Intra-Domain Routing and Traffic Engineering

- Review of Internet routing paradigm and routing algorithms/protocols
- Intra-domain routing design
  - topology design, convergence, stability, ...
- Traffic Engineering (TE)
  - MPLS and traffic engineering (will go over very briefly)
  - traffic engineering as network-wide optimization problem
  - TE through link weight assignments

Readings: do the required readings.
Internet Routing

• So far we have focused on data plane operations

• **Forwarding:** data plane
  - Directing a data packet to an outgoing link
  - Individual router using a forwarding table

• **Routing:** control plane
  - Computing the paths the packets will follow
  - Routers talking amongst themselves
  - Individual router creating a forwarding table
Routing

Routing protocol

Goal: determine “good” path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

• graph nodes are routers
• graph edges are physical links
  - link cost: delay, $ cost, or congestion level

• “good” path:
  - typically means minimum cost path
  - other def’s possible
Two-Tiered Internet Routing System

- **Interdomain routing:** between ASes
  - Routing policies based on *business relationships*
  - No common metrics, and limited cooperation
  - BGP: policy-based, path-vector routing protocol

- **Intrdomain routing:** within an AS
  - Shortest-path routing based on *link metrics*
  - Routers all managed by a single institution
  - OSPF and IS-IS: link-state routing protocol
  - RIP and EIGRP: distance-vector routing protocol
Shortest-Path Routing

• **Path-selection model**
  - Destination-based
  - Minimum hop count or sum of link weights
  - Dynamic vs. static link weights
Distance Vector Routing: Bellman-Ford

- Define distances at each node $x$
  - $d_x(y) =$ cost of least-cost path from $x$ to $y$
- Update distances based on neighbors
  - $d_x(y) = \min \{ c(x,v) + d_v(y) \}$ over all neighbors $v$

E.g., RIP and EIGRP
Link-State Routing: Dijkstra’s Algorithm

- Each router keeps track of its incident links
  - Link cost, and whether the link is up or down
- Each router broadcasts the link state
  - To give every router a complete view of the graph
- Each router runs Dijkstra’s algorithm
  - To compute shortest paths and forwarding table

E.g., OSPF and IS-IS
**OSPF (Open Shortest Path First)**

- **Link State Protocol**
  - link costs between 0 and 65,535
  - Cisco recommendation - link cost = 1/(link capacity)
  - rapid, loop-free convergence, scales well
  - topology map at each node, route computation using Dijkstra’s algorithm
  - OSPF advertisement carries one entry per neighbor router, advertisements flooded to entire Autonomous System

- multiple equal-cost paths allowed: flow equally split on all outgoing links belonging to shortest paths

- IS – IS (intermediate system-intermediate system) similar
# Routing Protocols

<table>
<thead>
<tr>
<th>Dissemination</th>
<th>Link State</th>
<th>Distance Vector</th>
<th>Path Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flood link state advertisements to all routers</td>
<td>Update distances from neighbors’ distances</td>
<td>Update paths based on neighbors’ paths</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Dijsktra’s shortest path</td>
<td>Bellman-Ford shortest path</td>
<td>Local policy to rank paths</td>
</tr>
<tr>
<td>Converge</td>
<td>Fast due to flooding</td>
<td>Slow, due to count-to-infinity</td>
<td>Slow, due to path exploration</td>
</tr>
<tr>
<td>Protocols</td>
<td>OSPF, IS-IS</td>
<td>RIP, EIGRP</td>
<td>BGP</td>
</tr>
</tbody>
</table>

**CSci5221: Intra-Domain Routing and TE**
Intra-Domain Routing Today

- Link-state routing with static link weights
  - Static weights: avoid stability problems
  - Link state: faster reaction to topology changes

- Most common protocols in backbones
  - OSPF: Open Shortest Path First
  - IS-IS: Intermediate System-Intermediate System

- Some use of distance vector in enterprises
  - RIP: Routing Information Protocol
  - EIGRP: Enhanced Interior Gateway Routing Protocol

- Growing use of Multi-Protocol Label Switching
What do Operators Worry About?

