Chapter 15: Network Applications

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning Overview

Licence

Outline

1. Traffic Placement
2. Capacity Management
3. Other Problems

Common Theme

- How can we get better performance out of a given network?
- Make network transparent
  - Users should not need to know about details
  - Service maintained even if failures occur
- Restricted by accepted techniques available in hardware
  - Interoperability between multi-vendor equipment
  - Very conversative deployment strategies

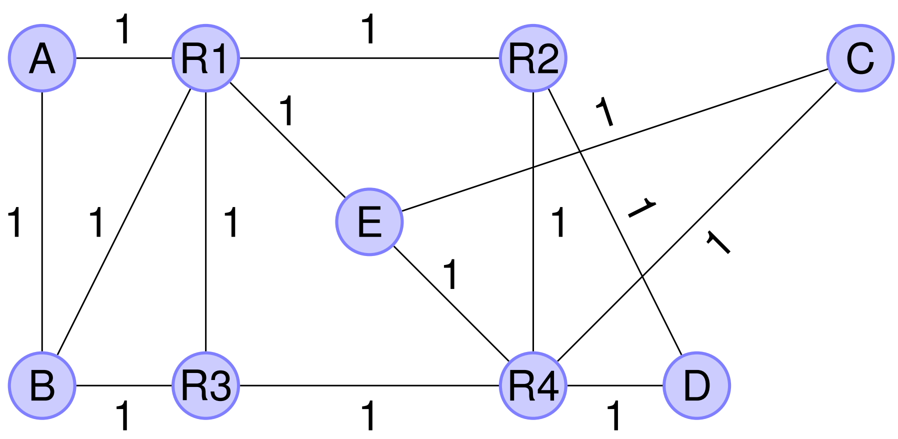
Reminder: IP Networks

- Packet forwarding
- Connection-less
- Destination based routing
  - Distributed routing algorithm based on shortest path algorithm
  - Routing metric determines preferred path
- Best effort
  - Packets are dropped when there is too much traffic on interface
  - Guaranteed delivery handled at other layers (TCP/applications)

Disclaimer

- Flexible border between CP and OR
- CP is ...
  - what CP people do.
  - what is published in CP conferences.
  - what uses CP languages.
- Does not mean that other approaches are less valid!

Example Network (Uniform metric 1, Capacity 100)



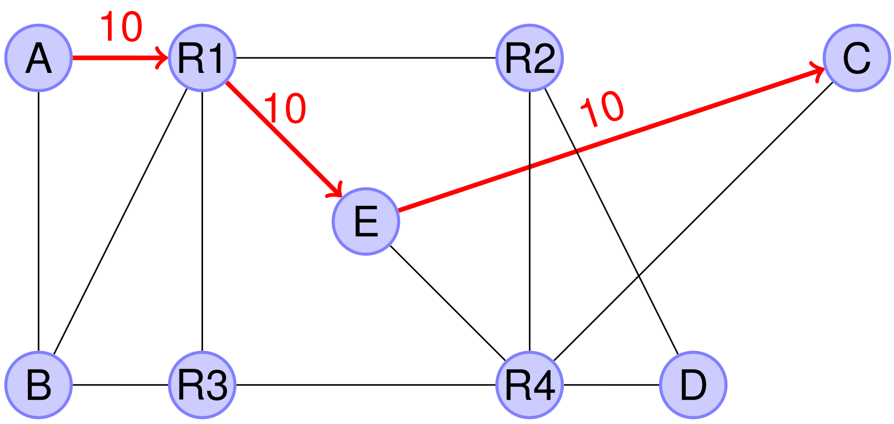
Example Traffic Matrix

Only partially filled in for example

	A	B	C	D	E
A	0	0	10	20	20
B	0	0	10	20	20
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

