

Intradomain Traffic Engineering

slides by Jennifer Rexford

Do IP networks manage themselves?

- ❑ In some sense, yes:
 - TCP senders send less traffic during congestion
 - Routing protocols adapt to topology changes
- ❑ But, does the network run *efficiently*?
 - Congested link when idle paths exist?
 - High-delay path when a low-delay path exists?
- ❑ How should routing adapt to the traffic?
 - Avoiding congested links in the network
 - Satisfying application requirements (e.g., delay)
- ❑ ... essential questions of traffic engineering

Traffic engineering

- ❑ What is traffic engineering?
 - Control and optimization of routing, to steer traffic through the network in the most effective way
- ❑ Two fundamental approaches to adaptation
 - Adaptive routing protocols
 - Distribute traffic and performance measurements
 - Compute paths based on load, and requirements
 - Adaptive network-management system
 - Collect measurements of traffic and topology
 - Optimize the setting of the “static” parameters
- ❑ Big debates still today about the right answer

Outline: Three alternatives

- ❑ Load-sensitive routing at *packet* level
 - Routers receive feedback on load and delay
 - Routers re-compute their forwarding tables
 - Fundamental problems with oscillation
- ❑ Load-sensitive routing at *circuit* level
 - Routers receive feedback on load and delay
 - Router compute a path for the next circuit
 - Less oscillation, as long as circuits last for a while
- ❑ Traffic engineering as a *management problem*
 - Routers compute paths based on “static” values
 - Network management system sets the parameters
 - Acting on network-wide view of traffic and topology

Load-sensitive routing protocols: Pros and Cons

□ Advantages

- Efficient use of network resources
- Satisfying the performance needs of end users
- Self-managing network takes care of itself

□ Disadvantages

- Higher overhead on the routers
- Long alternate paths consume extra resources
- Instability from reacting to out-of-date information

Packet-based load-sensitive routing

□ Packet-based routing

- Forward packets based on forwarding table

□ Load-sensitive

- Compute table entries based on load or delay

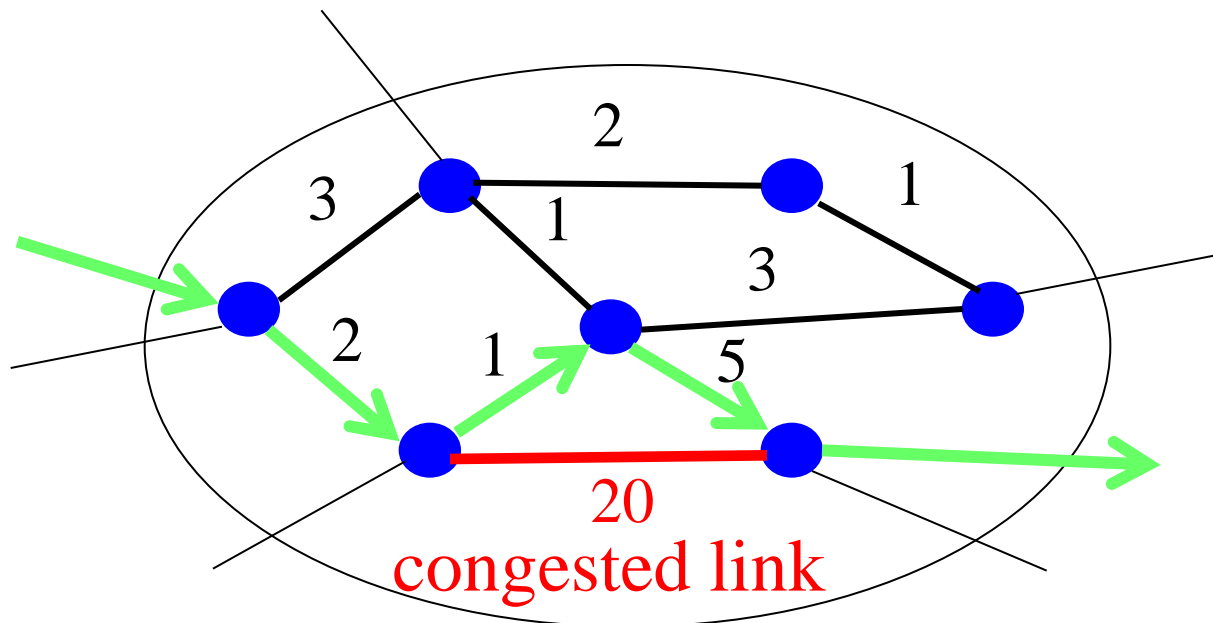
□ Questions

- What link metrics to use?
- How frequently to update the metrics?
- How to propagate the metrics?
- How to compute the paths based on metrics?