• **Topology design**
  - Small propagation delay and low congestion
  - Ability to tolerate node and link failures

• **Convergence delay (will discuss more later)**
  - Limiting the disruptions during topology changes
  - E.g., by trying to achieve faster convergence

• **Scalable routing designs**
  - Avoiding excessive protocol overhead
  - E.g., by introducing hierarchy in routing

• **Traffic engineering (focus today)**
  - Limiting propagation delay and congestion
  - E.g., by carefully tuning the “static” link weights
Topology Design: Intra-AS Topology

- **Node**: router
- **Edge**: link

Hub-and-spoke

Backbone
ISP Topology Design: Points-of-Presence (PoPs)

- **Inter-PoP links**
  - Long distances
  - High bandwidth

- **Intra-PoP links**
  - Short cables between racks or floors
  - Aggregated bandwidth

- **Links to other networks**
  - Wide range of media and bandwidth
Topology Design: Abilene Internet2 Backbone
Convergence: Detecting Topology Changes

• Beaconing
  - Periodic “hello” messages in both directions
  - Detect a failure after a few missed “hellos”

• Performance trade-offs
  - Detection speed
  - Overhead on link bandwidth and CPU
  - Likelihood of false detection
Convergence: Transient Disruptions

• Inconsistent link-state database
  - Some routers know about failure before others
  - The shortest paths are no longer consistent
  - Can cause transient forwarding loops
Convergence: Delay for Converging

• Sources of convergence delay
  - Detection latency
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table

• Performance during convergence period
  - Lost packets due to blackholes and TTL expiry
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination

• Very bad for VoIP, online gaming, and video
Convergence: Reducing Convergence Delay

• Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures

• Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority

• Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra algorithm

• Faster forwarding-table update
  - Data structures supporting incremental updates
Scalability: Overhead of Link-State Protocols

- Protocol overhead depends on the topology
  - Bandwidth: flooding of link state advertisements
  - Memory: storing the link-state database
  - Processing: computing the shortest paths
Scalability: Improving the Scaling Properties

• Dijkstra’s shortest-path algorithm
  - Simplest version: $O(N^2)$, where $N$ is # of nodes
  - Better algorithms: $O(L \times \log(N))$, where $L$ is # links
  - Incremental algorithms: great for small changes

• Timers to pace operations
  - Minimum time between LSAs for the same link
  - Minimum time between path computations

• More resources on the routers
  - Routers with more CPU and memory
Scalability: Introducing Hierarchy Through Areas

- Divide network into regions
  - Backbone (area 0) and non-backbone areas
  - Each area has its own link-state database
  - Advertise only path distances at area boundaries
Traffic Engineering

- **Goal:** configure routes to meet traffic demands
  - balanced load, low latency, service agreements
- operates at coarse timescales
  - Not to adapt to short-term sudden traffic changes
  - May take potential failures into consideration
- **Input to traffic engineering:**
  - **Topology:** connectivity & capacity of routers & links
  - **Traffic matrix:** offered load between points in the network
- **Traffic Engineering:** network-wide optimization
  - Subject to protocol mechanisms, configurable parameters and other practical constraints, ...
Traffic Engineering under Shortest Path Routing: Tuning Link Weights

- Problem: congestion along the blue path
  - Second or third link on the path is overloaded
- Solution: move some traffic to bottom path
  - E.g., by decreasing the weight of the second link
Limitations of Conventional Intra-Domain Routing

- **Overhead of hop-by-hop forwarding**
  - Large routing tables and expensive look-ups
- **Paths depend only on the destination**
  - Rather than differentiating by source or class
- **Only the shortest path(s) are used**
  - Even if a longer path has enough resources
- **Transient disruptions during convergence**
  - Cannot easily prepare in advance for changes
- **Limited control over paths after failure**
  - Depends on the link weights and remaining graph
Multi-Protocol Label Switching (MPLS)

• Motivating applications
  - Small routing tables and fast look-ups
  - Virtual Private Networks
  - Traffic engineering
  - Path protection and fast reroute

• Key ideas of MPLS
  - Label-switched path spans group of routers
  - Explicit path set-up, including backup paths
  - Flexible mapping of data traffic to paths

• Label Switching
  - MPLS header includes a label
  - Label switching between MPLS-capable routers
MPLS: Forwarding Based on Labels

- Hybrid of packet and virtual circuit switching
  - Logical circuit between a source and destination
  - Packets with different labels multiplex on a link
- Basic idea of label-based forwarding
  - Packet: fixed length label in the header
  - Switch: mapping label to an outgoing link
MPLS Concept: Route at Edge, Switch in Core
**MPLS: Pushing, Swapping, and Popping**

- **Pushing**: add the initial “in” label
- **Swapping**: map “in” label to “out” label
- **Popping**: remove the “out” label
MPLS Header