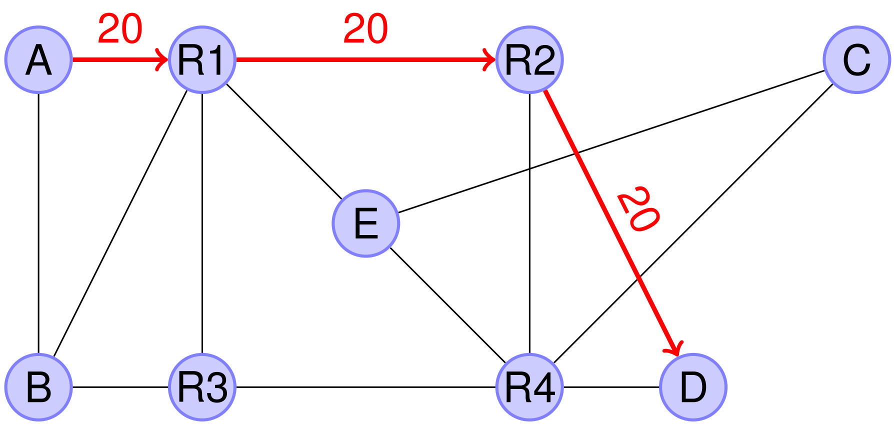
Using Routing

Demand AC 10

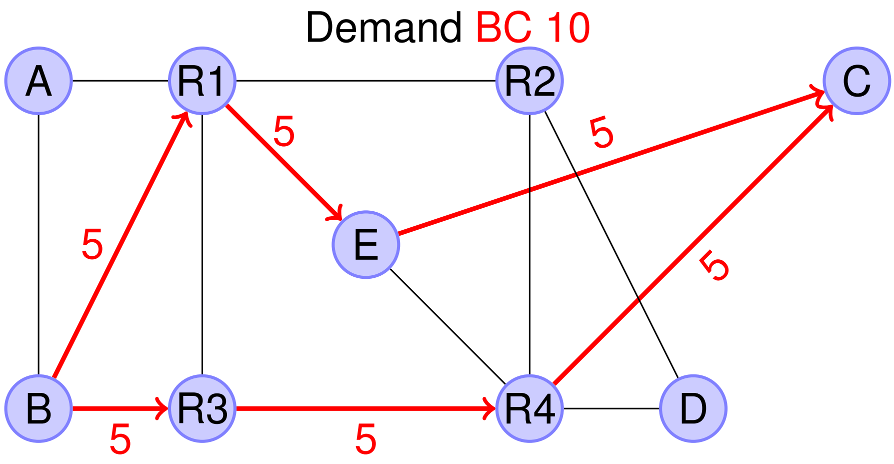


Using Routing

Demand AD 20

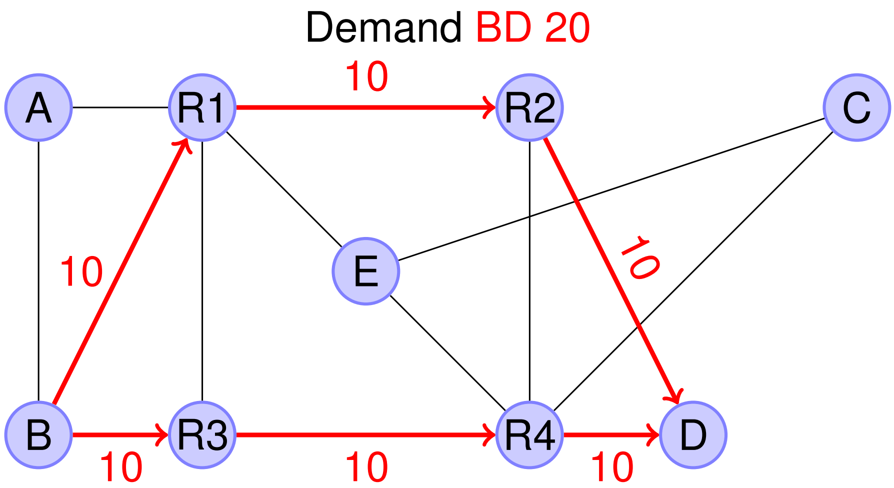


Using Routing

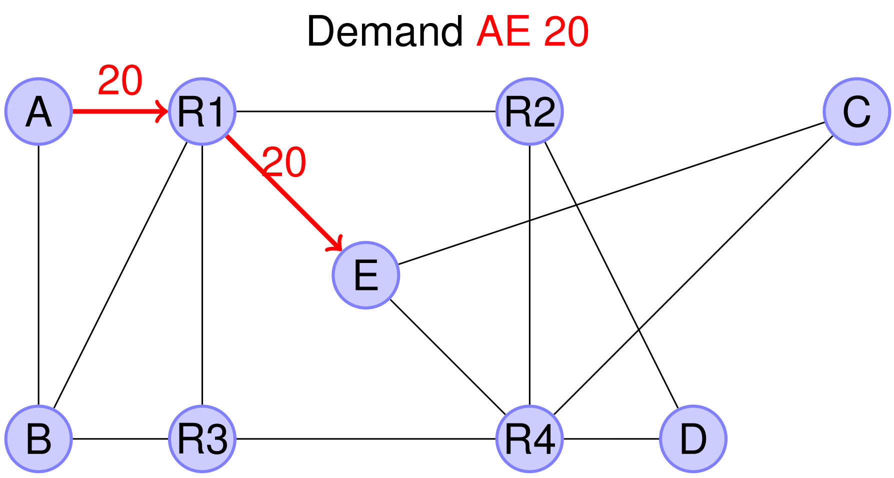


Traffic Placement
Capacity Management
Other Problems
Link Based Model
Path-Based Model
Node-Based Model
Commercial Solution
Multiple Paths

