B
  - 2) The rate of max-flow decrease changes with B
  - 3) Optimal strategy does not necessarily interdict the minimum-cut

NON-MONOTONE FLOW REDUCTION RATE

- **The rate of max flow decrease is non-monotone as communication and computation resources are removed**





budget < 1: remove partial communication resource at (s, w)
1 < budget < 2: remove all communication resource at (s, w); remove partial communication resource at (s, u)
2 < budget < 2.5: remove all communication resource at (s, w); remove partial communication resource at (u, t)

- **Consider the max flow LP again**

$$\begin{aligned}
\max \quad & f_{t's} \\
\text{s.t.} \quad & \sum_{u\in\tilde{V}:(u,v)\in\tilde{E}} f_{uv} - \sum_{w\in\tilde{V}:(v,w)\in\tilde{E}} f_{vw} = 0, \forall v \in \tilde{V}, \\
& f_{ww'} \leq \mu_w, \quad \forall w \in V, \\
& f_{uv} + f_{u'v'} \leq \mu_{uv}, \quad \forall (u,v) \in E, \\
& f_{uv} \geq 0, f_{u'v'} \geq 0, \quad \forall (u,v) \in E, \\
& f_{ww'} \geq 0, \quad \forall w \in V.
\end{aligned}$$

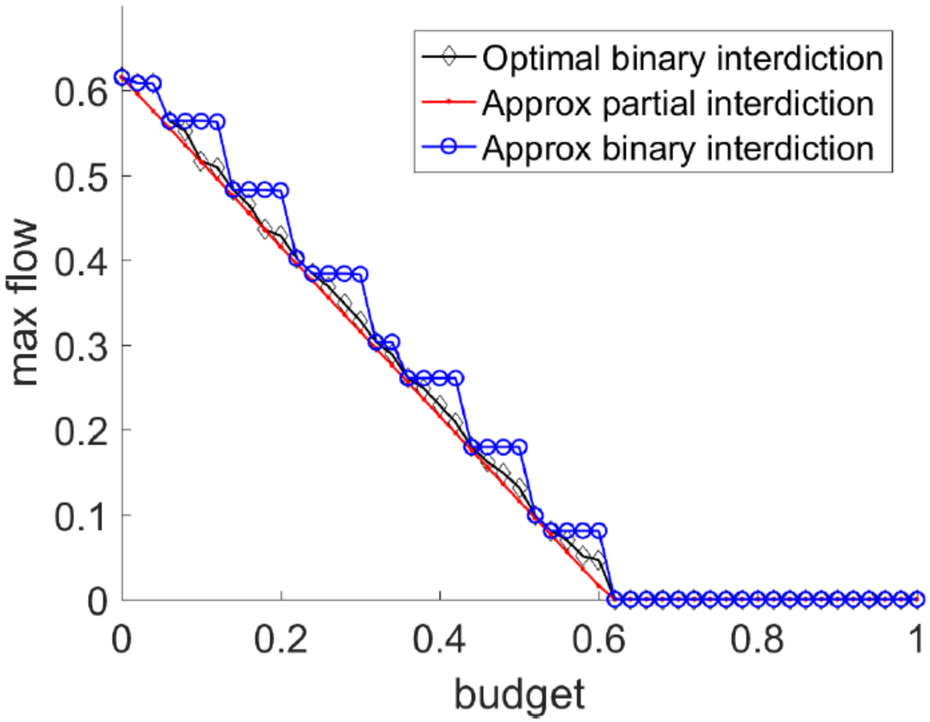
computation capacity constraint q_w

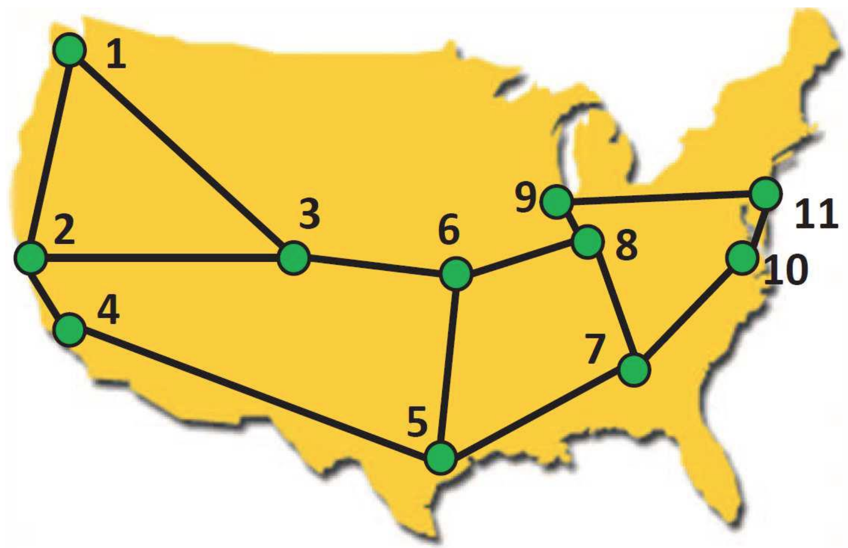
communication capacity constraint q_{uv}

- **Shadow prices**
  - Rate of change of the objective for one unit change of the right-hand side value of the constraint
  - q_w: rate of max-flow decrease for each unit processing capacity decrease at node w