- IP packet *encapsulated* in MPLS header and sent down LSP

```
32-bit MPLS Header
```

- IP packet *restored* at end of LSP by egress router
  - TTL adjusted by default
MPLS Header

- **label**
  - used to match packet to LSP
- **experimental bits**
  - carries packet queuing priority (CoS)
- **stacking bit**: can build “stacks” of labels
  - goal: nested tunnels!
- **time to live**
  - copied from IP TTL
MPLS Terminology

- **LDP**: Label Distribution Protocol
- **LSP**: Label Switched Path
- **FEC**: Forwarding Equivalence Class
- **LSR**: Label Switching Router
- **LER**: Label Edge Router (Useful term not in standards)
- **MPLS** “multi-protocol” both in terms of protocols it supports ABOVE and BELOW in protocol stack!
Example
Label Distribution Protocol (LDP)

- label distribution always done from downstream to upstream

- **downstream-unsolicited**: new route => send new label

- **downstream-on-demand**: upstream LSR asks for label
Forwarding Equivalence Class (FEC)

- **FEC** – rule for grouping IP packets
  - Packets that should be treated the same way,
    - e.g., forwarded over same path, with same forwarding treatment
  - Identified just once, at the edge of the network

- **Example FECs**
  - destination prefix
    - longest-prefix match in forwarding table at entry point
    - useful for conventional destination-based forwarding
  - same src and dst addresses
  - 5-tuple flows (src/dst addr, src/dst port, and protocol)
    - five-tuple match at entry point
    - useful for fine-grain control over the traffic
  - QoS class

A label is just a locally-significant identifier for a FEC
Regular IP Forwarding

IP destination address unchanged in packet header!
MPLS Label Distribution

<table>
<thead>
<tr>
<th>Intf In</th>
<th>Label In</th>
<th>Dest</th>
<th>Intf Out</th>
<th>Label Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.50</td>
<td>47.1</td>
<td>1</td>
<td>0.40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intf In</th>
<th>Label In</th>
<th>Dest</th>
<th>Intf Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.40</td>
<td>47.1</td>
<td>1</td>
</tr>
</tbody>
</table>
Label Switched Path (LSP)
MPLS Forwarding: Example

- IP packet destined to 134.112.1.5/32 arrives to SF
- San Francisco has route for 134.112/16
  - next hop is LSP to New York
MPLS Forwarding Example

- San Francisco pre-pends MPLS header onto IP packet, sends packet to first transit router on path

![MPLS Forwarding Example Diagram]
MPLS Forwarding Example

- because packet arrived to Santa Fe with MPLS header, Santa Fe forwards it using MPLS forwarding table
MPLS Forwarding Example

- packet arrives from penultimate router with label 0
- egress router sees label 0, strips MPLS header
- egress router performs standard IP forwarding
Status of MPLS

• Deployed in practice
  - Small control and data plane overhead in core
  - Virtual Private Networks
  - Traffic engineering and fast reroute

• Challenges
  - Protocol complexity
  - Configuration complexity
  - Difficulty of collecting measurement data

• Continuing evolution
  - Standards
  - Operational practices and tools
Traffic Engineering Framework

• Basic Requirements
  - Knowledge of Topology
    • Connectivity and capacities of routers/links of a network
  - Traffic Matrix
    • (average) traffic demand between different ingress/egress points of a network

• Instrumentation
  - Topology: monitoring of the routing protocols
  - Traffic matrix: “fine-grained” traffic measurement and inference, for example, via
    • SNMP
    • edge measurements + routing tables
    • network tomography
    • packet sampling
Traffic Engineering Framework (cont’d)

• Traffic Engineering as Network-Wide Optimization

• Network-wide models
  - Network topology:
    
    \[
    \text{graph } (V,E), \quad c_{ij} \text{: capacity of link } (i,j)
    \]
  - Traffic Matrix:
    
    \[K\text{ set of (ingress/egress) source-destination pair demands}\]
    
    \[k \in K, \quad d_k \text{- demand, } s_k \text{- source, } t_k \text{- destination}\]

• Optimization criteria, e.g.,
  - minimize maximum utilization,
  - minimize sum of link utilization
  - keep utilizations below 60%
Traffic Engineering as a Global Optimization Problem