Original ARPANET algorithm (1969)

□ Routing algorithm

- Shortest-path routing based on link metrics
- Instantaneous queue length plus a constant
- Distributed shortest-path algorithm (Bellman-Ford)



Performance of original ARPANET algo

□ Light load

- Delay dominated by the constant part (transmission delay and propagation delay)

□ Medium load

- Queuing delay is no longer negligible
- Moderate traffic shifts to avoid congestion

□ Heavy load

- Very high metrics on congested links
- Busy links look bad to all of the routers
- All routers avoid the busy links
- Routers may send packets on longer paths

Second ARPANET algorithm (1979)

□ Averaging of the link metric over time

- Old: Instantaneous delay fluctuates a lot
- New: Averaging reduces the fluctuations

□ Link-state protocol

- Old: Distributed path computation leads to loops
- New: Better to flood metrics and have each router compute the shortest paths

□ Reduce frequency of updates

- Old: Sending updates on each change is too much
- New: Send updates if change passes a threshold

Problem of long alternate paths

❑ Picking alternate paths

- Long path chosen by one router consumes resource that other packets could have used
- Leads other routers to pick other alternate paths

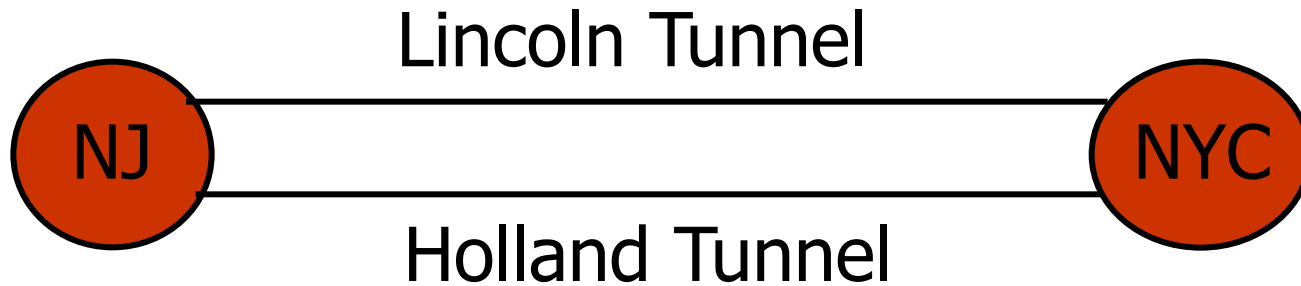
❑ Solution: Limit path length

- Bound the value of the link metric
- “This link is busy enough to go two extra hops”

❑ Extreme case

- Limit path selection to shortest paths
- Pick the least-loaded shortest path in the network

Problem of out-of-date information



“Backup at Lincoln” on radio triggers congestion at Holland

- ❑ Routers make decisions based on old information
 - Propagation delay in flooding link metrics
 - Thresholds applied to limit number of updates
- ❑ Old information leads to bad decisions
 - All routers avoid the congested links
 - ... leading to congestion on other links
 - ... and the whole things repeats

Avoiding oscillations from out-of-date info

- ❑ Send link metrics more often
 - But, leads to higher overhead
 - But, propagation delay is a fundamental limit
- ❑ Make the traffic last longer
 - Circuit switching: Phone network
 - Average phone call last 3 minutes
 - Plenty of time for feedback on link loads
 - Packet switching: Internet
 - Data packet is small (e.g., 1500 bytes or less)
 - But, feedback on link metrics also sent via packets
 - Better to make decisions on groups of packets