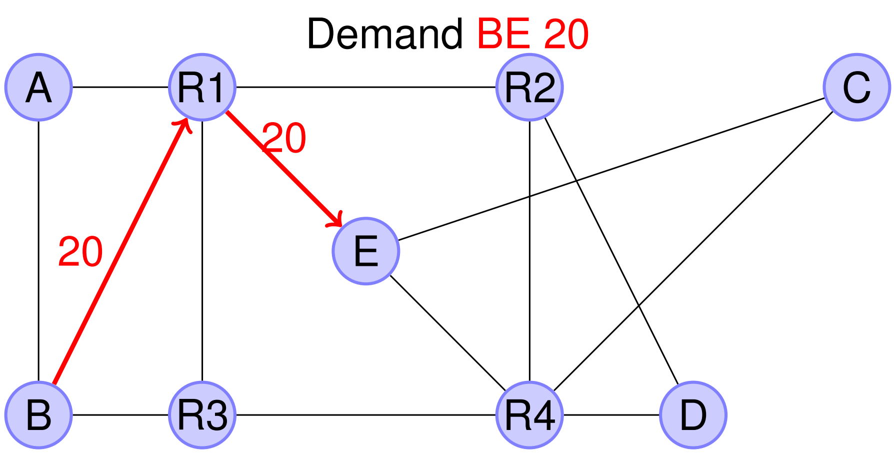
Using Routing



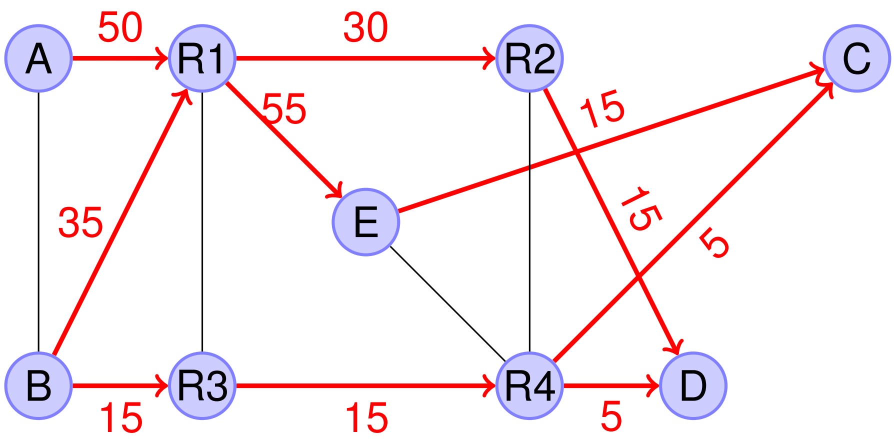
Using Routing



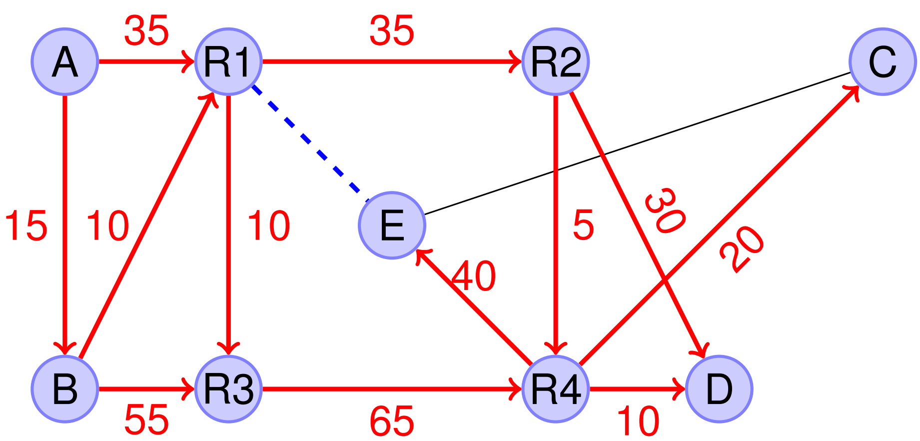
Using Routing



Resulting Network Load



Considering failure of R1-E



Can we do better?

- Choose single, explicit path for each demand
- Requires hardware support in routers (MPLS-TE)
- Baseline: CSPF, greedy heuristic

Why not just use Multi-Commodity Flow Problem Solution?

- Can not use arbitrary, fractional flows in hardware
- MILP does not scale too well

Modelling Alternatives

- Link based Model
- Path based Model
- Node based Model

Variants

- Demand Acceptance
  - Choose which demands to select fitting into available capacity
- Traffic Placement
  - All demands must be placed

Link-Based Model: Intuition

- Decide if demand d is run over link e
- Select which demands run over link e (Knapsack)
- Demand d must run from source to sink (Path)
- Sum of delay on path should be limited (QoS)

Link Based Model

$$\min_{\{X_{de}\}} \max_{e \in \mathbf{E}} \frac{1}{\mathtt{cap}(e)} \sum_{d \in \mathbf{D}} \mathtt{bw}(d) X_{de} \quad \mathtt{or} \quad \min_{\{X_{de}\}} \sum_{e \in \mathbf{E}, d \in \mathbf{D}} \mathtt{bw}(d) X_{de}$$

st.

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -1 & n = \mathtt{dest}(d) \\ 1 & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall e \in \mathbf{E}: \quad \sum_{d \in \mathbf{D}} \mathtt{bw}(d) X_{de} \leq \mathtt{cap}(e)$$

$$\forall d \in \mathbf{D}: \quad \sum_{e \in \mathbf{E}} \mathtt{del}(e) X_{de} \leq \mathtt{req}(d)$$

$$X_{de} \in \{0, 1\}$$

Solution Methods

- Lagrangian Relaxation
  - Path decomposition
  - Knapsack decomposition
- Probe Backtracking

Lagrangian Relaxation - Path decomposition

[Ouaja&Richards2003]

- Dualize capacity constraints
- Starting with CSPF initial solution
- Finite domain solver for path constraints
- Added capacity constraints from st-cuts
- At each step solve shortest path problems

Lagrangian Relaxation - Knapsack decomposition

[Ouaja&Richards2005]

- Dualize path constraints
- At each step solve knapsack problems
- Reduced cost based filtering

Probe Backtracking

[Liatsos et al 2003]

- Start with (infeasible) CSPF heuristic
- Consider capacity violation
  - Resolve by forcing one demand off/on link
  - Find new path respecting path and added constraints with ILP
- Repeat until no more violations, feasible solution
- Optimality proof when exhausted search space
  - Search space often very small

Path-Based Model: Intuition

- Choose one of the possible paths for demand d
- This paths competes with paths of other demands for bandwidth
- Usually too many paths to generate a priori, but most are useless