- topology $G = (V, E)$
- $c_{ij}$ - capacity of link $(i, j) \in E$
- $K$ - set of origin-destination flows (demands)
  - $k \in K$, $d_k$ - demand, $s_k$ - source, $t_k$ - destination
- $F_{ij}^k$: traffic load of O-D flow $k$ routed on link $(i, j)$
  \[ F_{ij} := \sum_{k} F_{ij}^k : \text{total load (of all demands) on link } (i, j) \]
- Optimization objective function: $\Phi(\{F_{ij}, c_{ij}\})$
  - e.g., $\Phi(\{F_{ij}, c_{ij}\}) := \max_{(i, j) \in E} \{F_{ij} / c_{ij}\}$
  - or $\Phi(\{F_{ij}, c_{ij}\}) := \sum_{(i, j) \in E} \{F_{ij} / c_{ij}\}$
More Cost Functions

- works for rich set of cost functions
- example:
  \[ \Phi = \sum_{(i,j) \in E} \Phi_{ij} (d_k X_{ij}^k) \]
- where \( F_{ij} \) are piecewise linear
Traffic Engineering as a Global Optimization Problem (cont’d)

- Objective function: minimize $\Phi(\{F_{ij}, c_{ij}\})$

- Constraints:
  -- flow conservation: total outflow vs. total inflow
  $$\sum_{j: (i, j) \in E} F^k_{ij} - \sum_{j: (j, i) \in E} F^k_{ji} = \begin{cases} 
  0, & i \neq s_k, i \neq t_k, k \in K \\
  d_k, & i = s_k, i \neq t_k, k \in K \\
  -d_k, & i \neq s_k, i = t_k, k \in K 
\end{cases}$$

  -- capacity and (non-negative load) constraints
  $$\sum_{k \in K} F^k_{ij} \leq c_{ij}, \quad (i, j) \in E$$
  $$F^k_{ij} \geq 0$$
Traffic Engineering Example: minimize maximum link utilization

- Minimize $\Phi(\{F_{ij}, c_{ij}\}) := \max_{\{(i,j) \in E\}} \{\frac{F_{ij}}{c_{ij}}\}$
- Multi-commodity flow problem
  - There exists polynomial time solutions to the problem
- Equivalent **linear programming** formulation
  - $\alpha$ -- maximum link utilization
  - Let $X_{ij}^k := F_{ij}^k / d_k$
    
    - $X_{ij}^k$ denotes fraction of demand $k$ on link $(i,j)$
    - $X_{ij}^k \in [0,1]$
Traffic Engineering: LP Formulation

\[ \min \alpha \]

\[ \text{s.t. } \sum_{j:(i,j) \in E} X_{ij}^k - \sum_{j:(j,i) \in E} X_{ji}^k = \begin{cases} 0, & i \neq s_k, t_k, k \in K \\ 1, & i \neq s_k, k \in K \\ -1, & i \neq t_k, k \in K \end{cases} \]

\[ \sum_{k \in K} d_k X_{ij}^k \leq c_{ij} \alpha, \quad (i, j) \in E \]

\[ 0 \leq X_{ij}^k \leq 1 \]
Traffic Engineering w/ MPLS

• We can set up label-switched paths (LSPs) between origin-destination pairs to realize the optimal TE traffic load distributions.

• Let \( \{X^k_{ij}\} \) be the optimal solutions:
  - If for a given \( k \) (corresponding to a given O-D pair), \( X^k_{ij} = 0 \) or \( 1 \),
    then we set up one LSP (or tunnels) for the O-D pair.
  - Otherwise, traffic load for flow (demand) \( k \) is carried over multiple paths, we need to set up multiple LSPs (or tunnels) for the given O-D pair.
    • In general, traffic split among multiple LSPs are not equal!

• worst-case complexity:
  - \( O(N^2E) \) LSPs/tunnels needed
Traffic Engineering w/o MPLS

- Can we perform traffic engineering without MPLS?
  - we need to use “shortest path” routing
  - But shortest paths are defined based on link weights
- TE becomes link weight assignment problem!
- Other constraints we need to take into account
  - destination-based routing: not <src, dst> pair-based!
  - multiple shortest paths ( “equal-cost” paths, ECPs) may exist and can be used for load-balancing
    - But typical equal splitting is used to split traffic among ECPs for a given destination prefix
    - On the other hand, multiple destination prefixes are mapped to the same egress point of a network!
Effect of link weights (see [FRT02])

- unit link weights
- local change to congested link
- global optimization
  - to balance link utilizations
Shortest Path Routing and Link Weight Assignment Problem

- Key Problem: how to assign link weights to optimize TE objectives under conventional link-state (shortest path) routing paradigm?
- Key Insight: traffic engineering optimization is closely related to optimal link weight assignment using “shortest path routing” (with some caveats!)
  - The relationship comes from duality properties of linear programming
    - optimal link weight assignment problem is a dual problem to the optimal traffic engineering problem!