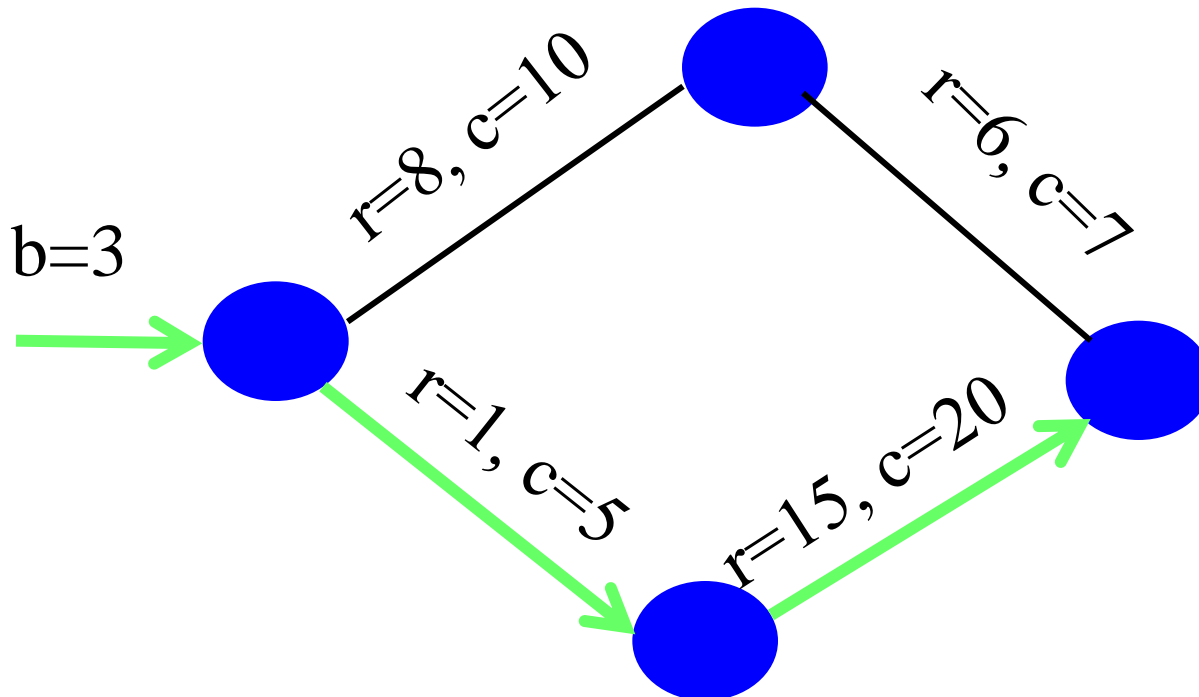
Quality-of-Service routing on circuits

Quality-of-Service routing with circuit switching

- ❑ Traffic performance requirement
 - Guaranteed bandwidth b per connection
- ❑ Link resource reservation
 - Reserved bandwidth r_i on link I
 - Capacity c_i on link i
- ❑ Signaling: Admission control on path P
 - Reserve bandwidth b on each link i on path P
 - Block: if $(r_i + b > c_i)$ then reject (or try again)
 - Accept: else $r_i = r_i + b$
- ❑ Routing: Ingress router selects the path

Source-directed QoS routing

- New connection with $b = 3$
 - Routing: Select path with available resources
 - Signaling: Reserve bandwidth along the path ($r = r + 3$)
 - Forwarding: Forward data packets along the selected path
 - Teardown: Free the link bandwidth ($r = r - 3$)



QoS routing: Path selection

□ Link-state advertisements

- Advertise available bandwidth ($c_i - r_i$) on link i
 - E.g., every T seconds, independent of changes
 - E.g., when metric changes beyond threshold
- Each router constructs view of topology

