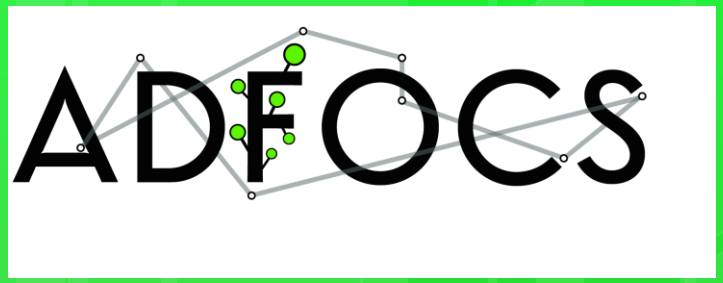
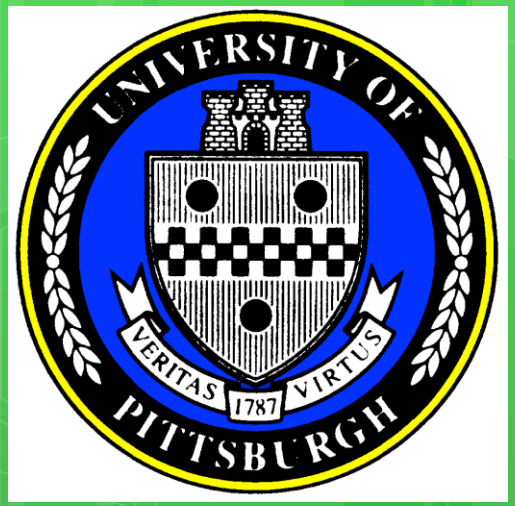


Kirk Pruhs



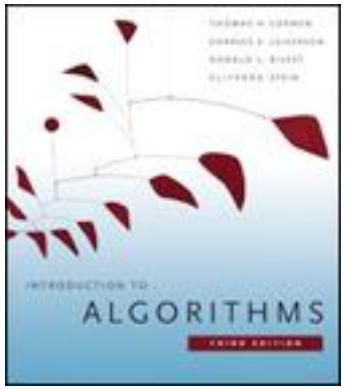
Green Computing Algorithmics

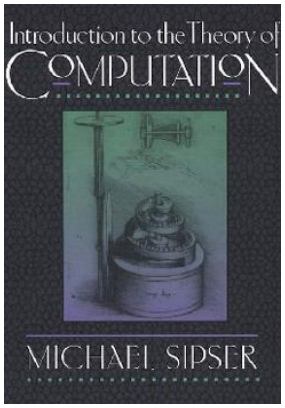
Talk 2: Energy Efficient Routing

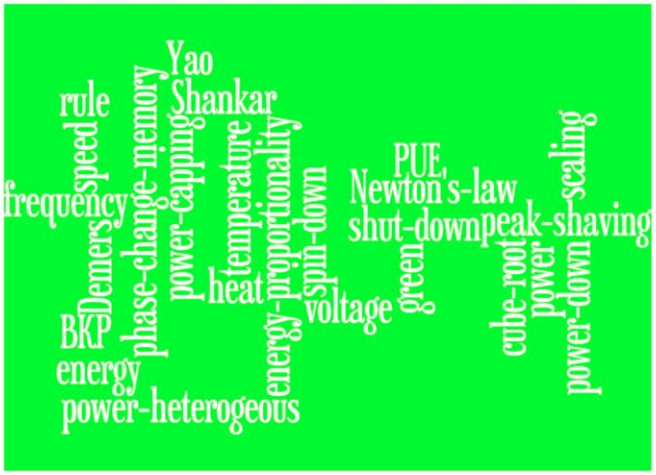
ADFOCS 2015

My Sales Pitch for Green Computing Algorithms

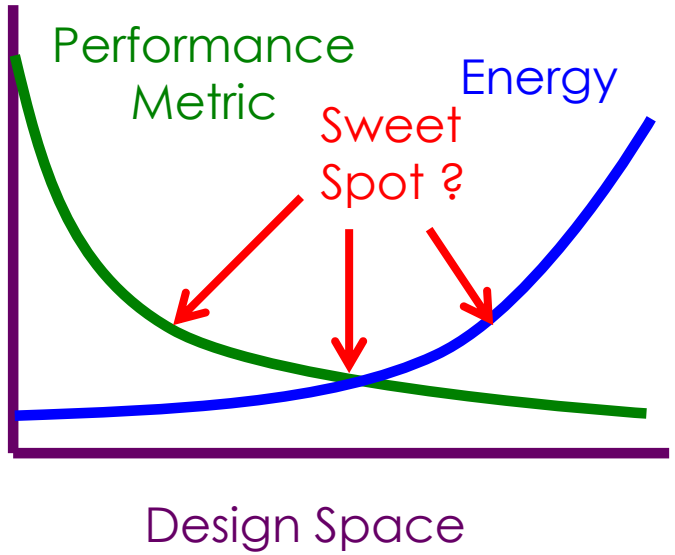
- Build a theory of energy as a computational resource that allows software engineers to reason abstractly about power, energy and temperature as effectively as they can currently abstractly reason about time and space





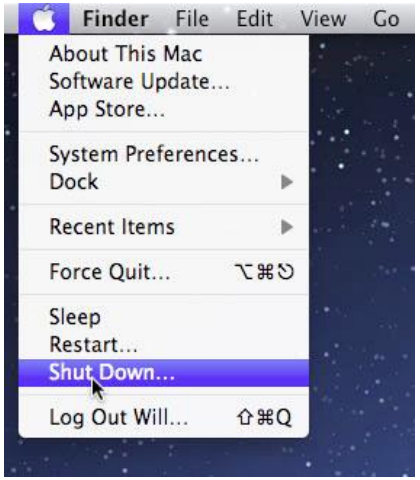


Current State Of Theory of Energy as a Computational Resource: Energy vs. Performance Tradeoffs