Cork Constraint Computation Centre

Path-Based Model

$$\max_{\{Z_d, Y_{id}\}} \sum_{d \in \mathbf{D}} \mathtt{val}(d) Z_d$$

st.

$$\forall d \in \mathbf{D} : \quad \sum_{1 \le i \le \mathtt{path}(d)} Y_{id} = Z_d$$

$$\forall e \in \mathbf{E} : \quad \sum_{d \in \mathbf{D}} \mathtt{bw}(d) \sum_{1 \le i \le \mathtt{path}(d)} h_{id}^{e} Y_{id} \le \mathtt{cap}(e)$$

$$Z_d \in \{0, 1\}$$

$$Y_{id} \in \{0, 1\}$$

Cork Constraint Computation Centre

Solution Methods

- Blocking Islands
- Local Search/ Finite Domain Hybrid
- (Column Generation)

Blocking Islands

[Frei&Faltings1999]

- Feasible solution only
- CSP with variables ranging over paths for demands
- No explicit domain representation
- Possible to perform forward checking by updating blocking island structure

Local Search/Finite Domain Hybrid

[Lever2004]

- Start with (feasible) CSPF heuristic
- Add more demands one by one
  - Use repair to solve capacity violations
- Use Finite Domain model to check necessary conditions
  - Determine bottlenecks by st-cuts
  - Force paths on/off links
- Define neighborhood by rerouting demands currently over violations

Node Based Model: Intuition

- For each demand, decide for each router where to go next
  - Many routers not used
- Treat link capacity with cumulative/diffn constraints
- Pure Finite Domain model, no global cost view

Cisco ISC-TEM

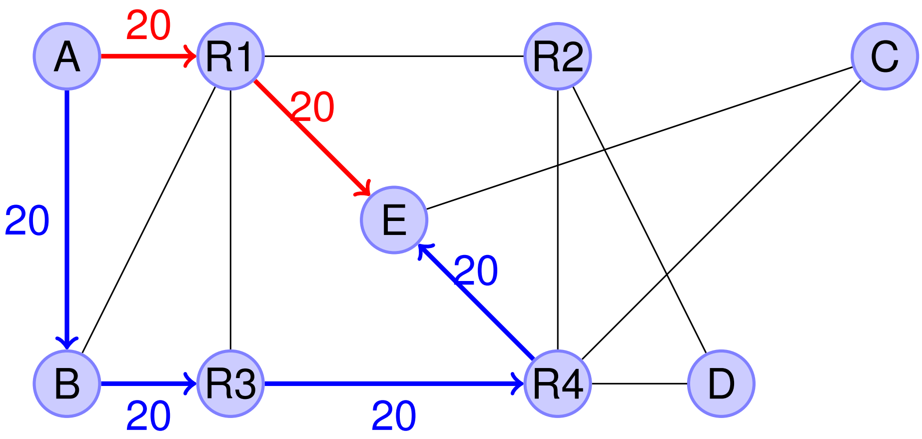
- Path placement algorithm developed for Cisco by PTL and IC-Parc (2002-2004)
- Internal competitive selection of approaches
- Strong emphasis on stability
- Written in ECLiPSe
- PTL bought by Cisco in 2004
- Part of team moved to Boston

Problem

- What happens if element on selected path fails?
- Choose second path which is link (element) disjoint
- State bandwidth constraints for each considered failure case
- Problem: Very large number of capacity constraints

Example

Primary/Secondary path for demand AE



Which bandwidth to count?

Failed Element	No Failure	A-R1	R1-E	All Others
Capacity for Path	Primary	Secondary	Secondary	Primary

Multiple Path Model

$$\max_{\{Z_d, X_{de}, W_{de}\}} \sum_{d \in \mathbf{D}} \mathtt{val}(d) Z_d$$

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathtt{dest}(d) \\ Z_d & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall e \in \mathbf{E}: \quad \sum_{d \in \mathbf{D}} \mathtt{bw}(d) * X_{de} \leq \mathtt{cap}(e)$$

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} W_{de} - \sum_{e \in \mathbf{IN}(n)} W_{de} = \begin{cases} -Z_d & n = \mathtt{dest}(d) \\ Z_d & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall e \in \mathbf{E}, \forall e' \in \mathbf{E} \setminus e: \quad \sum_{d \in \mathbf{D}} \mathtt{bw}(d) * (X_{de} - X_{de'} * X_{de} + X_{de'} * W_{de}) \leq \mathtt{cap}(e)$$

$$\forall d \in \mathbf{D}, \forall e \in \mathbf{E}: \quad X_{de} + W_{de} \leq 1$$

$$Z_d \in \{0,1\}, X_{de} \in \{0,1\}, W_{de} \in \{0,1\}$$

Solution Method

- Benders Decomposition [Xia&Simonis2005]
- Use MILP for standard demand acceptance problem
- Find two link disjoint paths for each demand
- Sub-problems consist of capacity constraints for failure cases
- Benders cuts are just no-good cuts for secondary violations

The Problem

- How to provide cost effective, high quality services running an IP network?
- Easy to build high quality network by massive over-provisioning
- Easy to build consumer grade network disregarding Quality of Service (QoS)
- Very hard to right-size a network, providing just enough capacity

Cork Constraint Computation Centre

The Approach

- Bandwidth on Demand
  - Create temporary bandwidth channels for high-value traffic
  - Avoid disturbing existing traffic
- Resilience Analysis
  - Find out how much capacity is required for current traffic
  - Provide enough capacity to survive element failures without service disruption