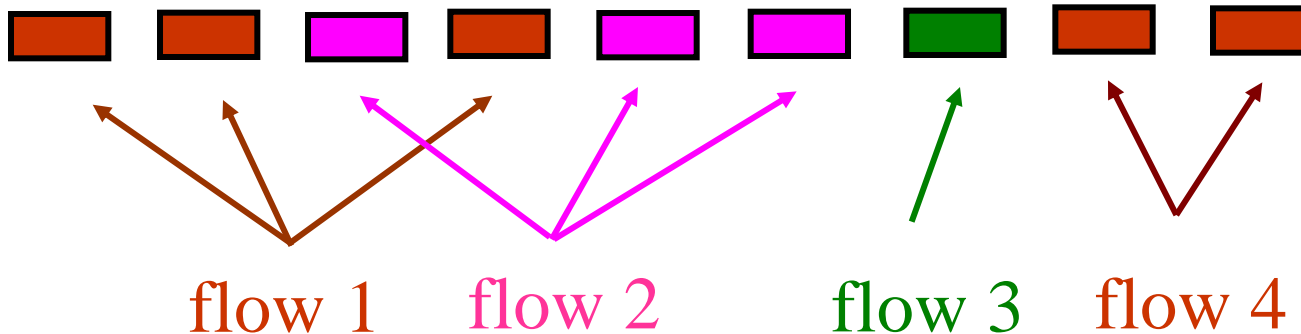
□ Path computation at each router

- E.g., Shortest widest path
 - Consider paths with largest value of $\min_i(c_i - r_i)$
 - Tie-break on smallest number of hops
- E.g., Widest shortest path
 - Consider only paths with minimum hops
 - Tie-break on largest value of $\min_i(c_i - r_i)$ over paths

How to get IP packets on to circuits?

- ❑ Who initiates the circuit?
 - End system application or operating system?
 - Edge router?
- ❑ Edge router can infer the need for a circuit
 - Match on packet header bits
 - E.g., source, destination, port numbers, etc.
 - Apply policy for picking bandwidth parameters
 - E.g., Web connections get 10 Kbps, video gets 2 Mbps
 - Trigger establishment of circuit for the traffic
 - Select path based on load and requirements
 - Signal creation of the circuit
 - Tear down circuit after an idle period

Grouping IP packets into flows



- ❑ Group packets with the “same” end points
 - Application level: single TCP connection
 - Host level: single source-destination pair
 - Subnet level: single source prefix and dest prefix
- ❑ Group packets that are close together in time
 - E.g., 60-sec spacing between consecutive packets

But, staleness can still be a problem...

❑ Link state updates

- High update rate leads to high overhead
- Low update rate leads to oscillation

❑ Connections are too short

- Average Web transfer is just 10 packets
- Requires high update rates to ensure stability

❑ Idea: QoS routing only for long transfers!

- Small fraction of transfers are very large
- ... and these few transfers carry a lot of traffic
- Forward most transfers on static routes
- ... and compute dynamic routes for long transfers

Identifying the long transfers

- ❑ A nice property of transfer sizes
 - Most transfers are short, but a few are *very* long
 - Distribution of transfer sizes is “heavy tailed”
- ❑ A nice property of heavy tails
 - After you see 10 packets, it is likely a long transfer
 - Even the *remainder* of the transfer is long
- ❑ Routing policy
 - Forward initial packets on the static default route
 - After seeing 10 packets, try to signal a circuit
 - Forward the remaining packets on the circuit
- ❑ Avoids oscillation even for small update rates
 - <http://www.cs.princeton.edu/~jrex/papers/sigcomm99.ps>

Ongoing work on QoS routing

□ Standards activity

- Traffic-engineering extensions to the conventional routing protocols (e.g., OSPF and IS-IS)
- Use of MPLS to establish the circuits over the links
- New work on Path Computation Elements that compute the load-sensitive routes for the routers