For materials in slides 53-62, see [WWZ01]
Duality of Linear Programming

**Primal**

\[
\text{maximize} \quad c^T x \\
\text{subject to} \quad Ax = b, \quad x \geq 0,
\]

**Dual**

\[
\text{minimize} \quad y^T b \\
\text{subject to} \quad y^T A \leq c^T
\]

- Y’s are Lagrange multipliers for equality constraints \( Ax=b \); \( z \geq 0 \)
  - Lagrange multipliers for inequality constraints \( x \geq 0 \)
- Lagrange function: \( L(x,y,z) := c^T x - y^T (Ax - b) + z^T x \geq c^T x, \) if \( x \) is a feasible solution
- Lagrange dual: \( g(y,z) := \inf_{x \geq 0} L(x,y,z) \geq c^T x^\star \) (\( x^\star \) optimal solution)
  - \( g(y,z) = y^T b \) if \( c^T - y^T A + z^T \geq 0 \) or \( y^T A \leq c^T \) as \( z^T \geq 0 \)
Complementary Slackness.

Let $x$ and $y$ be feasible solutions. A necessary and sufficient condition for them to be optimal is that for all $i$

1. $x_i > 0 \implies y^\top A_i = c_i$
2. $x_i = 0 \iff y^\top A_i < c_i$

Here $A_i$ is $i$-th column of $A$.
Example: Primal (P-SP)

- topology $G = (V,E)$, link weights $\{w_{ij} : (i, j) \in E\}$

$$\min \sum_{(i,j) \in E} w_{ij} \left( \sum_k X_{ij}^k \right)$$

s.t.

$$\sum_{j : (i,j) \in E} X_{ij}^k - \sum_{j : (j,i) \in E} X_{ji}^k = 0, \quad i \neq s_k, t_k$$

$$\sum_{j : (s_k,j) \in E} X_{ij}^k - \sum_{j : (j,s_k) \in E} X_{ji}^k = 1,$$

$$\sum_{j : (t_k,j) \in E} X_{ij}^k - \sum_{j : (j,t_k) \in E} X_{ji}^k = -1,$$

$$X_{ij}^k \geq 0$$
Example: Dual (D-SP)

\[\text{max} \sum_{k \in K} U^k_{t_k}\]

s.t. \[U^k_j - U^k_i \leq w_{ij}, \quad k \in K\]
\[(i, j) \in E\]

\[U^k_{s_k} = 0, \quad k \in K\]
Dual Solution Interpretation

\[
\max \sum_{k \in K} U^k_{t_k}
\]

s.t. \quad U^k_j - U^k_i \leq w_{ij}, \quad (i, j) \in E

\[
U^k_i = 0, \quad k \in K
\]

- \( \{ \overline{U}^k_i \} \) optimal solution to dual problem
- \( \overline{X}^k_{ij} > 0 \Rightarrow \overline{U}^k_j - \overline{U}^k_i = w_{ij}, \quad \overline{U}^k_j \) length of shortest path from \( s_k \) to \( j \)
- \( \overline{U}^k_i \) length of shortest path from \( s_k \) to \( t_k \)
Duality (More General Form)

**Primal**

maximize $c^T x$

subject to $Ax = b_1$

$x \geq 0,$

**Dual**

minimize $y_1^T b_1 + y_2^T b_2$

subject to $y_1^T A + y_2^T A' \leq c^T$

$A' x \geq b_2$

$y_2 \geq 0$

$y_1$: Lagrange multipliers for equality constraints $Ax=b_1$;

$y_2 \geq 0$: Lagrange multipliers for inequality constraints $A'x \geq b_2$

Lagrange dual: $g(y_1, y_2) := \inf_{x \geq 0} \{ c^T x + y_1^T (b_1 - Ax) + y_2^T (b_2 - A'x) \}$
Load Balancing Optimization Problem

\[
\begin{align*}
\min & \quad \sum_{(i,j) \in E} \sum_{k \in K} d_k X^k_{ij} \\
\text{s.t.} & \quad \sum_{j:(i,j) \in E} X^k_{ij} - \sum_{j:(j,i) \in E} X^k_{ji} = \begin{cases} 
0, & i \neq s_k, t_k, k \in K \\
1, & i \neq s_k, k \in K \\
-1, & i \neq t_k, k \in K 
\end{cases} \\
\sum_{k \in K} d_k X^k_{ij} & \leq c_{ij}, \quad (i,j) \in E \\
0 & \leq X^k_{ij} \leq 1
\end{align*}
\]
Re-formulating the Problem

