Network Traffic Engineering (I)

EL 933, Class9
Yong Liu
11/15/2005

Motivation

- Network is highly dynamic
  - traffic demand: time variant, flash crowd
  - network resource: add/drop/failure
- Network control
  - rate control: end-end
  - routing conf.: network operator, inter/intra domain
- Good traffic engineering
  - quality of service (QoS) of end users
  - efficient use of network resource
    - minimize network cost given traffic demand
    - maximize traffic demand given resource
- beyond performance: security/reliability/resilience

Outline

- Overview of Internet routing
  - slides from Prof. Lixin Gao, UMass
- Paper1:
  - slides modified from authors'
- Paper2:
  - slides modified from authors'

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
Route Construction

- **Static Routing**
  - listed manually: change route slowly
  - not robust: reachability is independent of network condition
  - stable

- **Dynamic Routing**
  - learn route via routing protocols
  - react to topology, traffic or configuration changes directly
  - might not converge or oscillate
  - might have loop

Dynamic Routing Algorithms

- **Global or Link state algorithm**
  - use global knowledge about topology and cost

- **Decentralized or Distance Vector algorithm**
  - use only knowledge of attached links and neighbors
  - iterative algorithm

Global or Link State Algorithm

- **Dijkstra’s shortest path algorithm**

**Implementation:**
- each node broadcast its connectivity and link costs to all nodes

A Link-State Routing Algorithm

- **Dijkstra’s algorithm**
  - net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
  - computes least cost paths from one node (“source”) to all other nodes
  - gives routing table for that node
  - iterative: after k iterations, know least cost path to k dest.’s

**Notation:**
- c(i,j): link cost from node i to j.
  cost infinite if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known
Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>start</th>
<th>D(B),p(B)</th>
<th>D(C),p(C)</th>
<th>D(D),p(D)</th>
<th>D(E),p(E)</th>
<th>D(F),p(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2,A</td>
<td>5,A</td>
<td>1,A</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2,A</td>
<td>4,D</td>
<td>2,D</td>
<td>infinity</td>
<td>infinity</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2,A</td>
<td>3,E</td>
<td>4,E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3,E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decentralized or Distance Vector Algorithm

- each node communicates only with directly-attached neighbors
- computes shortest path
- continues until no nodes exchange information

Distance Vector Routing Algorithm

iterative:
- continues until no nodes exchange info.
- self-terminating: no “signal” to stop

distributed:
- each node communicates only with directly-attached neighbors
- example: in node X, for dest. Y via neighbor Z:
  \[ D(Y,Z) = c(X,Z) + \min \{D(Y,w)\} \]

Distance Table: example

<table>
<thead>
<tr>
<th>D^E()</th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>
Routing in the Internet

So far
- all routers identical
- network "flat"
... not true in practice

scale: with 50 million destinations:
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

Internet Structure

- Thousands of Organizations
- Many many Routers
- Lots of Hosts

Routing Protocols

- Divide into Autonomous Systems (AS)
  - according to administrative domains
    - internet Service Providers (ISP)
    - cooperate networks
    - college campuses

- Two kinds of routing protocols
  - intra-Domain Routing (IGP)
    - within one domain
  - inter-Domain Routing (EGP)
    - among different domains

Intra-Domain Routing

- Goal:
  - find a "good" path (sequence of routers) through network from source to destination
    - delay, loss, bandwidth, cost or other definitions

- Static routing

- Popular dynamic routing protocols
  - RIP: Routing Information Protocol
  - IS-IS: Intermediate-System-to-Intermediate System
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
Intra-Domain Routing

- **Routing Information Protocol (RIP)**
  - Distance Vector Algorithm
- **Open Shortest Path First (OSPF)**
  - Link State Algorithm
- **IS-IS**
  - Link State Algorithm

**OSPF**

- Link state routing
- Each router keeps a complete map of the network
  - Rather than just how to get to each of the other routers
  - All routers should have exactly the same map.
- Routing updates are "flooded" to all nodes
- Compute shortest paths between any two points
  - Dijkstra's shortest path algorithm
- Converge fast when the network topology changes
- Link weights configured by the network operator

Inter-Domain Routing

- Internet consists of ~12,000 Autonomous Systems
- ASes exchange info about who they can reach
- Local policies for selecting and propagating routes
- Policies configured by the AS's network operators