  - q_{uv}: rate of max-flow decrease for each unit transmission capacity decrease at link (u, v)

- **Greedy binary interdiction (cost equals capacity):**
  - Iteratively cut a link or node that has the largest shadow price (within budget)
- **Greedy binary interdiction (arbitrary cost):**
  - Iteratively cut a link or node that has the largest cost-efficiency (within budget): shadow price * capacity / interdiction-cost
- **Partial interdiction**
  - Same as above, reduce capacity of selected link up to budget
- **Exact solution ILP**
  - Algorithm based on duality from Minimum Fractional Cut ILP

- **Approximation algorithms have good performance**
  - Abilene network (11 nodes, 14 links)
  - Randomly generated capacities and costs
  - Source node 1, destination node 2

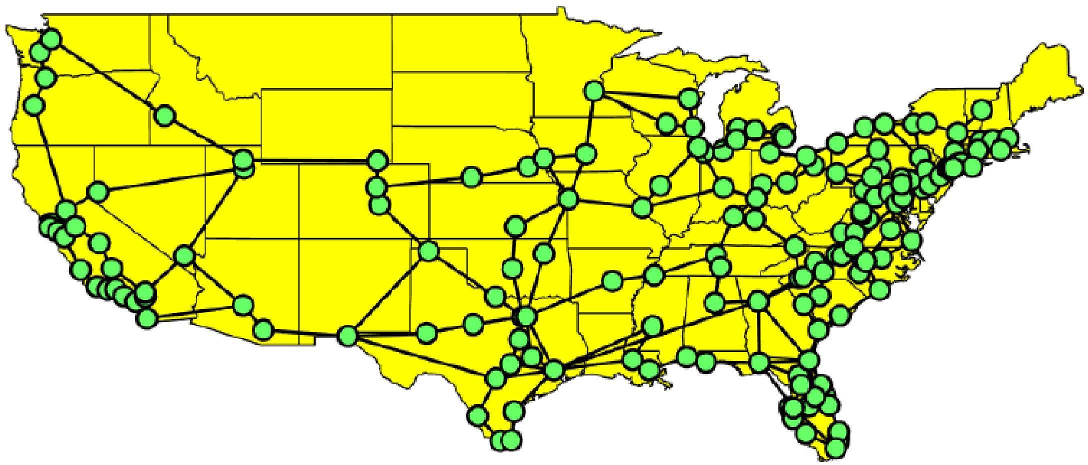




Cost = removed capacity

Greedy algorithm has good performance

- **CenturyLink network (170 nodes, 230 links)**



- **Running time comparison**
  - 10 randomly chosen s-t pairs. 6 levels of budgets. Total: 60 instances
  - Exact solution (ILP) fails to output a solution within 10 minutes for 54 instances
  - Greedy algorithm outputs a solution usually within a few seconds

- **Model for a distributed computing network**
  - Both communication and computation resource constraints