Common Energy Management Mechanisms

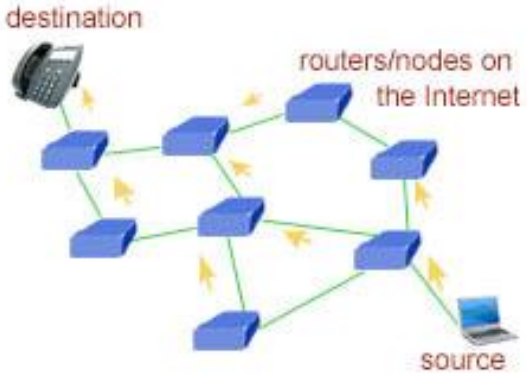
- Shutdown



- Intra-device power heterogenity
- Inter-device power heterogenity

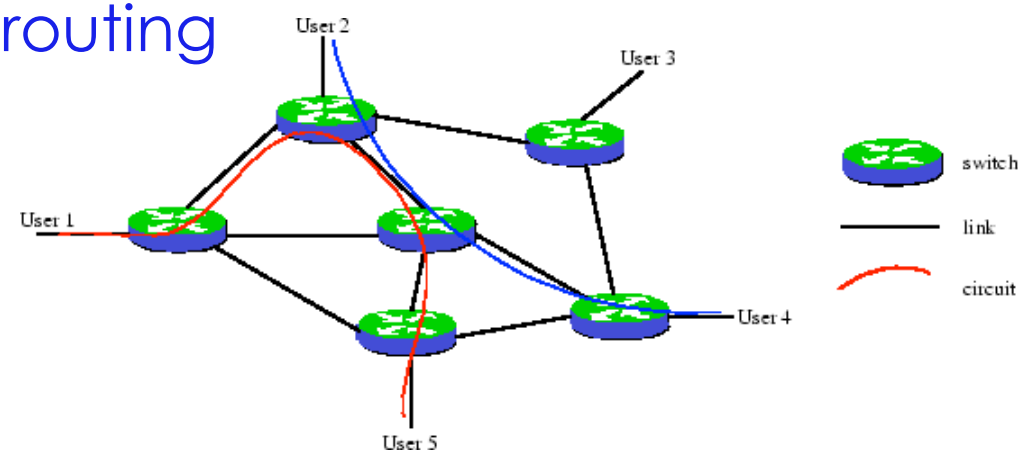
Network Routing Paradigms

- Datagram packet routing

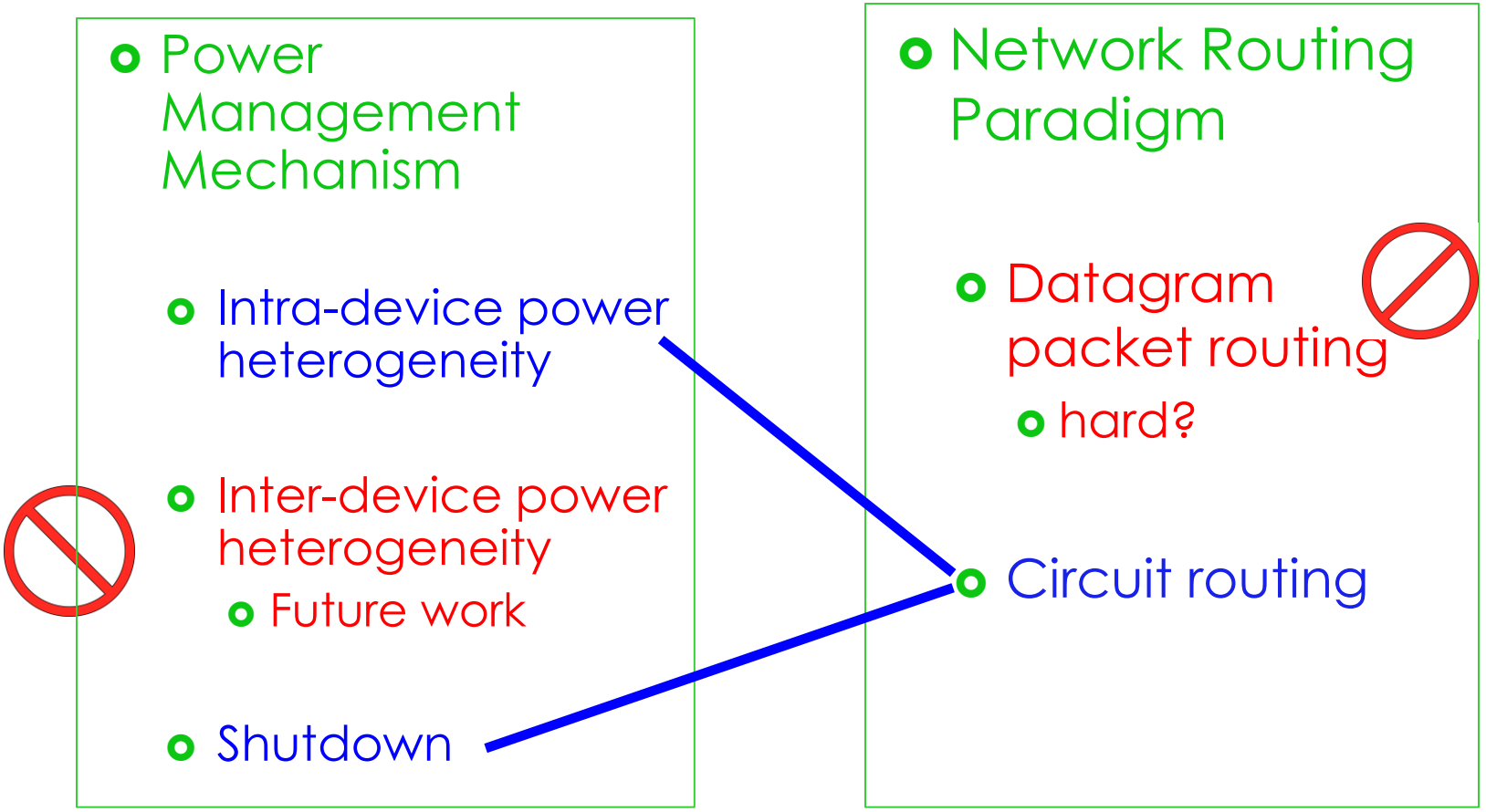


IP packets can take different routes

- (Virtual) circuit routing

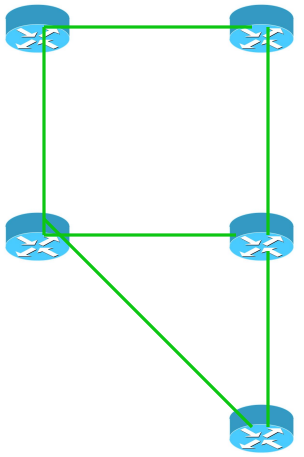


Energy Efficient Network Routing Research Program: This Talk

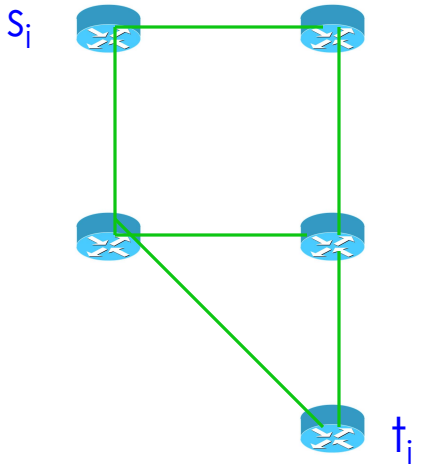


Circuit Model

- Network = undirected multigraph



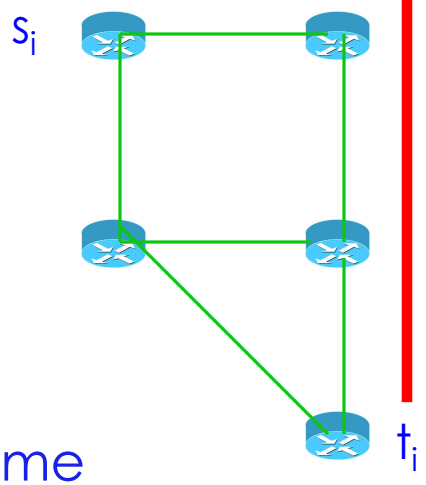
Circuit Model



- Network = undirected multigraph

- Input:
  - Requests for connections arrive over time
  - Request i consists of:
    - Source node s_i
    - Destination node t_i
    - Load (without great loss of generality assume unit loads for this talk)

Circuit Model



- Network = undirected multigraph
- Input:
  - Requests for connections arrive over time
  - Request i consists of:
    - Source node s_i
    - Destination node t_i
    - Load (without great loss of generality assume unit loads for this talk)
- Output: In response to request i, a (s_i, t_i) path must be specified