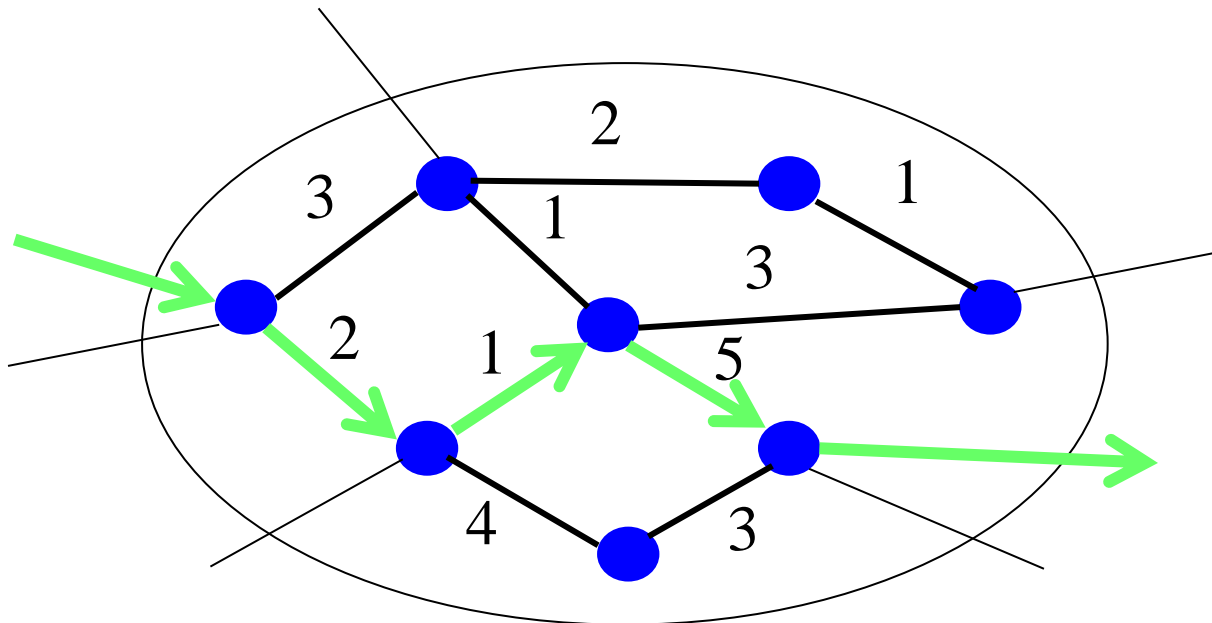
□ Research activity

- Avoid propagating dynamic link-state information
- Based decisions based on past success or failure
- Essentially inferring the state of the links

Traffic engineering as a network-management problem

Using traditional routing protocols

- Routers flood information to learn topology
 - Determine “next hop” to reach other routers...
 - Compute shortest paths based on link weights
- Link weights configured by network operator