**Inter-Domain Routing Protocols**

- Use EGP in NSFNET
- **Border Gateway Protocol (BGP)**
  - BGP-4: de facto standard
  - Path Vector Algorithm
Traffic Engineering With Traditional IP Routing Protocols

B. Fortz, J. Rexford, and M. Thorup

IP Network Operations

- Don’t IP networks manage themselves?
  - TCP adapts sending rate to network congestion
  - routing protocols adapt to changes in topology
- … not if we want to network to run well
  - adjust the routing of traffic to the prevailing load
  - ensure the network can accommodate failures
  - plan the outlay of new routers and links over time
- The driving goals
  - good end-to-end performance for users
  - efficient use of the network resources
  - reliable system even in the presence of failures

Outline

- IP network operations
  - motivation and examples
  - measure, model, and control
- Traffic engineering
  - measuring traffic and topology
  - modeling intradomain routing
  - optimization of routing weights
- Conclusions and ongoing work

Our Approach: Measure, Model, and Control

Operational network

Network-wide "what if" model

Topology/Configuration

Offered traffic

Changes to the network

measure

control
Key Ingredients of Our Approach

- **Instrumentation**
  - offered load: widely deployed traffic measurement
  - topology: monitoring of the routing protocols

- **Network-wide models**
  - representations of traffic and topology
  - "What-if" models of resource allocation policies

- **Network optimization**
  - efficient algorithms to find good configurations
  - operational experience to identify key constraints

Example: traffic engineering by tuning routing protocols

Traffic Engineering in an ISP Backbone

- **Network topology**
  - connectivity and capacity of routers and links

- **Configurable policies for resource allocation**
  - interdomain policies and intradomain weights

- **Traffic demands**
  - expected load between points in the network

- **Performance objective**
  - balanced load, low delay, service level agreements

- **Question**: Given the topology and traffic, which routing configuration should be used?

Traffic Matrix: offered Traffic

- **Flow-level measurement (Cisco Netflow)**
  - measurements at the level of TCP/UDP flows
  - addresses, port #s, #bytes/packets, start/finish
  - collected on links connecting AT&T to its peers

- **Collection of the measurement data**
  - distributed set of collection servers in the network
  - software for online aggregation of the data
  - computation of a “traffic matrix” for the network

Topology/Routing

- **Router configuration files**
  - daily snapshot of network assets & configuration
  - software to parse the router config commands
  - network-wide view of topology & routing policies
  - also useful for detecting configuration mistakes

- **Routing monitors**
  - online monitoring of routing protocol messages
  - real-time view of routes via neighboring ASes
  - real-time view of paths within the AS
  - software for aggregating and querying the data
  - also useful for detecting and diagnosing anomalies
Network Model

- **Data model**
  - physical level, IP level, router-complex level
  - traffic demands, router attributes, link attributes

- **Routing model**
  - shortest-path routing, with tie-breaking
  - multi-homed customers, inter-domain routing
  - book-keeping to accumulate load on each link

- **Visualization environment**
  - coloring/sizing to illustrate link and node statistics
  - querying to show statistics for links and nodes
  - what-if experiments with routing configurations

Network Optimization: The Problem

- **Intradomain traffic engineering**
  - predict influence of weight changes on traffic flow
  - minimize objective function (say, of link utilization)

- **Inputs**
  - networks topology: capacitated, directed graph
  - routing configuration: routing weight for each link
  - traffic matrix: offered load each pair of nodes

- **Outputs**
  - shortest path(s) for each node pair
  - volume of traffic on each link in the graph
  - value of the objective function

Example: Traffic Through Backbone

Source node: public peering link in New York
Destination nodes: AT&T access routers

<table>
<thead>
<tr>
<th>Color/size of node: proportional to traffic to this router (high to low)</th>
</tr>
</thead>
</table>

| Color/size of link: proportional to traffic carried (high to low) |

Link Weights Adjustment: example

- **Balance traffic by changing link weights**
Network Optimization: Our Approach

- Local search
  - generate a candidate setting of the weights
  - predict the resulting load on the network links
  - compute the value of the objective function
  - repeat, and select solution with lowest objective function