Standard Energy Model

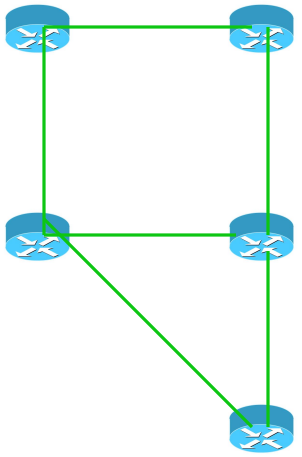
- Power = static power + dynamic power
  - = σ+ speed$^{\alpha}$
    - Speed in [0, ∞)
  - = σ+ load$^{\alpha}$

- A shutdown device uses no power

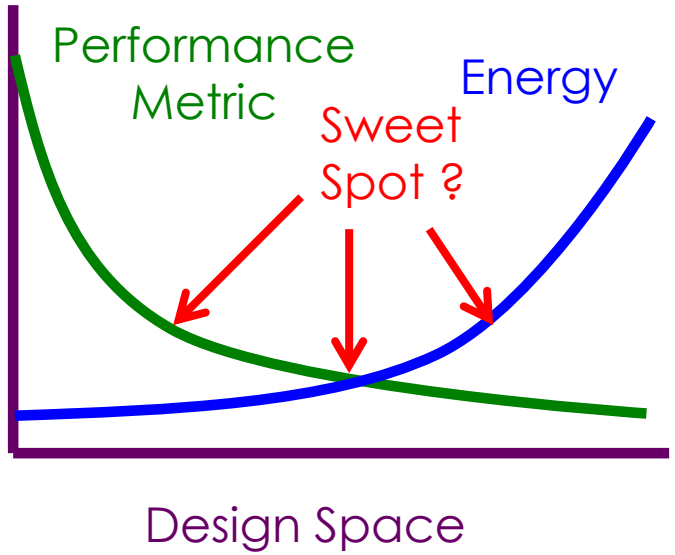
- For reasons of mathematical tractability, assume power management happens on edges
  - Later we'll say something about about the case where power management happens on nodes

Energy Efficient Routing Problem [AAZ2010]

- Feasible Solution: A routing of all requests
- Objective: Minimize aggregate power over all edges
  - $= \Sigma_{\text{edges e powered on}} (\sigma + \text{load}(e)^{\alpha})$



Current State Of Theory of Energy as a Computational Resource: Energy vs. Performance Tradeoffs



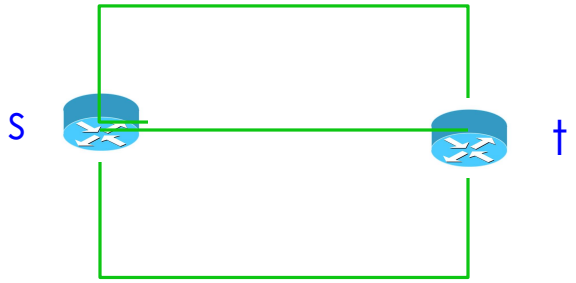
Roadmap

- Many warmup problems to build intuition
- Offline algorithm almost full analysis
- Online algorithm and hint at analysis
- Merest of hints on difficulty of node based routing

Warm-up Problem 1:

$\sigma + load^{\alpha}$

- Assume static power $\sigma = 0$
- Assume all $s_i = s$
- Assume all $t_i = t$

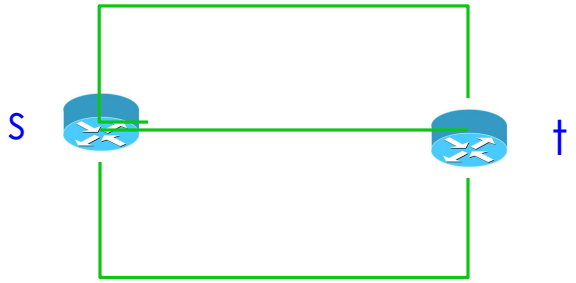


- Question: What is the optimal solution for this network?

Warm-up Problem 1:

$\sigma + \text{load}^{\alpha}$

- Assume static power $\sigma = 0$
- Assume all $s_i = s$
- Assume all $t_i = t$

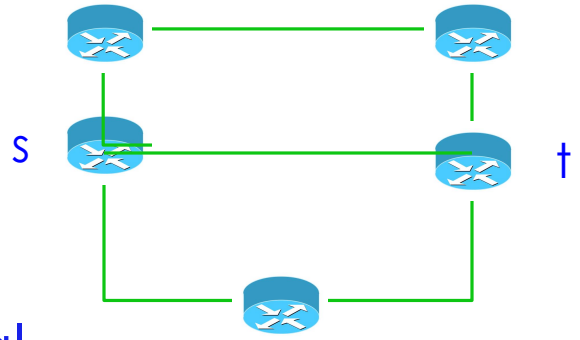


- Question: What is the optimal solution for this network?
- Answer: Put 1/3 of the paths on each edge

Warm-up Problem 2:

$\sigma + \text{load}^{\alpha}$

- Assume static power $\sigma = 0$
- Assume all $s_i = s$
- Assume all $t_i = t$

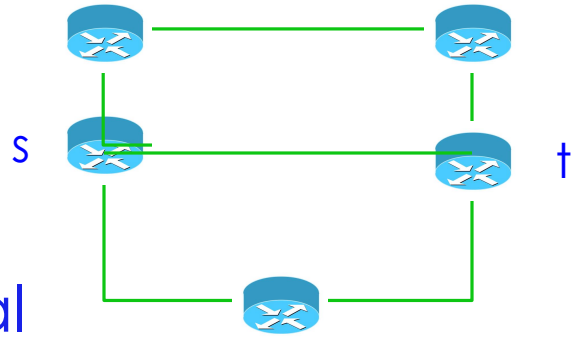


- Question: What is the optimal solution for this network, and how would you compute it?

Warm-up Problem 2:

- Assume static power $\sigma = 0$
- Assume all $s_i = s$
- Assume all $t_i = t$

$\sigma + \text{load}^{\alpha}$



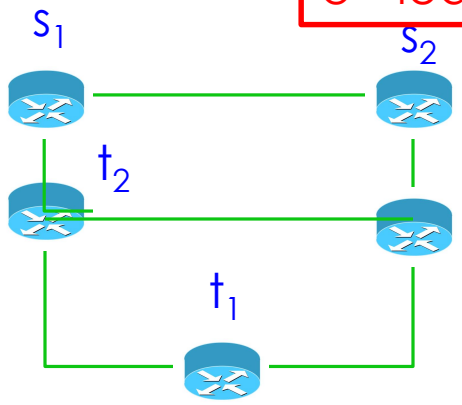
- Question: What is the optimal solution for this network and how would you compute it?

- Answer: The aggregate hypopower (derivative of power with respect to speed) on each of the three s-t paths should be identical. Greedy.

Warm-up Problem 3:

- Assume static power $\sigma = 0$
- Question: What is a reasonable candidate online algorithm for a general instance?
  - Hint: Generalize the solution for the previous warmup problem

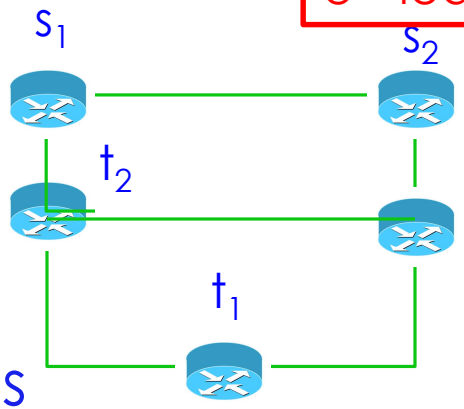
$\sigma + \text{load}^{\alpha}$



Warm-up Problem 3:

- Assume static power $\sigma = 0$
- Greedy Algorithm: route each request in such a way that the aggregate increase in power is minimized
  - How do you compute this?
- How would you approach analyzing this algorithm?