- **Robustness metrics**
  - Complexity analysis of Min communication/computation/joint cut
  - Algorithms for computing max flow and min cuts
  - Arbitrary gap between max flow and communication/computation cut
  - Factor of two gap between max flow and joint cut

- **Network flow interdiction problems**
  - Formulations for budgeted flow interdiction problem
  - Exact solution (ILP) and greedy algorithms for interdiction

- **Min joint cut is at most twice the max flow**

- **Exact solution (ILP) to binary interdiction**

Proof that the min cut is at most twice the max flow

- **Layered graph**

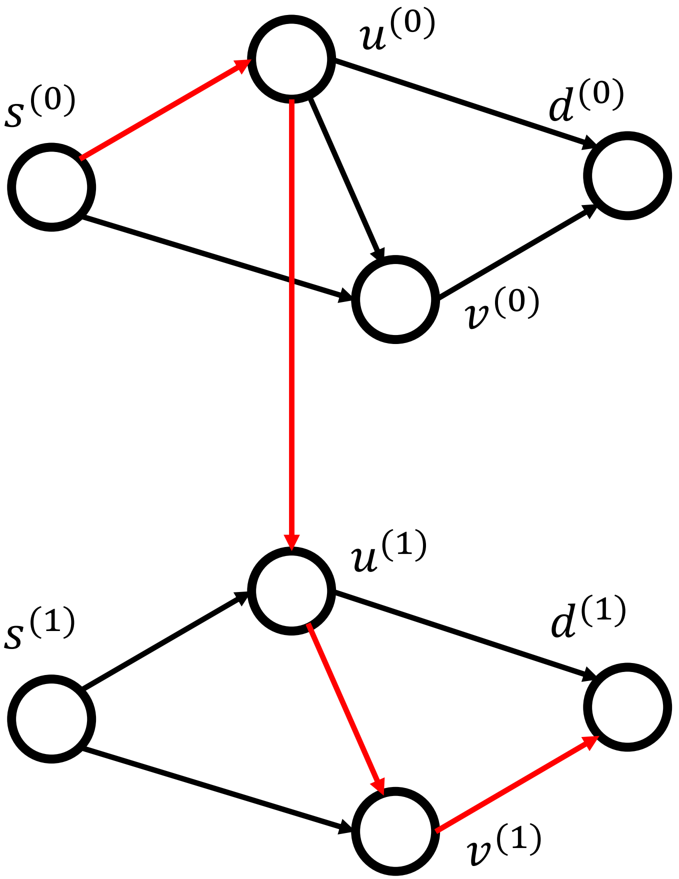
Relaxing the communication capacity constraint in the layered graph G'. Treat G' as the classical flow network model.

Max flow value in G' is at most twice the max flow value in G.

Min cut value in G' equals the max flow value in G'.

Min cut value in G is at most the min cut value in G'.

Min cut value in G is at most twice the max flow value in G.



- **Variables**
  - z_{uv}: whether link (u, v) is removed
  - z_w: whether computation resource at node w is removed
  - $\mu_{uv}\beta_{uv}$: amortized amount of flow contributed by link (u, v)
  - $\mu_w\beta_w$: amortized amount of flow contributed by node w

$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in E} \mu_{uv}\beta_{uv} + \sum_{w\in V} \mu_w\beta_w \\
\text{s.t.} \quad & p_v - p_u + \beta_{uv} + z_{uv} \geq 0, \quad \forall (u,v) \in E \\
& p_{v'} - p_{u'} + \beta_{uv} + z_{uv} \geq 0, \quad \forall (u,v) \in E \\
& -p_w + p_{w'} + \beta_w + z_w \geq 0, \quad \forall w \in V \\
& p_s - p_{t'} \geq 1, \\
& \sum_{(u,v)\in E} c_{uv} z_{uv} + \sum_{w\in V} c_w z_w \leq B, \\
& 0 \leq \beta_{uv} \leq 1, z_{uv} \in \{0,1\}, \quad \forall (u,v) \in E, \\
& 0 \leq \beta_w \leq 1, z_w \in \{0,1\}, \quad \forall w \in V.
\end{aligned}
$$