Cork Constraint Computation Centre

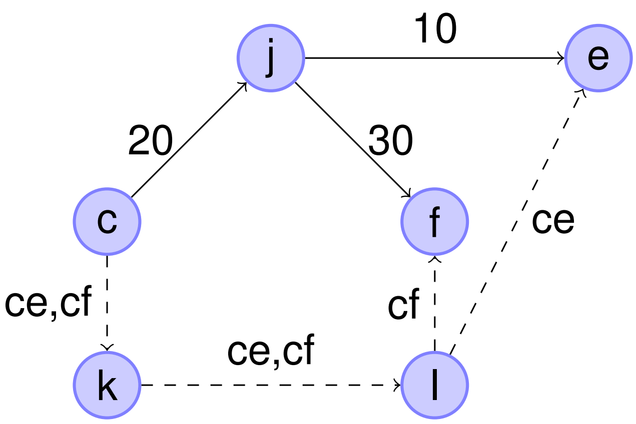
Background

- Failures of network should not affect services running on network
- Not cost effective to protect connections in hardware
- Response time is critical
  - Interruption > 50ms not acceptable for telephony
  - Reconvergence of IGP 1 sec (good setup)
  - Secondary tunnels rely on signalling of failure (too slow)
  - Live/Live connections too expensive

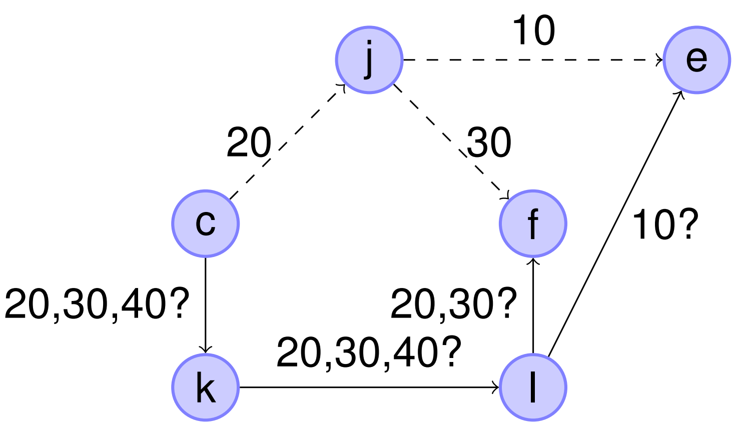
Approach

- Fast Re-route
  - If element fails, use detour around failure
  - Local repair, not global reaction
  - Pre-compute possible reactions, allows offline optimization
- Link protection rather easy
- Node protection quite difficult

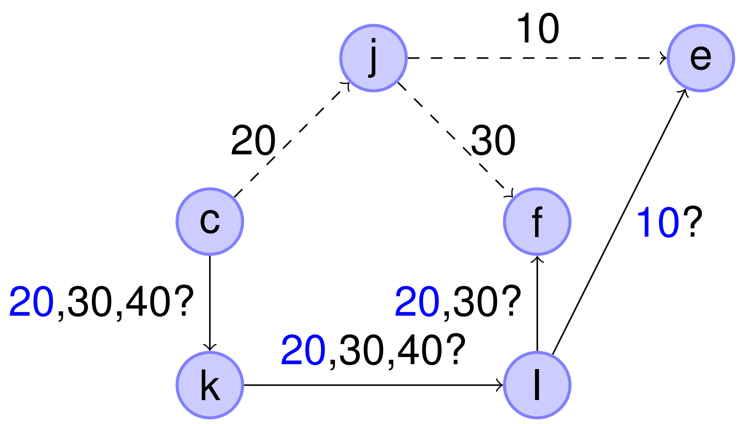
Example Problem



Node j Failure



Node j Failure (Result)



Bandwidth Protection Model

$$\min_{\{X_{fe}\}} \sum_{f \in \mathbf{F}} \sum_{e \in \mathbf{E}} X_{fe}$$

$$st. \begin{cases} \forall f \in \mathbf{F}: & \begin{cases} \forall n \in \mathbf{N} \setminus \{\texttt{orig}(f), \texttt{dest}(f)\}: & \sum_{e \in \mathbf{IN}(n)} X_{fe} = \sum_{e \in \mathbf{OUT}(n)} X_{fe} \\ n = \texttt{orig}(f): & \sum_{e \in \mathbf{OUT}(n)} X_{fe} = 1 \\ n = \texttt{dest}(f): & \sum_{e \in \mathbf{IN}(n)} X_{fe} = 1 \end{cases} \\ \forall e \in \mathbf{E}: & \texttt{cap}(e) \geq \begin{cases} \max_{\{Q_{fe}\}} \sum_{f \in \mathbf{F}} X_{fe} Q_{fe} \\ st. \begin{cases} \forall o \in \texttt{orig}(\mathbf{F}): & \texttt{ocap}(o) \geq \sum_{f:\texttt{orig}(f)=o} Q_{fe} \\ \forall d \in \texttt{dest}(\mathbf{F}): & \texttt{dcap}(o) \geq \sum_{f:\texttt{dest}(f)=d} Q_{fe} \end{cases} \end{cases} \\ & X_{fe} \in \{0, 1\} \\ & \texttt{quan}(f) \geq Q_{fe} \geq 0 \end{cases}$$

Solution Techniques

[Xia, Eremin & Wallace 2004]

- MILP
  - Use of Karusch-Kahn-Tucker condition
  - Removal of nested optimization
  - Large set of new variables
  - Not scalable
- Problem Decomposition
  - Integer Multi-Commodity Flow Problem
  - Capacity Optimization
- Improved MILP out-performs decomposition [Xia 2005]