$\sigma + \text{load}^{\alpha}$



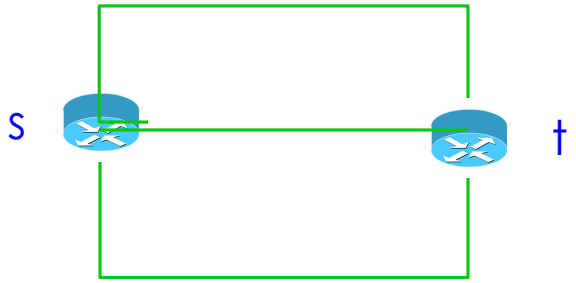
Recall Online Algorithm Design and Analysis Technique

- Design
  - Express the problem as a mathematical program
  - View the online problem as constraints of this primal program arriving online
  - Consider the online greedy algorithm that raises the primal variables to satisfy the newly arriving constraint so as to minimize the increase in the primal objective
- Analysis (Dual fitting)
  - Set the dual variable for the new constraint to be rate that costs are increasing for online at that time
  - Show that the dual cost isn't too much less than the online cost

Warm-up Problem 4:

$\sigma + \text{load}^{\alpha}$

- Assume static power $\sigma = \infty$
- Assume all $s_i = s$
- Assume all $t_i = t$

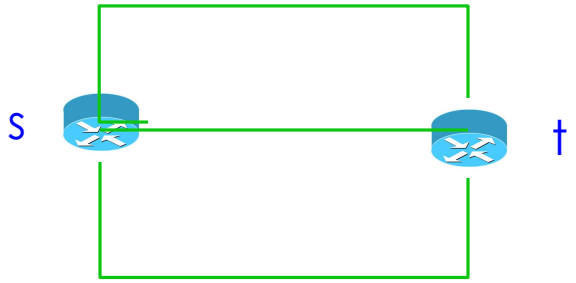


- Question: What is the optimal solution for this network?

Warm-up Problem 4:

$\sigma + \text{load}^{\alpha}$

- Assume static power $\sigma = \infty$
- Assume all $s_i = s$
- Assume all $t_i = t$



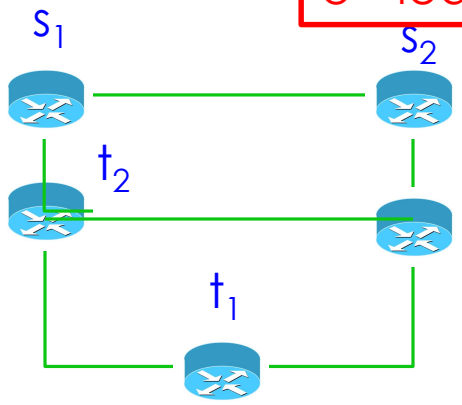
- Question: What is the optimal solution for this network?

- Answer: Use only 1 edge, and shutdown the rest

Warm-up Problem 5:

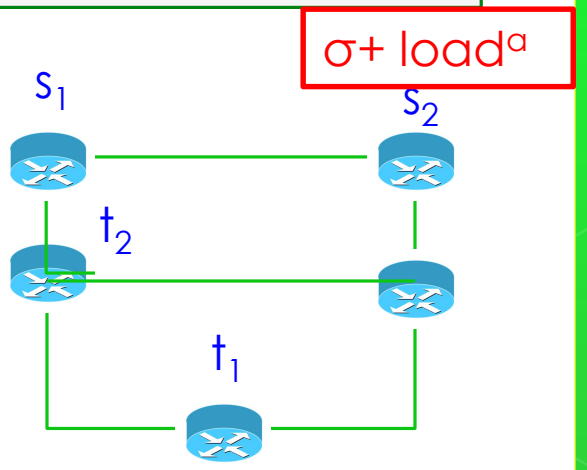
- Assume static power $\sigma = \infty$
- Question: What is the optimal solution for a general network?

$\sigma + \text{load}^{\alpha}$



Warm-up Problem 5:

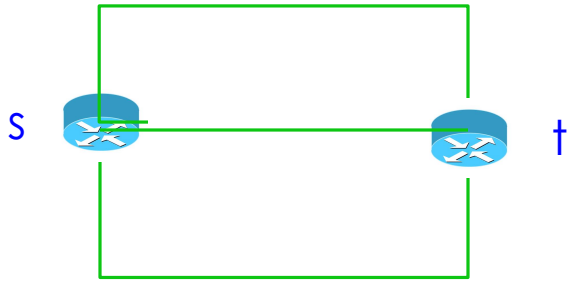
- Assume static power $\sigma = \infty$
- Question: What is the optimal solution for a general network?
- Answer: Buy a Steiner tree with a minimum number of edges
  - NP-hard but O(1)-approximation is possible in polynomial time



Warm-up Problem 6:

$\sigma + \text{load}^{\alpha}$

- k requests
- Many parallel edges
- Assume all $s_i = s$
- Assume all $t_i = t$

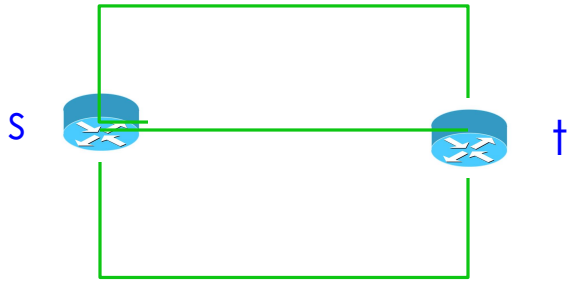


- Question: What is the optimal solution for this network?

Warm-up Problem 6:

$\sigma + \text{load}^{\alpha}$

- k requests
- Many parallel edges
- Assume all $s_i = s$
- Assume all $t_i = t$



- Question: What is the optimal solution for this network?

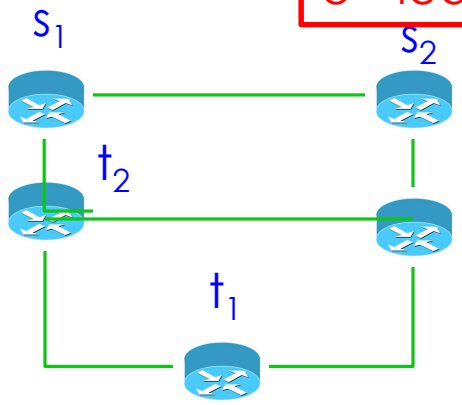
- Answer: Have q paths on each of k/q edges, and shutdown the rest of the edges
  - $q = \sigma^{1/\alpha}$ is load at which dynamic power = static power

Warm-up Problem 7:

- Assume static power $\sigma = 1$ =flow per request pair

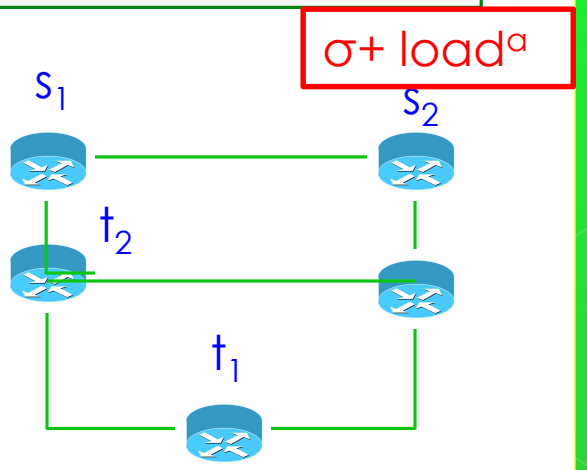
- Question: How to get an O(1)-approximate routing in poly-time?

