Backbone Traffic Engineering

Lecture 20, Computer Networks (198:552)
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)
• … these are the essential questions of traffic engineering
Routing With “Static” Link Weights

• 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

![Routing Diagram]
Setting the Link Weights

- How to set the weights
  - Inversely proportional to link capacity?
  - Proportional to propagation delay?
  - Network-wide optimization based on traffic?
Measure, Model, and Control

Network-wide “what if” model

Topology/Configuration

Offered traffic

Changes to the network

Operational network

measure

control
Key Ingredients

- **Measurement**
  - Topology: monitoring of the routing protocols
  - Traffic matrix: passive 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 constraints
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 $l$
Equal-Cost Multipath (ECMP)

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

• Computing the link utilization
  • Link load: \( u_l = \sum_{i,j} M_{i,j} \cdot P_{i,j,l} \)
  • Utilization: \( u_l / c_l \)

• Objective functions
  • \( \min(\max_i(u_l / c_l)) \)
  • \( \min(\sum_i f(u_l / c_l)) \)
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

• Clearly suboptimal, but effective in practice
  • Fast computation of the link weights
  • Good performance, compared to “optimal” solution
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
Central Control over Distributed Routing

Stefano Visicchio et al., ACM SIGCOMM’15
Context: Scalable & robust backbone TE

- Traditional IGPs perform distributed computations
  - Scalable
  - Robust to link and node failures
  - But not flexible in terms of expressed paths
- SDNs perform (logically) centralized path computations
  - Highly flexible
  - But handling large networks is challenging
  - Handling topology updates is challenging

- Could we combine the best of both worlds?
An example
Key idea: Lies to the routers
Another example

(a) Initial topology
(b) Augmented topology
Workflow

Compilation

Augmentation

Optimization

Injection/Monitoring

path reqs. + network topology → per-destination forwarding DAGs → augmented topology → reduced topology → running network
Fibbing: Expressiveness

**Theorem 1.** Any set of per-destination forwarding DAGs can always be enforced by augmenting a Fibbing-compliant topology even only with globally-scoped lies.
Key primitives required

- Initial setting of static routing weights: “Fibbing-compliance”
- Local and global scoping of IGP announcements
- Forwarding traffic to fake nodes on any desired link
Augmentation algorithms

(a) Requirements

(b) Simple augmentation

(c) Merger augmentation
Cross-destination optimization
A comparison of approaches

<table>
<thead>
<tr>
<th></th>
<th>centralized/SDN</th>
<th>distributed/traditional</th>
<th>hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>forwarding paths:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>configuration</td>
<td>simple (declarative &amp; global)</td>
<td>complex (indirect &amp; per-device)</td>
<td>simple (declarative &amp; global)</td>
</tr>
<tr>
<td>manageability</td>
<td>high (direct control)</td>
<td>low [7, 8] (need for coordination)</td>
<td>high (direct control)</td>
</tr>
<tr>
<td>path installation</td>
<td>slow (by controller, per-device)</td>
<td>fast (by device, distributed)</td>
<td>fast (by device, distributed)</td>
</tr>
<tr>
<td><strong>robustness:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>network failures</td>
<td>slow (by controller)</td>
<td>fast (local)</td>
<td>fast (local)</td>
</tr>
<tr>
<td>controller failures</td>
<td>hard (ad-hoc synch)</td>
<td>native (distributed)</td>
<td>easy (synch via IGP)</td>
</tr>
<tr>
<td>partitions</td>
<td>hard (uncontrollable devices)</td>
<td>best (distributed)</td>
<td>best (fallback on distributed)</td>
</tr>
<tr>
<td><strong>routing policies:</strong></td>
<td>highest (any path)</td>
<td>- low for IGP (shortest paths)</td>
<td>high (any non-loopy paths)</td>
</tr>
</tbody>
</table>
Acknowledgment

• Slides heavily adapted from material by Jennifer Rexford and Stefano Visicchio