Approaches for setting the link weights

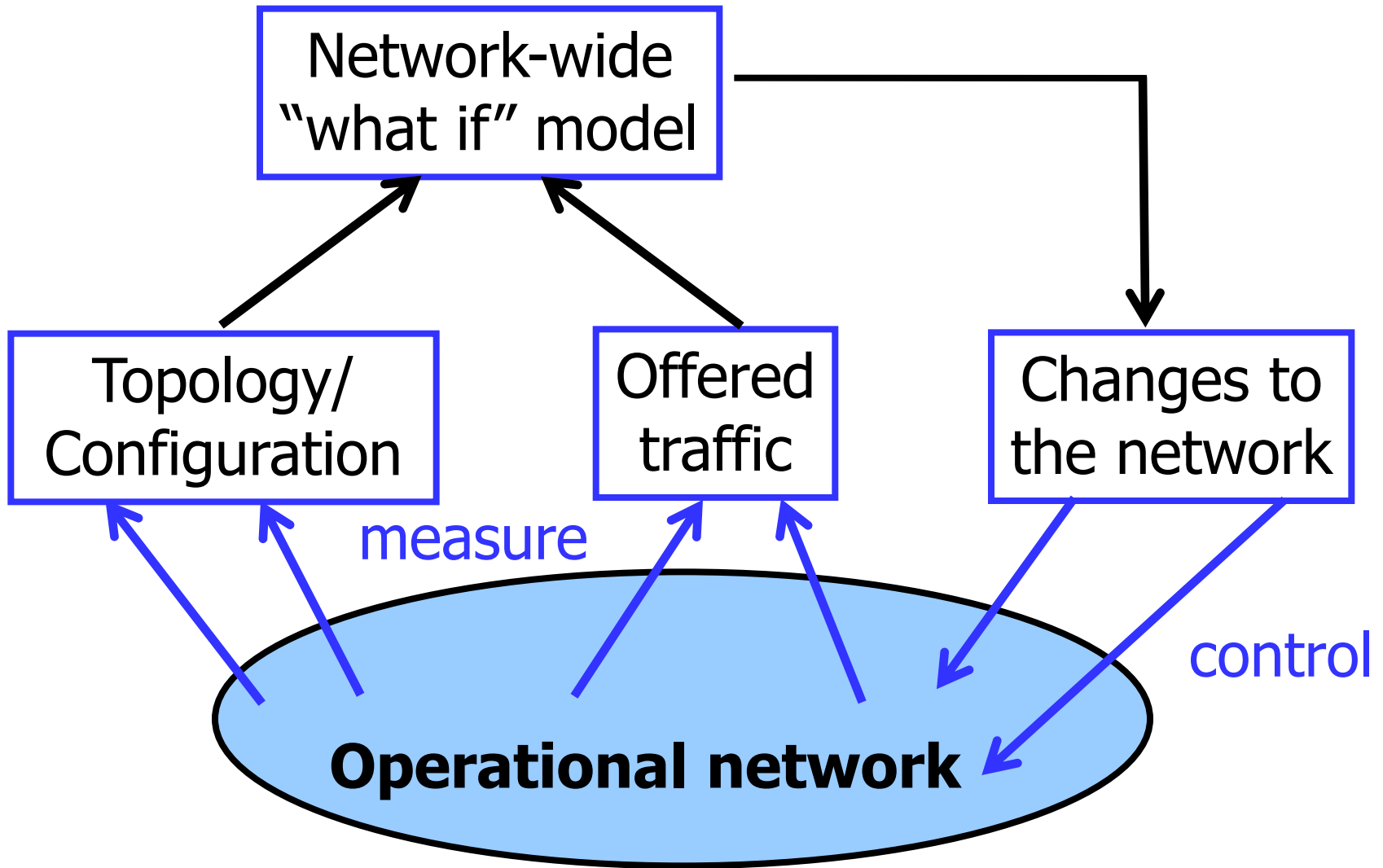
□ Conventional static heuristics

- Proportional to physical distance
 - Cross-country links have higher weights
 - Minimizes end-to-end propagation delay
- Inversely proportional to link capacity
 - Smaller weights for higher-bandwidth links
 - Attracts more traffic to links with more capacity

□ Tune the weights based on the offered traffic

- Network-wide optimization of the link weights
- Directly minimizes metrics like max link utilization

Measure, model, and control



Traffic engineering in ISP backbone

- ❑ Topology
 - Connectivity and capacity of routers and links
- ❑ Traffic matrix
 - Offered load between points in the network
- ❑ Link weights
 - Configurable parameters for routing protocol
- ❑ Performance objective
 - Balanced load, low latency, service level agreements ...
- ❑ Question: Given the *topology* and *traffic matrix*, which *link weights* should be used?

Key ingredients of the approach

□ Instrumentation

- Topology: monitoring of the routing protocols
- Traffic matrix: fine-grained traffic measurement

□ Network-wide models

- Representations of topology and traffic
- “What-if” models of shortest-path routing

□ Network optimization

- Efficient algorithms to find good configurations
- Operational experience to identify key constraints

Formalizing the optimization problem

□ Input: graph $G(R,L)$

- R is the set of routers
- L is the set of unidirectional links
- c_l is the capacity of link l