$\sigma + \text{load}^{\alpha}$



Warm-up Problem 7:

- Assume static power $\sigma = 1$ =flow per request pair



- Question: How to get an O(1)-approximate routing in poly-time?

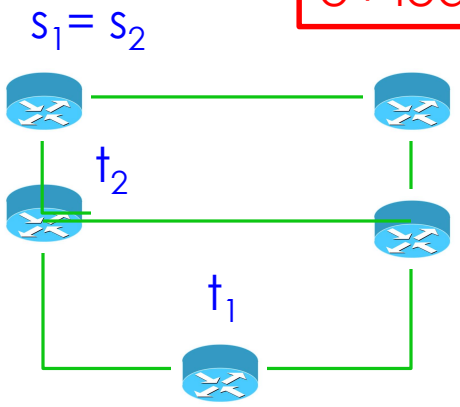
- Answer: Greedy unsplittable routing
  - Dual fitting analysis for greedy splittable routing can be adapted to unsplittable routing

Warm-up Problem 8:

- All sinks are identical. Multicast routing

- Question: How to get an O(1)-approximate offline poly-time algorithm?
- What edges can you afford for sure to power on?

σ+ load$^{\alpha}$

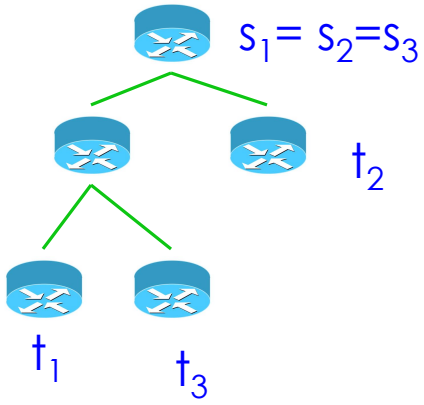


Warm-up Problem 8:

- All sinks are identical. Multicast routing

- Offline Algorithm:
  - power on a Steiner tree
  - Now what?

σ+ load$^{\alpha}$

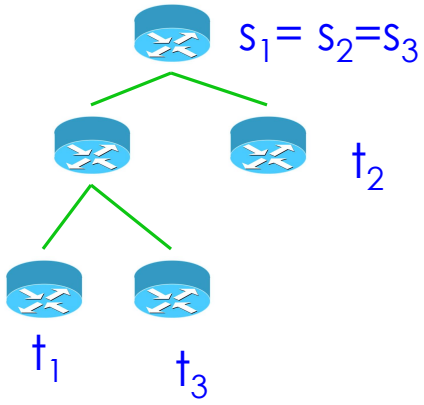


Warm-up Problem 8:

- All sinks are identical. Multicast routing

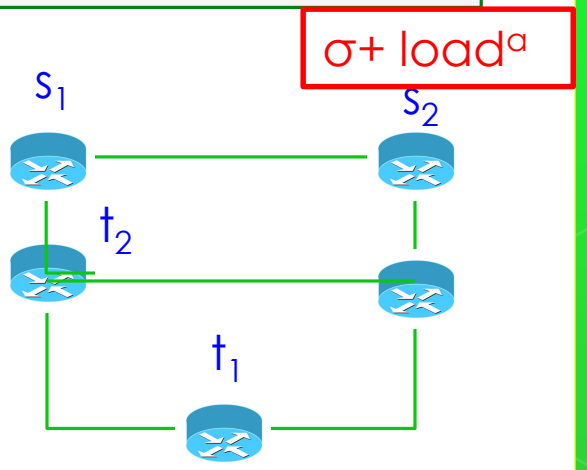
- Offline Algorithm:
  - power on a Steiner tree
  - Aggregate demands from sources into groups of size q = $\sigma^{1/\alpha}$ by routing on the Steiner tree
  - Use greedy to route from aggregated sources to common sink

σ+ load$^{\alpha}$



Real Problem:

- Question: What is the optimal solution for a general network?



Roadmap

- Many warmup problems to build intuition
- Offline algorithm almost full analysis
- Online algorithm and hint at analysis
- Merest of hints on difficulty of node based routing

Offline Algorithm

- What edges can we afford to power on for sure?

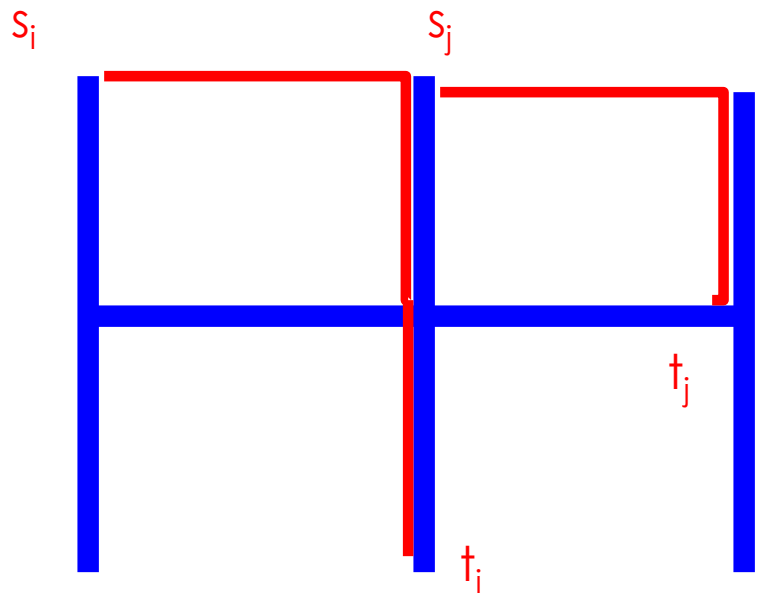
Offline Algorithm

- Power-on a Steiner forest to guarantee minimal connectivity
- In multicast routing, we finished by aggregating on the tree, and then using greedy routing. Why won't that work here?
- Still the Steiner tree may not have enough capacity. How might we decide what additional capacity to add?

Offline Algorithm

- Power-on a Steiner forest to guarantee minimal connectivity
- Hallucination:
  - Sparsification: With probability $\Theta(\log k)/q$ each request pair hallucinates its demand is q
  - Greedy algorithm is used to unsplittably route this "hallucinated flow"
  - The "hallucinated" edges used to route hallucinated flow are powered on
- Greedy algorithm is used to route flow on the "powered on" edges

The Powered-on Edges in the Hallucination Algorithm



- Steiner Forrest
  - Insure connectivity
  - Intuitively, high capacity highways
- Hallucinated edges
  - Greedy paths between randomly selected pairs
  - Intuitively, lower capacity

Offline Analysis

- Theorem: The algorithm is $O(\log^{\alpha} k)$-approximate
  - Proof: Both the static power and the dynamic power are $O(\log^{\alpha} k)$*Opt

Offline Analysis: Static Power

- Lemma: The static power for the Steiner forest edges is O(1)*Opt
  - Proof?