• Let \(\{X^*_{kij}\}\) be optimal solutions, then \(d_kX^*_{kij}\) is the load of demand (flow) \(k\) placed on link \((i,j)\)

• Define \(C^*_{ij} := \sum_k d_kX^*_k\)
  -- total load of all demands on link \((i,j)\); \(C^*_{ij} b C_{ij}\)

Primal Problem:  \[
\min \quad \sum_{(i,j) \in E} \sum_{k \in K} d_k X^*_{ij}
\]

s.t. \[
\sum_{j:(i,j) \in E} X^k_{ij} - \sum_{j:(j,i) \in E} X^k_{ji} = \begin{cases} 
0, & i \neq s_k, t_k, k \in K \\
1, & i \neq s_k, k \in K \\
-1, & i \neq t_k, k \in K 
\end{cases}
\]

\[\sum_{k \in K} d_k X^k_{ij} \leq c_{ij}, \quad (i, j) \in E\]

\[0 \leq X^k_{ij} \leq 1\]
Dual Formulation

• dual variables \( \{U_i^k\}, \{W_{ij}\} \)

\[
\max \sum_{k \in K} d_k t_k U^k_I - \sum_{(i,j) \in E} c_{ij} W_{ij}
\]

s.t. \( U_j^k - U_i^k \leq W_{ij} + 1, \quad k \in K, (i, j) \in E \)

\( W_{ij} \geq 0 \)

\( U^k_s = 0 \)
Properties of Primal-Dual Solutions

- optimal solution to primal problem
  \[ \{ \overline{X}^{k}_{ij} \} \]
  dual problem
  \[ \{ \overline{U}^{k}_{i} \}, \{ \overline{W}_{ij} \} \]
- if \( \overline{X}^{k}_{ij} > 0 \), then
  \[ \overline{U}^{k}_{j} - \overline{U}^{k}_{i} = \overline{W}_{ij} + 1 \]
- can think of \( \overline{U}^{k}_{j} \) as shortest path distance from \( s_{k} \) to \( j \) when link weights are
  \[ \{ \overline{W}_{ij} + 1 \} \]

Therefore: solution to TE problem is also solution to shortest path problem with

\[ w_{ij} = \overline{W}_{ij} + 1 \]
Link Weight Assignment: Generalization

- works for rich set of cost functions
- example:
  \[ \Phi = \sum_{(i,j) \in E} \Phi_{ij} \left( \sum_{k \in K} d_k X_{ij}^k \right) \]
  - where \( F_{ij} \) are piecewise linear
Issues

• solutions are flow specific - need destination specific solutions
  - not a big deal, can reformulate to account for this

• solutions may not support equal split rule of OSPF
  - accounting for this yields NP-hard problem
  - modify IP routing
One approach to overcome the “splitting problem”

- current routing tables have thousands of routing prefixes
- instead of routing each prefix on all equal cost paths, selectively assign next hops to (each) prefix
  - i.e., remove some equal cost next hops assigned to prefixes
- goal: to approximate optimal link load

see [FT00], [FRT02] and [SDG05]
Example: EQUAL-SUBSET-SPLIT

Prefix A: 5
Prefix B: 1
Prefix C: 8
Prefix D: 10

Prefixes: D C
5 + 4 = 9

Prefixes: A B
2.5 + 0.5 = 3

Prefixes: D C B A
5 + 4 + 2.5 + 0.5 = 12
Advantages

• requires no change in data path
• can leverage existing routing protocols
• current routers have 10,000s of routes in routing tables
  - provides large degree of flexibility in next hop allocation to match optimal allocation
Summary

• can use OSPF/ISIS to support traffic engineering objectives

• performance objectives link weights
  - Further considerations:
    • Link weight assignment under multiple traffic matrices, and/or under multiple topologies (under link failures)

• equal splitting rule complicates problem
  - heuristics provide good performance
  - small changes to IP routing provide in better performance

• MPLS suffers none of these problems, but protocol more complex!