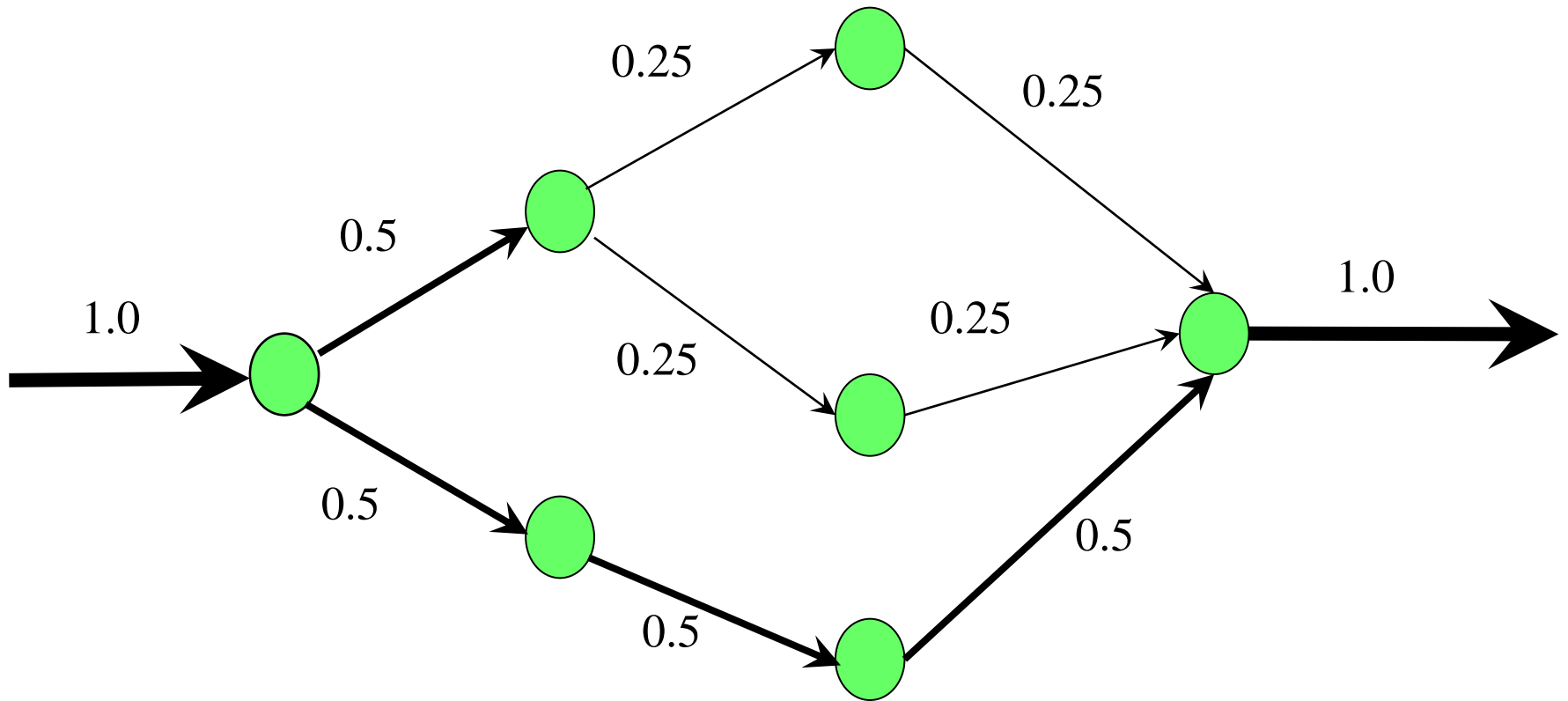
□ Input: traffic matrix

- $M_{i,j}$ is traffic load from router i to j

□ Output: setting of the link weights

- w_l is weight on unidirectional link l
- $P_{i,j,l}$ is fraction of traffic from i to j traversing link l

Multiple shortest paths with even splitting



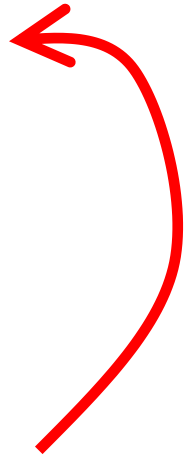
Values of $P_{i,j,l}$

Complexity of optimization problem

- NP-complete optimization problem
 - No efficient algorithm to find the link weights
 - Even for simple objective functions
- What are the implications?
 - Have to resort to *searching* through weight settings

Optimization based on local search

- Start with an initial setting of the link weights
 - E.g., same integer weight on every link
 - E.g., weights inversely proportional to capacity
 - E.g., existing weights in the operational network
- Compute the objective function
 - Compute the all-pairs shortest paths to get $P_{i,j,l}$
 - Apply the traffic matrix $M_{i,j}$ to get link loads u_l
 - Evaluate the objective function from the u_l/c_l
- Generate a new setting of the link weights



repeat

Incorporating operational realities

- ❑ Minimize number of changes to the network
 - Changing just 1 or 2 link weights is often enough
- ❑ Tolerate failure of network equipment
 - Weights settings usually remain good after failure
 - ... or can be fixed by changing one or two weights
- ❑ Limit dependence on measurement accuracy
 - Good weights remain good, despite random noise
- ❑ Limit frequency of changes to the weights
 - Joint optimization for day & night traffic matrices

Application to AT&T's backbone

□ Performance of the optimized weights

- Search finds a good solution within a few minutes
- Much better than link capacity or physical distance
- Competitive with multi-commodity flow solution

□ How AT&T changes the link weights

- Maintenance every night from midnight to 6am
- Predict effects of removing link(s) from network
- Reoptimize the link weights to avoid congestion
- Configure new weights before disabling equipment