- Lemma: The static power for the hallucinated edges is $O(\log^{\alpha} k) * Opt$
  - Proof?

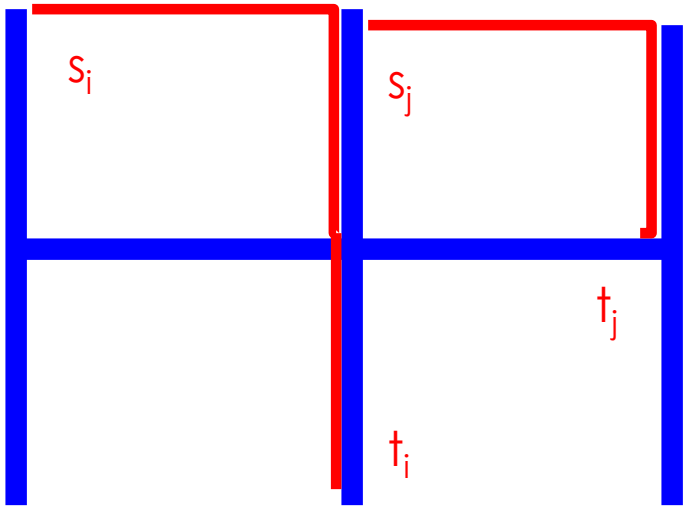
Offline Analysis: Static Power

- Lemma: The static power for the Steiner forest edges is O(1)*Opt
  - Proof: Property of Steiner tree approximation algorithm

- Lemma: The static power for the hallucinated edges is $O(\log^a k) * Opt$
  - Proof:
    - The greedy algorithm that is used to route hallucinated flow is O(1)-competitive
    - Sparsification doesn't increase the expected cost for Opt on any edge by more than $O(\log^a k)$ factor on

Offline Analysis: Dynamic Power: via Congestion

- Capacification: Each Steiner edge is given capacity q log k and each hallucinated edge is given capacity q



- Lemma: This is enough capacity to route all the flow

- Corollary: The dynamic power used by the algorithm is $O(\log^a k)$ *Opt
- Proof?

Offline Analysis: Dynamic Power: via Congestion

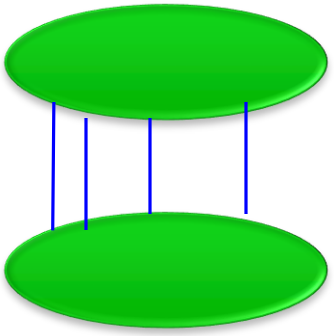
- Defn: Sparsity of a cut Q = capacity of edges in Q / demand across Q
- Main Lemma: The sparsity of every cut is $\Omega(\log k)$
- Corollary: There is a O(1)-congestion routing
  - Proof: flow-cut gap for multi-commodity flow. The one big hammer

Offline Analysis: Dynamic Power: via Congestion

- Lemma: The sparsity of every cut is $\Omega(\log k)$
  - Proof:
    - Tree edges have enough capacity if there is low demand across a cut.
    - Sublemma: Hallucinated edges have enough capacity in expectation if there is high demand across a cut. Nontrivial union bound.

Offline Analysis: Dynamic Power: via Congestion

- Sublemma: Hallucinated edges have enough capacity in expectation if there is high demand across a cut.

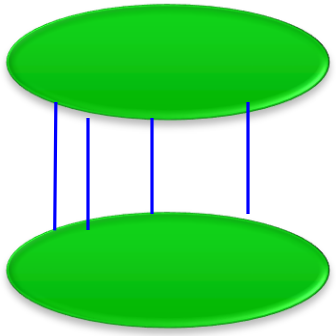


j tree edges across cut

- How many cuts with j Steiner tree edges across the cut?
  - WLOG, Steiner tree has at most 4k nodes

Offline Analysis: Dynamic Power: via Congestion

- Sublemma: Hallucinated edges have enough capacity in expectation if there is high demand across a cut.
- At most $C(4k, j)\ 2^k \approx k^j$ such cuts



j tree edges across cut

- Assume $r >> jq$ request pairs across the cut
- Then the expected number of hallucinated pairs across cut = $r \log k / q$
- And the probability this is << expectation
  - $< \exp(C r \log k / q)$ (Chernoff bound)
  - $< \exp(C j \log k)$ (since r is large)
  - $<< k^j$ (Since constant C is large)

Roadmap

- Many warmup problems to build intuition
- Offline algorithm almost full analysis
- Online algorithm and hint at analysis
- Merest of hints on difficulty of node based routing

- Offline Algorithm
  - Power-on a Steiner forest to guarantee minimal connectivity
  - Hallucination:
    - Sparsification: With probability $\Theta(\log k)/q$ each request pair hallucinates its demand is q
    - Greedy algorithm is used to unsplittably route this "hallucinated flow"
    - The "hallucinated" edges used to route hallucinated flow are powered on
  - Greedy algorithm is used to route flow on the "powered on" edges
  - Online algorithm?

Online Algorithm

- For each new request (s_j, t_j)
  - Turn on a minimal number of routers so that sj and tj are connected (Greedy Steiner Forrest)
  - Hallucination:
    - Sparsification: With probability $\Theta(\log k)/q$ request j hallucinates its demand is q
    - Greedy algorithm is used to unsplittably route this "hallucinated flow"
    - The "hallucinated" edges used to route hallucinated flow are powered on
  - Greedy algorithm is used to route flow on the "powered on" edges

What Has to Change for Analysis of Online Algorithm?

- Offline Analysis
  - Static power of Steiner tree is O(1) Opt
  - Static power of hallucinated edges is O(polylog) Opt
  - Dynamic power is O(polylog) Opt since there is a low congestion routing
- Hint: One doesn't change, one changes in a minor way, one changes in a major way