- Computation
  - explore the "neighborhood" around good solutions
  - exploit efficient incremental graph algorithms

- Performance results on AT&T's network
  - much better than simple heuristics
    - weights inversely proportional to capacity
    - weights proportional to physical distance
  - competitive with multi-commodity flow solution
    - Optimal routing possible with more flexible routing protocols

Network Optimizations: Operational Realities

- Minimize changes to the network
  - changing just one or two 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 the number of distinct weight values
  - small number of integer values is sufficient
- Limit dependence on accuracy of traffic matrix
  - good weights remain good after introducing random noise
- Limit frequency of changes to the weights
  - joint optimization for day and night traffic matrices

Conclusions

- Our approach
  - measure: network-wide view of traffic and routing
  - model: data representations and "what-if" tools
  - control: intelligent changes to operational network

- Other applications
  - visualization of traffic, performance, and reliability
  - capacity planning to place new routers and links
  - estimating impact of new customers on network
  - evaluating the effects of router and link failures
  - comparing benefits of different routing protocols

Internet Traffic Engineering by Optimizing OSPF Weights
Bernard Fortz and Mikkel Thorup
Outline

- Base-line optimal routing
- Difficulty in OSPF weights Optimization
- Local search algorithm
- Experiments on operational networks

Optimal Routing

- Design Space:
  - given source destination pairs (TM)
  - any possible paths between any two pairs

- Objective:
  - link incurs cost by carrying traffic
  - network cost is the summation of all link costs
  - find a set of routes to minimize network cost

Optimal Routing: formulation

- **routing variable:** \( f_a^{(s,t)} , \alpha \in A, (s, t) \in N \times N \)
- **link data rate:** \( l(\alpha) = \sum_{(s,t) \in N \times N} f_a^{(s,t)} \)
- **link cost:** increasing, convex function of link data rate
  - example: \( \Phi_\alpha = \frac{l(\alpha)}{c(\alpha) - l(\alpha)} \) 
    - \( M/M/1 \) delay

- **network cost**
  - summation: \( \Phi = \sum_{\alpha \in A} \Phi_\alpha \)
  - normalization: \( \Phi^* = \frac{\Phi}{\sum_{(s,t) \in N \times N} (D(s, t) \times dist_1(s, t))} \)

Optimal Routing: formulation

- **network cost:** \( \min \Phi = \sum_{\alpha \in A} \frac{l(\alpha)}{c(\alpha) - l(\alpha)} \)
- **link rate:** \( l(\alpha) = \sum_{(s,t) \in N \times N} f_a^{(s,t)} , \alpha = (x, y) \in N \times N \)
- **feasible flow:** 
  - \( \sum_{x:(x,y) \in A} f^{(s,t)}_{(x,y)} - \sum_{z:(y,z) \in A} f^{(s,t)}_{(y,z)} = \begin{cases} -D(s, t) & \text{if } y = s \\ D(s, t) & \text{if } y = t \\ 0 & \text{otherwise} \end{cases} \)
  - \( f^{(s,t)}_{(x,y)} \geq 0, \forall x, y, s, t \in N \)

...can be solved by centralized and distributed algorithms...
Linear Programming

- approximate link cost function with piece-wise linear function

\[ \min \Phi = \sum_{a \in A} \Phi_a \]

subject to

\[ \sum_{x \in \mathcal{V} \times \mathcal{A}} f^{(x)}_{a} - \sum_{x \in \mathcal{V} \times \mathcal{A}} f^{(x)}_{a} = \begin{cases} -D(x) & \text{if } y = x, \\ 0 & \text{otherwise} \end{cases} \]

\[ \ell(a) = \sum_{s,t \in \mathcal{N} \times \mathcal{N}} f^{(s,t)}_{a}, \quad a \in A, \quad (2) \]

\[ \Phi_a \geq \ell(a) \quad a \in A, \quad (3) \]

\[ \Phi_a \geq \beta \cdot c(a) \quad a \in A, \quad (4) \]

\[ \Phi_a \geq 1000 \cdot c(a) - \frac{\beta}{\alpha} \cdot c(a) \quad a \in A, \quad (5) \]

\[ \Phi_a \geq 70 \cdot c(a) - \frac{\beta}{\alpha} \cdot c(a) \quad a \in A, \quad (6) \]

\[ \Phi_a \geq 500 \cdot c(a) - \frac{\beta}{\alpha} \cdot c(a) \quad a \in A, \quad (7) \]

\[ f^{(s,t)}_{a} \geq 0 \quad a \in A, \quad s,t \in \mathcal{A}. \quad (9) \]

Limitation with OSPF routing

- free repartition of flow not possible using OSPF
  - traffic equally split between shortest paths
- contrived example: cost of optimal OSPF policy may be up to 5000 times the cost of the optimal routing.
- the optimization of OSPF weights is NP-hard.