Cork Constraint Computation Centre

Cisco Tunnel Builder Pro

- Algorithm/Implementation built by PTL/IC-Parc for Cisco
- Not based on published techniques above
- In period 2000-2003
- Written in ECLiPSe
- Embedded in Java GUI
- Now subsumed by ISC-TEM

Cork Constraint Computation Centre

Planning Ahead

- Consider demands with fixed start and end times
- Demands overlapping in time compete for bandwidth
- Demands arrive in batches, not always in temporal sequence
- Problem called Bandwidth on Demand (BoD)

Model: BoD

$$\max_{\{Z_d, X_{de}\}} \sum_{d \in \mathbf{D}} \mathtt{val}(d) Z_d$$

st.

$$\mathbf{T} = \{\mathtt{start}(d) | d \in \mathbf{D}\}$$

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N} : \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathtt{dest}(d) \\ Z_d & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall t \in \mathbf{T}, \forall e \in \mathbf{E} : \sum_{\substack{d \in \mathbf{D} \\ \mathtt{start}(d) \leq t \\ t < \mathtt{end}(d)}} \mathtt{bw}(d) X_{de} \leq \mathtt{cap}(e)$$

$$Z_d \in \{0, 1\}$$

$$X_{de} \in \{0, 1\}$$

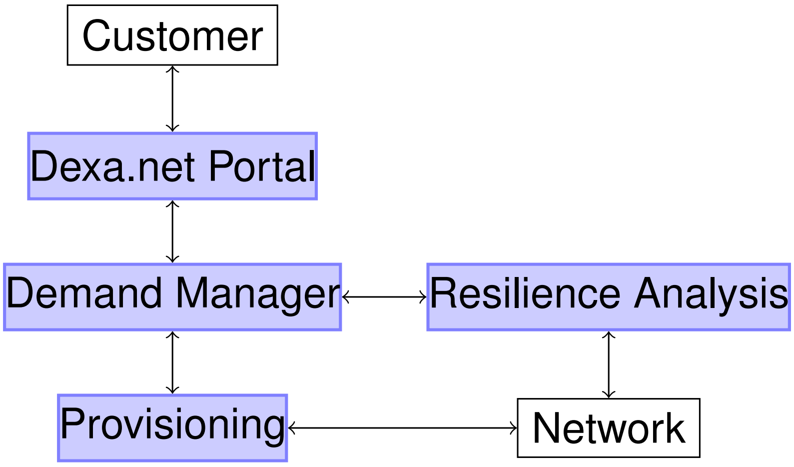
Solution Methods

- France Telecom for ATM network [Lauvergne et al 2002, Loudni et al 2003]
- Schlumberger Dexa.net (PTL, IC-Parc)

Schlumberger Dexa.net

- Small, but global MPLS TE+diffserv network
- Oil field services
- (Very) High value traffic
  - Well logging
  - Video conferencing
- Bandwidth demand known well in advance, fixed period
- Low latency, low jitter required

Architecture



Workflow

- Customer requests capacity for time slot via Web-interface
- Demand Manager determines if request can be satisfied
  - Based on free capacity predicted by Resilience Analysis
  - Taking other, accepted BoD requests into account
- Email back to customer
- At requested time, DM triggers provisioning tool to
  - Set up tunnel
  - Change admission control
- At end of period, DM pulls down tunnel

How much free capacity do we have in network?

- Easy for normal network state (OSS tools)
- Challenge: How much is required for possible failure scenarios?
- Consider single link, switch, router, PoP failures
- Classical solution
  - Get Traffic Matrix
  - Run scenarios through simulator

How to get a Traffic Matrix?

- Many algorithms assume given traffic matrix
- Traffic flow information is not collected in the routers
- Only link traffic is readily available
- Demand pattern changes over time, often quite dramatically
- Measuring traffic flows with probes is very costly

From a network consultant:

> *We have been working on extracting a TM for this network for 15 months, and we still don’t have a clue if we’ve got it right.*

Idea

- Use the observed traffic to deduce traffic flows
- Network Tomography [Vardi1996]
  - All flows routed over a link cause the observed traffic
  - Must correct for observation errors
  - Highly dependent on accurate routing model
- Gravity Model [Medina et al 2002]
  - Ignore core of network
  - Assume that flows are proportional to product of ingress/egress size
- Results are very hard to validate/falsify

Model: Traffic Flow Analysis

$$\forall i,j \in \mathbf{N}: \quad \min_{\{F_{ij}\}} / \max_{\{F_{ij}\}} \quad F_{ij}$$

st.