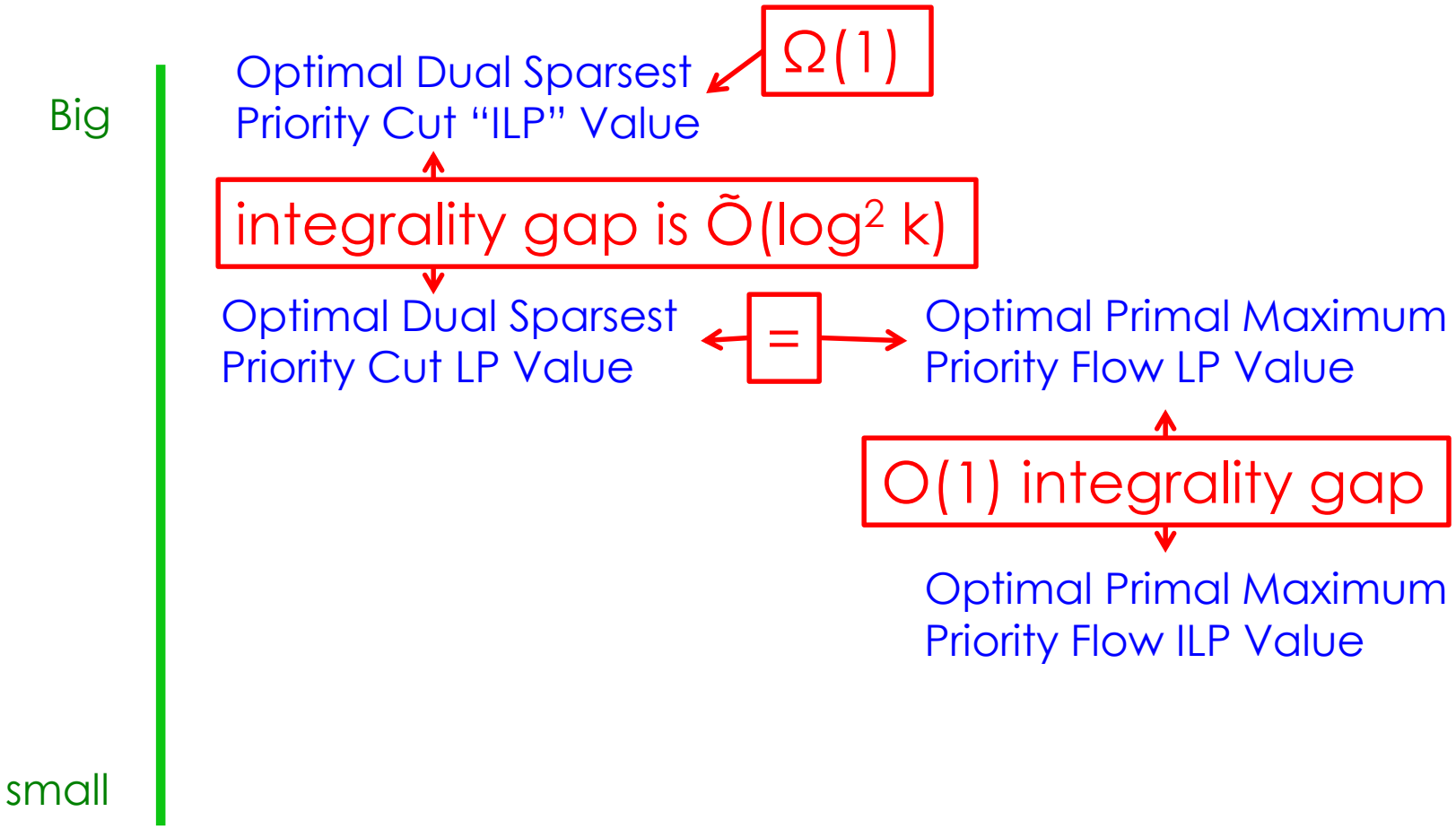
What Has to Change for Analysis of Online Algorithm?

- Offline Analysis
  - Static power of Steiner tree is O(polylog) Opt
    - Well known analysis for online Steiner tree
  - Static power of hallucinated edges is O(polylog) Opt
  - Dynamic power is O(polylog) Opt since there is a low congestion priority routing
    - Priority routing = each path only routes along edges powered on by the online algorithm by the time that request arrived

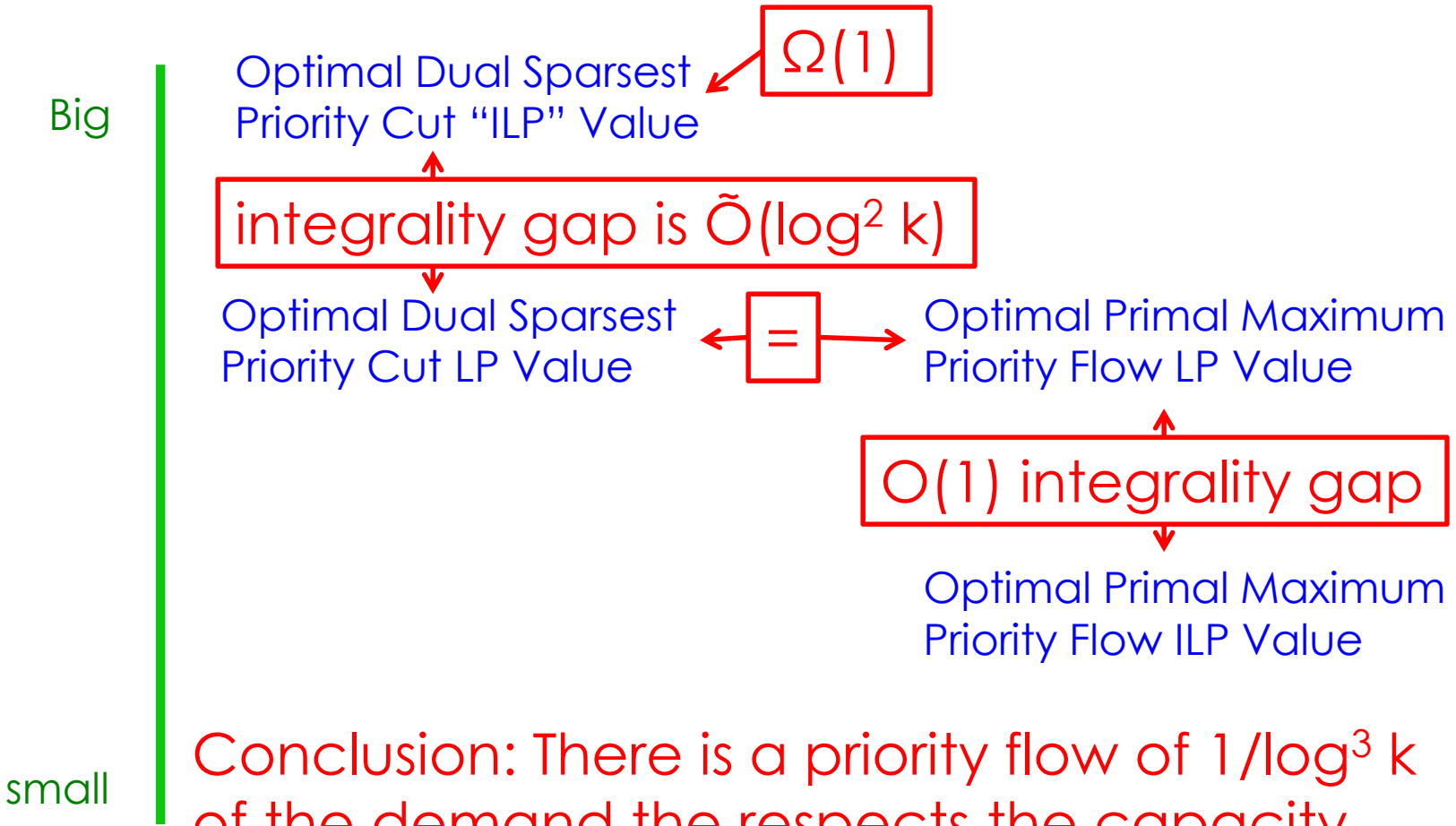
Online Analysis: Dynamic Power

- Lemma: The greedy algorithm is O(1)-approximate for dynamic power for priority routings
  - Proof: Same dual fitting analysis as nonpriority routing
- Essentially all the technical difficulty: Need to prove that Opt can't greatly profit from powering on edges before online does
  - To mimic the analysis in the offline case, we need to show that there is a low congestion priority routing

Strategy to Show Low Congestion Priority Routing



Strategy to Show Low Congestion Priority Routing



Conclusion: There is a priority flow of $1/\log^3 k$ of the demand the respects the capacity constraints.