Local Search Heuristics

- Local search to reduce network cost
- Neighborhoods for OSPF weight setting
- Hashing tables to escape local minima
- Diversification
- Algorithmic aspects
- Numerical results

Neighborhood of OSPF weight setting

- Change the weight of one arc.
- Change the weights of all arcs leaving a node to make paths of equal weights.

Too big neighborhoods: random sampling.
Hash tables to avoid cycling

- A solution is completely determined by a set of weights.
- Hashing: compression of a set of weights into one hash value (storable on 16 bits).
- Record hash value of each iterate and reject solutions having a value already encountered.
- Completely avoids cycling.

Diversification

- Local search converges to local minimum
- Key tool for exploring different regions of the solution space.
- Two approaches:
  - smaller hashing table used to create collisions and reject more and more solutions;
  - random perturbations.

Cost Evaluation

- Evaluation network cost if one weight setting is implemented (algorithm perf. bottleneck)
- Shortest path tree to a destination \( t \)
  - reverse link direction
  - shortest path tree from "source \( t \)" in the reversed graph
- Each node find all outgoing links on shortest path \( t \), and equally split traffic on those links
- Link rate is the summation of traffic between all src-dst pairs allocated on it

Routing Computation

For each destination \( t \):
- Shortest paths to \( t \) → Backward Dijkstra.
- Build the graph \( G' = (N, A') \) of arcs that belong to a shortest path to \( t \).
  \[ A' = \{(i, j) \in A : d(i, t) = w_{ij} + d(j, t)\} \]
- Visit the nodes in order of decreasing distance to \( t \). When visiting \( v \), set
  \[ l = \frac{1}{|\delta^+(v)|} \left( D(v, t) + \sum_{(u, v) \in A'} l'_{(u, v)} \right) \]
and set \( l'_{(u, v)} = l \) for each \((v, w) \in A'\).
Algorithmic Aspects

- Only a few weights change when considering a neighbor of a solution.
- Dynamic updates of the cost:
  - recompute only shortest paths that really change;
  - recompute flows only from nodes for which an incoming or outgoing arc has appeared in or disappeared from the shortest paths graph;
  - details in papers
- Computation time divided by 20!

Numerical Experiments

- AT&T WorldNet and synthetic networks.
- 5000 iterations of local search.
- Comparison with:
  - Optimal routing (lower bound).
  - UnitOSPF.
  - InvCapOSPF (CISCO recommendation).
  - L2OSPF
  - RandomOSPF (performs really bad).

Numerical Experiments

AT&T backbone with 90 nodes and 274 arcs
Numerical Experiments

- Our heuristic allows to deal with an increase of demands of 55% compared to simple OSPF strategies.
- Optimal routing can cope with a 3% increase in demands only, so it would be preferable to increase the hardware capacity.
- For realistic network topologies, OSPF works well (with our heuristic).

Extensions

- Restoring capacity with a few weight changes.
- Robustness vs. hot spots and link failures.
- Multiple demand matrices.
- Robust optimization for link failure scenarios.
To Read More...

- **Overview papers**
  - "Traffic engineering for IP networks"
    (http://www.research.att.com/~jrex/papers/ieeenet00.ps)

- **Topology and configuration**
  - "IP network configuration for intradomain traffic engineering"
    (http://www.research.att.com/~jrex/papers/ieeenet01.ps)
  - "An OSPF topology server: Design and evaluation"
    (http://www.cse.ucsc.edu/~amagh/jsac01-paper.pdf)

- **Intradomain route optimization**
  - "Optimizing OSPF/IS-IS weights in a changing world"
    (http://www.research.att.com/~mthorup/PAPERS/change_ospf.ps)

- **Routing optimization**