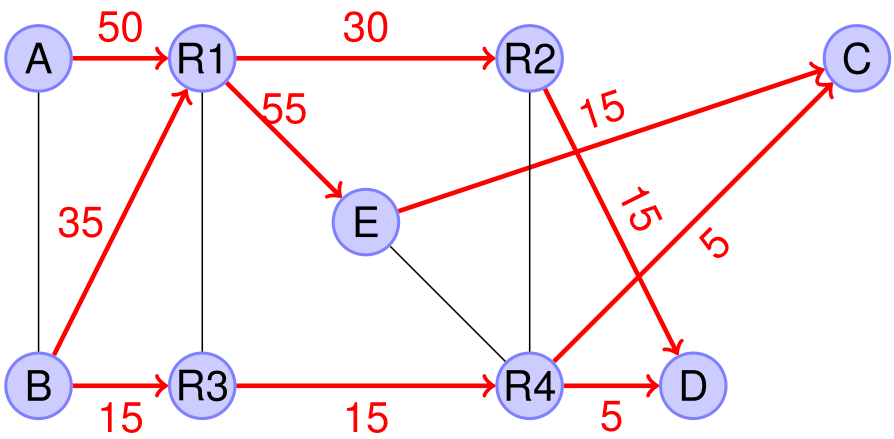
$$\forall e \in \mathbf{E}: \quad \sum_{i,j \in \mathbf{N}} r_{ij}^{e} F_{ij} = \mathtt{traf}(e)$$

$$\forall i \in \mathbf{N}: \quad \sum_{j \in \mathbf{N}} F_{ij} = \mathtt{ext}^{in}(i)$$

$$\forall j \in \mathbf{N}: \quad \sum_{i \in \mathbf{N}} F_{ij} = \mathtt{ext}^{out}(j)$$

$$F_{ij} \geq 0$$

Start with Link Traffic



Setup Model to Find Flows

```
[AC,AD,BC,BD,AE,BE] :: 0.0 .. 1.0Inf,
AC + AD + AE $= 50, % A R1
0.5*BC + 0.5*BD + BE $= 35, % B R1
0.5*BC + 0.5*BD $= 15, % B R3
AD + 0.5*BD $= 30, % R1 R2
AC + 0.5*BC + AE +BE $= 55, % R1 E
AD + 0.5*BD $= 30, % R2 D
0.5*BC + 0.5*BD $= 15, % R3 R4
AC + 0.5*BC $= 15, % E C
0.5*BC $= 5, % R4 C
0.5*BD $= 10, % R4 D
```

Solve for Different Flows

```
min(AC,MinAC),max(AC,MaxAC),
min(AD,MinAD),max(AD,MaxAD),
min(BC,MinBC),max(BC,MaxBC),
min(BD,MinBD),max(BD,MaxBD),
min(AE,MinAE),max(AE,MaxAE),
min(BE,MinBE),max(BE,MaxBE),
...
```


Results of Analysis

	C	D	E
A	10	20	20
B	10	20	20

Problem solved, no?

Benchmark Problems

Network	Routers	PoPs	Lines	Lines/router
dexa	51	24	59	1.15
as1221	108	57	153	1.41
as1239	315	44	972	3.08
as1755	87	23	161	1.85
as3257	161	49	328	2.03
as3967	79	22	147	1.86
as6461	141	22	374	2.65

Cork Constraint Computation Centre

TFA Result for Benchmarks

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time (sec)
dexa	0	2310.65	1190	11
as1221	0.09	8398.64	11556	1318
as1239	n/a	n/a	n/a	n/a
as1755	0.15	6255.31	7482	699
as3257	0.04	12260.03	25760	12389
as3967	0.1	5387.10	6162	500
as6461	0.28	8688.39	19740	8676

Cork Constraint Computation Centre

Reduce Problem Size

- Pop Level Analysis
- Only consider flows between PoPs, not routers
- Local area connections typically not bottlenecks
- Modelling routing can be tricky

PoP Level Results

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time (sec)
dexa	0	1068.37	557	5
as1221	0.24	2964.93	3205	424
as1239	0.63	1401.72	1931	101359
as1755	0.66	1263.28	526	103
as3257	0.30	2028.73	2378	2052
as3967	0.1	1209.37	483	90
as6461	1.47	951.41	481	768

Increase Accuracy

- LSP Counters
  - In MPLS networks only, provide improved resolution
  - Implementation buggy, not all counters can be used
- Netflow
  - Collect end-to-end flow information in router
  - Impact on router (memory)
  - Impact on network (data aggregation)

TFA with LSP Counters

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time (sec)
dexa	30.35	249.71	1190	7
as1221	9.94	685.37	11556	885
as1239	10.74	1151.03	98910	72461
as1755	25.29	269.30	7482	397
as3257	23.77	425.67	25760	5121
as3967	24.47	300.17	6162	275
as6461	19.43	477.44	19740	2683

PoP TFA with LSP Counters

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time (sec)
dexa	60.62	145.85	557	3
as1221	28.49	499.16	3205	271
as1239	33.36	211.84	1931	2569
as1755	50.33	169.37	526	46
as3257	36.82	249.16	2378	640
as3967	40.72	182.97	483	36
as6461	34.05	210.93	481	136

What now?

- Choose some particular solution?
- Which one? How to validate assumptions?
- Massively under-constrained problem
  - $|N|^2$ variables
  - $|E| + 2|N|$ constraints
  - $2|N|^2$ queries
- Ill-conditioned even after error correction
- Aggregation helps
  - We are usually not interested in individual flows
  - We want to use the TM to investigate something else

Resilience Analysis

- How much capacity is needed to survive all reasonable failures?
- Use normal state as starting point
- Consider routing in each failure case
- Aggregate flows in rerouted network
- Calculate bounds on traffic in failure case

Traffic Placement
Capacity Management
Other Problems
Bandwidth Protection
Bandwidth on Demand
Resilience Analysis

Model: Resilience Analysis

$$\forall e \in \mathbf{E}: \quad \min_{\{F_{ij}\}} / \max_{\{F_{ij}\}} \sum_{i,j \in \mathbf{N}} \bar{r}_{ij}^{e} F_{ij}$$

st.