Maximum Priority Multicommodity Flow LP

$$\max \gamma \qquad \text{(PriorityFlowLP)}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq \gamma \qquad \forall i \in [k] \quad (2)$$

$$\sum_{p | e \in p} f(p) \leq 1 \qquad \forall e \in G(k) \quad (3)$$

$$0 \leq f(p) \leq 1 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i \quad (4)$$

- f(p) = flow routed on path p
- Priority component: $\mathcal{P}_i$ = priority (s_i, t_i) paths
- Objective: Fractionally route as large of a fraction of each unit demand as possible
- wlog capacities are 1 by duplicating edges

Taking the Dual

Dual Variables

η_i

d_e

$$\max \gamma \qquad \text{(PriorityFlowLP)}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq \gamma \qquad \forall i \in [k] \quad (2)$$

$$\sum_{p | e \in p} f(p) \leq 1 \qquad \forall e \in G(k) \quad (3)$$

$$0 \leq f(p) \leq 1 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i \quad (4)$$

Primal variables

γ

$f(p)$

Dual LP: Sparsest Priority Cut

$$\min \sum_{e \in G(k)} d_e \qquad \text{(SparsestPriorityCutLP)}$$

$$\text{s.t.} \sum_{i=1}^{k} \eta_i \geq 1 \qquad (5)$$

$$\sum_{e \in p} d_e \geq \eta_i \qquad \forall p \in \mathcal{P}_i \, \forall i \in [k] \qquad (6)$$

$$d_e \geq 0 \qquad \forall e \in G(k) \qquad (7)$$

$$\eta_i \geq 0 \qquad \forall i \in [k] \qquad (8)$$

- Defn: (s_i, t_i) are priority cut by edges Q if removing Q makes them disconnected at time i
- Defn: Priority sparsity of cut Q = | Q | / (number of requests priority separated by Q

- "ILP" = sparsest priority cut problem
  - d_e = 1/(number of priority cut requests) if e is in Q, and 0 otherwise
  - η_i = 1/(number of priority cut requests) if request i is cut, and 0 otherwise

Roadmap

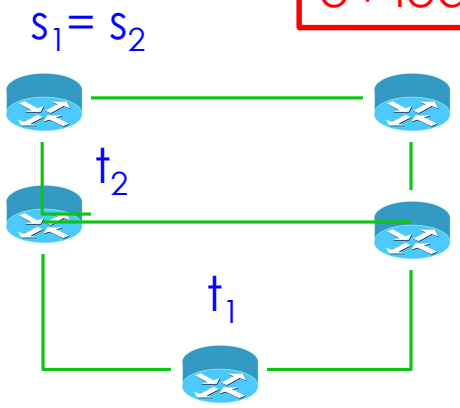
- Many warmup problems to build intuition
- Offline algorithm almost full analysis
- Online algorithm and hint at analysis
- Merest of hints on difficulty of node based routing

Vertex Version

- Assume now the energy used by a vertex with load L is:
  - 0 if L=0
  - $\sigma + L^{\alpha}$ if L > 0
- Consider single sink version

- What was the algorithm for a single sink when vertices were free?

σ+ load$^{\alpha}$



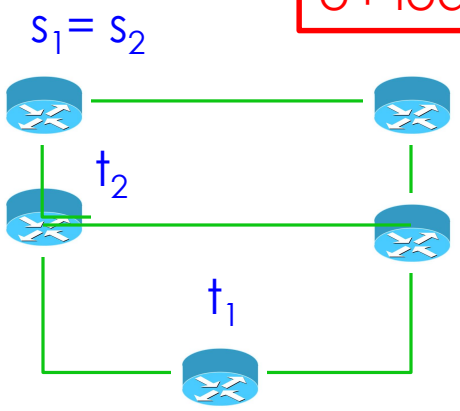
Vertex Version

- Assume now the energy used by a vertex with load L is:
  - 0 if L=0
  - $\sigma + L^{\alpha}$ if L > 0
- Consider single sink version

- Algorithm:
  - Power-on a minimum Steiner tree
  - Aggregate sources into groups of size $q=\sigma^{1/\alpha}$ in tree
  - Route greedily from aggregated locations

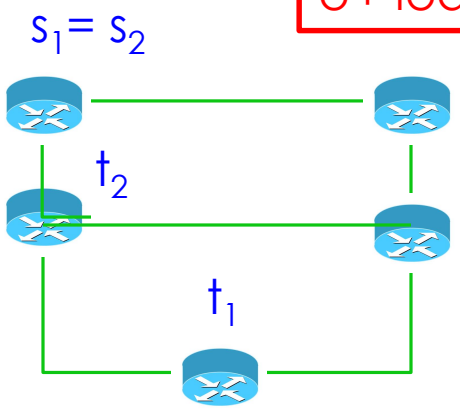
- What doesn't work in vertex version?

$\sigma + load^{\alpha}$

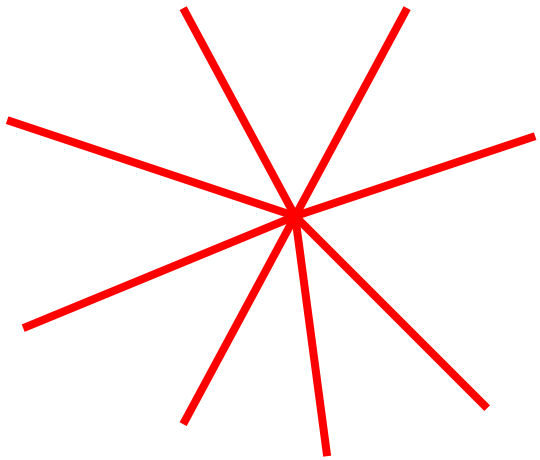


Vertex Version

- Assume now the energy used by a vertex with load L is:
  - 0 if L=0
  - $\sigma + L^{\alpha}$ if L > 0
- Consider single sink version



- Algorithm:
  - Power-on a minimum Steiner tree
  - Aggregate sources into groups of size $q=\sigma^{1/\alpha}$ in tree
  - Route greedily from aggregated locations
- Low cost aggregation isn't possible if Steiner tree was



Steiner Tree

First Step to Getting an Algorithm

- Theorem: There is a Steiner tree where this approach will work.
- Proof: Consider how optimal routes flow

After Many Big Hammers …

- Theorem [STOC 2014]: There is a poly-time poly-log-approximate algorithm for energy efficient routing if the power management happens at the nodes

Many Open Questions: Roughly Ranked

1. For offline edge version get poly-log approximation where poly doesn't depend on a
2. Get an analysis that reasons about energy directly (not via congestion)
3. An analysis for the offline node version that is not ridiculously complicated
4. An online algorithm and analysis for the node version
5. Tighten-up priority sparsity lower bound and/or priority sparsest cut integrality gap bounds to get better bounds for online edge version

Covered Papers

- [WAOA2012] Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs: Online Primal-Dual for Non-linear Optimization with Applications to Speed Scaling. Workshop on Approximation and Online Algorithms, 2012: 173-186
- [MedAlg2012] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, Kirk Pruhs, Cliff Stein: Multicast Routing for Energy Minimization Using Speed Scaling. Mediterranean Conference on Algorithms 2012: 37-51
- [SODA2014] Antonios Antoniadis, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, Viswanath Nagarajan, Kirk Pruhs, Cliff Stein, Hallucination Helps: Energy Efficient Virtual Circuit Routing, ACM-SIAM Symposium on Discrete Algorithms 2014.
- [STOC2014] Ravishankar Krishnaswamy, Viswanath Nagarajan, Kirk Pruhs, Cliff Stein, Cluster before you hallucinate: approximating node-capacitated network design and energy efficient routing. STOC 2014: 734-743

Thanks to all my collaborators!
Thanks for listening!