$$\forall e \in \mathbf{E}: \quad \sum_{i,j \in \mathbf{N}} r_{ij}^{e} F_{ij} = \mathtt{traf}(e)$$

$$\forall i \in \mathbf{N}: \quad \sum_{j \in \mathbf{N}} F_{ij} = \mathtt{ext}^{in}(i)$$

$$\forall j \in \mathbf{N}: \quad \sum_{i \in \mathbf{N}} F_{ij} = \mathtt{ext}^{out}(j)$$

$$F_{ij} \geq 0$$

Resilience Analysis

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time (sec)	Cases
dexa	68.91	108.25	3503	57	59
as1221	85.75	102.60	14191	2869	153
as1239	*92.53*	*102.64*	*4499*	*44205*	*10*
as1755	92.82	105.39	8409	1815	161
as3257	93.69	103.15	31093	39934	328
as3967	91.60	108.79	9090	1635	141
as6461	96.51	103.44	24808	20840	374

Results over 100 runs

Network	lower bound/simul		upper bound/ simul	
	average	stdev	average	stdev
dexa	91.50	0.14	108.28	0.16
as1755	88.65	0.11	106.08	0.056
as3967	94.08	0.073	106.88	0.091
as1221	87.34	0.10	102.05	0.025

Results with LSP counters

Network	$\frac{Low}{Simul}$ (%)	$\frac{High}{Simul}$ (%)	Obj	Time	Cases
dexa	97.76	101.33	3503	36	59
as1221	98.15	100.69	14191	1840	153
as1239	*99.37*	*100.38*	*4499*	*3974*	*10*
as1755	99.28	100.66	8409	964	161
as3257	99.41	100.44	31093	13381	328
as3967	98.88	101.00	9090	819	147
as6461	99.44	100.52	24808	8006	374

Results over 100 runs (with LSP Counters)

Network	lower bound/simul		upper bound/ simul	
	average	stdev	average	stdev
dexa	99.60	0.029	100.33	0.025
as1755	99.31	0.016	100.63	0.015
as3967	99.41	0.014	100.61	0.014
as1221	98.10	0.025	100.57	0.010

Perspectives

- High polynomial complexity
- Possible to reduce number of queries
  - Small differences between failure cases
  - Many queries are identical or dominated
- Possible to reduce size of problem dramatically
- Integrate multiple measurements in one model
- Which other problems can we solve without explicit TM?

Cork Constraint Computation Centre

Problem

- Which links should be used to build network structure?
- Link speed is related to cost
- Model simple generalization of path finding
- Assumptions about routing in target network?

Cork Constraint Computation Centre

Model

$$\min_{\{X_{de}, W_{ie}\}} \sum_{e \in \mathbf{E}} \sum_{1 \leq i \leq \texttt{alt}(e)} \texttt{cost}(i,e) W_{ie}$$

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -1 & n = \texttt{dest}(d) \\ 1 & n = \texttt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall e \in \mathbf{E}: \quad \sum_{d \in \mathbf{D}} \texttt{bw}(d) X_{de} \leq \sum_{1 \leq i \leq \texttt{alt}(e)} \texttt{cap}(i,e) W_{ie}$$

$$\forall e \in \mathbf{E}: \quad \sum_{1 \leq i \leq \texttt{alt}(e)} W_{ie} = 1$$

$$W_{ie} \in \{0,1\}$$

$$X_{de} \in \{0,1\}$$

Issues

- Real-life problem not easily modelled
- Possible choices/costs not easily obtained (outside US)
- Choices often are inter-related
- Package deals by providers
- Some regions don’t allow any flexibility at all

Problem

- How to set weights in IGP to avoid bottlenecks?
- Easy to beat default values
- Single/equal cost paths required/allowed/forbidden?

Model

$$\min_{\{Y_{id},W_e\}} \max_{e\in\mathbf{E}} \frac{1}{\mathtt{cap}(e)} \sum_{d\in\mathbf{D}} \mathtt{bw}(d) \sum_{1\leq i\leq \mathtt{path}(d)} h_{id}^e Y_{id}$$

st.

$$\forall d \in \mathbf{D}: \sum_{1\leq i\leq \mathtt{path}(d)} Y_{id} = 1$$

$$\forall d \in \mathbf{D}, 1 \leq i \leq \mathtt{path}(d): \quad P_{id} = \sum_{e\in\mathbf{E}} h_{id}^e W_e$$

$$\forall d \in \mathbf{D}, 1 \leq i,j \leq \mathtt{path}(d): \quad P_{id} = P_{jd} \implies Y_{id} = Y_{jd} = 0$$

$$\forall d \in \mathbf{D}, 1 \leq i,j \leq \mathtt{path}(d): \quad P_{id} < P_{jd} \implies Y_{jd} = 0$$

$$Y_{id} \in \{0, 1\}$$

$$\text{integer} \quad W_e \geq 1$$

$$P_{id} \geq 0$$

Solution Methods

- Methods tested at IC-Parc
  - Branch and price
  - Tabu search
  - Set constraints
- Very hard to compete with (guided) local search

Further Reading

H. Simonis. Constraint Applications in Networks. Chaper 25 in F. Rossi, P van Beek and T. Walsh: Handbook of Constraint Programming. Elsevier, 2006.

Summary

- Network problems can be solved competitively by constraint techniques.
- Hybrid methods required, simple Finite Domain models usually don't work.
- Constraint based tools commercial reality.
- Open Problems
  - How to make this easier to develop?
  - How to make this more stable to solve?